Information Sciences 220 (2013) 492–506

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

# Variations of biogeography-based optimization and Markov analysis

Haiping Ma [a,c,d,*], Dan Simon [b], Minrui Fei [c,d], Zhikun Xie [a]

[a] Department of Electrical Engineering, Shaoxing University, Shaoxing, Zhejiang, China
[b] Department of Electrical and Computer Engineering, Cleveland State University, Cleveland, OH, USA
[c] Shanghai Key Laboratory of Power Station Automation Technology, Shanghai University, Shanghai, China
[d] School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, China

ARTICLE INFO

*Article history:*
Received 15 September 2011
Received in revised form 29 June 2012
Accepted 15 July 2012
Available online 25 July 2012

*Keywords:*
Biogeography-based optimization
Evolutionary algorithms
Population distribution
Markov chain
Real-world optimization problems

ABSTRACT

Biogeography-based optimization (BBO) is a new evolutionary algorithm that is inspired by biogeography. Previous work has shown that BBO is a competitive optimization algorithm, and it demonstrates good performance on various benchmark functions and real-world optimization problems. Motivated by biogeography theory and previous results, three variations of BBO migration are introduced in this paper. We refer to the original BBO algorithm as partial immigration-based BBO. The new BBO variations that we propose are called total immigration-based BBO, partial emigration-based BBO, and total emigration-based BBO. Their corresponding Markov chain models are also derived based on a previously-derived BBO Markov model. The optimization performance of these BBO variations is analyzed, and new theoretical results that are confirmed with simulation results are obtained. Theoretical results show that total emigration-based BBO and partial emigration-based BBO perform the best for three-bit unimodal problems, partial immigration-based BBO performs the best for three-bit deceptive problems, and all these BBO variations have similar results for three-bit multimodal problems. Performance comparison is further investigated on benchmark functions with a wide range of dimensions and complexities. Benchmark results show that emigration-based BBO performs the best for unimodal problems, and immigration-based BBO performs the best for multimodal problems. In addition, BBO is compared with a stud genetic algorithm (SGA), standard particle swarm optimization (SPSO 07), and adaptive differential evolution (ADE) on real-world optimization problems. The numerical results demonstrate that BBO outperforms SGA and SPSO 07, and performs similarly to ADE for the real-world problems.



## 1. Introduction

Mathematical models of biogeography describe the immigration and emigration of species between habitats. Biogeography-based optimization (BBO) was first presented in 2008 [28] and is an extension of biogeography to evolutionary algorithms (EAs). BBO has demonstrated good performance on unconstrained and constrained benchmark functions [8,9,19]. It has also been applied to real-world optimization problems, including sensor selection [28], economic load dispatch [3], satellite image classification [21], power system optimization [24], and others.

Like other EAs [1,10,12,27,37], BBO probabilistically shares information between candidate solutions. We use the shorthand notation "solution" to refer to a candidate solution. In BBO, each solution is comprised of a set of independent variables, also called features. Each solution immigrates features from other solutions based on its immigration rate, and emigrates

* Corresponding author at: Department of Electrical Engineering, Shaoxing University, Shaoxing, Zhejiang, China. Tel.: +86 575 88345673.
*E-mail addresses:* Mahp@usx.edu.cn (H. Ma), d.j.simon@csuohio.edu (D. Simon), mrfei@staff.shu.edu.cn (M. Fei).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.07.007

features to other solutions based on its emigration rate. In the original BBO paper [28], immigration rates are first used to decide whether or not immigrate features to a given solution, and then emigration rates are used to choose the emigrating solution. However, there are other methods that could be used to implement migration, and these other methods are introduced and analyzed in this paper. For example, one of our proposals here is to first use emigration rates to decide whether or not to emigrate features from a given solution, and then use immigration rates to select the immigrating solution.

Markov models have been a valuable theoretical tool to analyze EAs, including simple genetic algorithms [7,20,25,33–36] and simulated annealing [16]. Markov models have already been derived for BBO [29,30], along with Markov model comparisons between BBO and genetic algorithms [31]. A Markov chain is a random process that has a discrete set of possible states $s_i (i = 1, 2, \ldots, T)$. The probability that the system transitions from state $s_i$ to $s_j$ is given by the probability $p_{ij}$, which is called a transition probability. The $T \times T$ matrix $P = [p_{ij}]$ is called the transition matrix. A Markov state in [30] represents a BBO population distribution. Each state describes how many individuals there are at each point of the search-space. $p_{ij}$ is the probability that the population transitions from the $i$th population distribution to the $j$th population distribution in one generation. This paper uses Markov models to analyze our proposed BBO variations.

Section 2 gives a simple description of the original BBO algorithm, and introduces three variations of BBO. Section 3 introduces and analyzes the Markov chain models for BBO and its variations. Section 4 compares these BBO variations based on simulation results. Some concluding remarks and directions for future work are provided in Section 5.

## 2. Variations of biogeography-based optimization

This section presents an overview of the original BBO algorithm (Section 2.1), and then introduces three variations of BBO (Section 2.2).

### 2.1. Biogeography-based optimization

Suppose that we have an optimization problem and some candidate solutions. Each candidate solution is comprised of a set of features, which are similar to genes in GAs. A good solution is analogous to a biological habitat with a high habitat suitability index (HSI). This corresponds to a geographical area that is well suited for hosting biological species. In optimization problems, HSI corresponds to the goodness of a BBO solution, and is called fitness in standard EAs notation. A poor solution is analogous to a habitat that is not well suited for hosting biological species. High fitness BBO solutions correspond to biological habitats with a large number of species, and low fitness BBO solutions correspond to habitats with a small number of species. High fitness solutions are more likely to share their features with other solutions (emigration), and low fitness solutions are more likely to accept shared features from other solutions (immigration). Similar to other EAs, BBO consists of two main steps: information sharing (which is implemented with migration in BBO) and mutation.

**Migration** is a probabilistic operation. The migration rates of each solution are used to probabilistically share features between solutions. For each feature of a given solution $y_k$, the immigration rate $\lambda_k$ of $y_k$ is used to probabilistically decide whether or not to immigrate. If immigration is selected, then the emigrating solution $y_j$ is probabilistically chosen based on the emigration rate $\mu_j$. Migration is written as

$$y_k(s) \leftarrow y_j(s) \tag{1}$$

where $s$ is a solution feature. Immigration rates and emigration rates are based on migration curves, such as the linear migration curves in Fig. 1, where the maximum immigration rate and maximum emigration rate are both equal to 1. Nonlinear curves are discussed in [17,18].

**Mutation** is a probabilistic operator that randomly modifies a solution feature. The purpose of mutation is to increase diversity among the population.

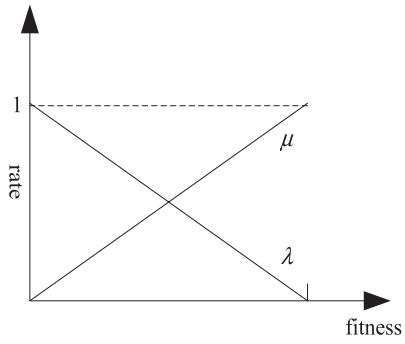


**Fig. 1.** Linear migration curves for BBO. $\lambda$ is the immigration rate and $\mu$ is the emigration rate, and we assume that the maximum immigration rate and maximum emigration rate are both equal to 1.

**Algorithm 1.** One generation of the BBO algorithm, where $N$ is the population size. $y$ is the entire population of candidate solutions, $y_k$ is the $k$th candidate solution, and $y_k(s)$ is the $s$th feature of $y_k$. This algorithm is called partial immigration-based BBO

```
For each solution y_k, define emigration rate μ_k proportional to fitness of y_k, where μ_k ∈ [0,1]
For each solution y_k, define immigration rate λ_k = 1 − μ_k
z ← y
For each solution z_k (k = 1 to N)
    For each solution feature s
        Use λ_k to probabilistically decide whether to immigrate to z_k
        If immigrating then
          Use {μ_i} to probabilistically select the emigrating solution y_j
          z_k(s) ← y_j(s)
        End if
    Next solution feature
    Probabilistically decide whether to mutate z_k
Next solution
y ← z
```

A description of one generation of BBO is given in Algorithm 1. Migration and mutation of the entire population take place before any of the solutions are replaced in the population, which requires the use of the temporary population $z$ in the algorithm. In Algorithm 1, the statement "use $\lambda_k$ to probabilistically decide whether to immigrate to $z_k$" can be implemented with the following logic, where rand (0, 1) is a random number uniformly distributed between 0 and 1:

```
If λ_k < rand (0, 1) then
        Immigration Flag = true
else
        Immigration Flag = false
end if
```

Also in Algorithm 1, the statement "Use $\{\mu_i\}$ to probabilistically select the emigrating solution $y_j$" can be implemented with any fitness-based selection method (since $\mu_i$ is proportional to the fitness of $y_i$). For instance, we could use tournament selection by randomly choosing two or more solutions for a tournament, and then selecting $y_j$ as the fittest solution in the tournament. In this paper, as in most other BBO implementations, we use $\{\mu_i\}$ in a roulette-wheel algorithm so that the probability that each individual $y_i$ is selected for emigration is proportional to its emigration rate $\mu_i$.

### 2.2. Variations of BBO

The BBO algorithm presented in Algorithm 1 is called partial immigration-based BBO. The word "partial" means that only one solution feature is considered at a time for immigration. That is, for solution $z_k$, $\lambda_k$ is tested against a random number once for every feature to decide whether or not to replace that feature in $z_k$. The term "immigration-based" means that $\lambda_k$ is first used to decide whether or not to immigrate to $z_k$; then the $\mu_j$ variables are used to choose the emigrating solution, but only if immigration was selected as described in the previous subsection.

However, there are also other ways that could be used to implement migration. Instead of testing $\lambda_k$ against a random number once for each solution feature, we could test $\lambda_k$ against a random number only once for each solution, and then if immigration were selected, we could replace all of the solution features in $z_k$. We call this total immigration-based BBO. One generation of total immigration-based BBO is described in Algorithm 2.

**Algorithm 2.** One generation of total immigration-based BBO, where $N$ is the population size. $y$ is the entire population of candidate solutions, $y_k$ is the $k$th candidate solution, and $y_k(s)$ is the $s$th feature of $y_k$

```
For each solution y_k, define emigration rate μ_k proportional to fitness of y_k, where μ_k ∈ [0, 1]
For each solution y_k, define immigration rate λ_k = 1 − μ_k
z ← y
For each solution z_k (k = 1 to N)
        Use λ_k to probabilistically decide whether to immigrate to z_k
        If immigrating then
```

```
    For each solution feature s
      Use {μ_i} to probabilistically select the emigrating solution y_j
      z_k(s) ← y_j(s)
    Next solution feature
  End if
  Probabilistically decide whether to mutate z_k
Next solution
y ← z
```

As a third option, we could first use $\mu_k$ to decide whether or not to emigrate a solution feature from a given solution. Then, if emigration were selected, the $\lambda_j$ values could be used to select the immigrating solution. This idea results in partial emigration-based BBO. One generation of partial emigration-based BBO is described in Algorithm 3.

Finally, instead of testing $\mu_k$ against a random number once for each solution feature, we could test $\mu_k$ against a random number only once for each solution, and then if emigration were selected, all solution features could be emigrated from $y_k$. This is called total emigration-based BBO. One generation of total emigration-based BBO is described in Algorithm 4.

**Algorithm 3.** One generation of partial emigration-based BBO, where $N$ is the population size. $y$ is the entire population of candidate solutions, $y_k$ is the $k$th candidate solution, and $y_k(s)$ is the $s$th feature of $y_k$

```
For each solution y_k, define emigration rate μ_k proportional to fitness of y_k, where μ_k∈ [0, 1]
For each solution y_k, define immigration rate λ_k = 1 − μ_k
z ← y
For each solution y_r(r = 1 to N)
    For each solution feature s
      Use μ_r to probabilistically decide whether to emigrate from y_r
      If emigrating then
        Use {λ_j} to probabilistically select the immigrating solution z_k
        z_k(s) ← y_r(s)
      End if
    Next solution feature
Next solution
For each z_k in the population, probabilistically decide whether to mutate z_k
Y ← z
```

**Algorithm 4.** One generation of total emigration-based BBO, where $N$ is the population size. $y$ is the entire population of candidate solutions, $y_k$ is the $k$th candidate solution, and $y_k(s)$ is the $s$th feature of $y_k$

```
For each solution y_k, define emigration rate μ_k proportional to fitness of y_k, where μ_k∈ [0, 1]
For each solution y_k, define immigration rate λ_k = 1 − μ_k
z ← y
For each solution y_r(r = 1, ... ,N)
    Use μ_r to probabilistically decide whether to emigrate from y_r
    If emigrating then
      For each solution feature s
        Use {λ_j} to probabilistically select the immigrating solution z_k
        z_k(s) ← y_r(s)
      Next solution feature
    End if
Next solution
For each z_k in the population, probabilistically decide whether to mutate z_k
y ← z
```

These four combinations of partial/total and immigration/emigration are inspired by the original philosophy of BBO migration. Therefore, we derive Markov models for each of the three new variations in the following section, and we test their performances in Section 4.

## 3. Markov analysis

This section presents an overview of a Markov model of partial immigration-based BBO (Section 3.1), and then derives Markov models for the three new variations of BBO (Section 3.2).

### 3.1. Markov analysis of partial immigration-based BBO

In [30] a partial immigration-based BBO Markov chain model is derived for Algorithm 1. This subsection reviews this Markov model. A Markov model of BBO provides us with the transition probability $p_{ij}$ from the $i$th population distribution to the $j$th population distribution. In BBO, two main steps, migration and mutation, are significant, so the transition probability includes the migration probability and the mutation probability for one generation.

Consider a problem whose candidate solutions $\{x_1, \ldots, x_n\}$ are in a binary search space. The set of candidate solutions is the set of all bit strings $x_i$ consisting of $q$ bits each. Therefore, the cardinality of the search space is $n = 2^q$. Use $N$ to denote the population size, and use $v$ to denote the population count vector, where the component $v_i$ is the number of candidate solutions $x_i$ in the population. Note that

$$\sum_{i=1}^{n} v_i = N \tag{2}$$

We use $y_k$ to denote the $k$th individual in the population:

$$\text{Population} = \{y_1, \cdots y_N\} = \{\underbrace{x_1, x_1, \cdots, x_1}_{v_1 \text{ copies}}, \underbrace{x_2, x_2, \cdots, x_2}_{v_2 \text{ copies}}, \cdots \underbrace{x_n, x_n, \cdots, x_n}_{v_n \text{ copies}}\} \tag{3}$$

where the $y_k$ individuals have been ordered to group identical individuals. $s$ is used to denote the $s$th feature (that is, the $s$th bit) of a solution. According to [30], for partial immigration-based BBO, the probability that an immigration opportunity for the $s$th solution feature of $y_k$ during generation $t$ results in a new solution feature $x_i(s)$ at generation $t+1$, is the following.

$$\begin{aligned}\Pr(y_{k,t+1}(s) = x_i(s)) &= \Pr(\text{no immigration to } y_{k,t})\Pr(y_{k,t+1}(s) \\ &= x_i(s)|\text{no immigration}) + \Pr(\text{immigration to } y_{k,t})\Pr(y_{k,t+1}(s) = x_i(s)|\text{immigration}) \\ &= (1-\lambda_{m(k)})1_0(x_{m(k)}(s) - x_i(s)) + \lambda_{m(k)}\frac{\sum_{j\in\varsigma_i(s)} v_j\mu_j}{\sum_{j=1}^{n} v_j\mu_j}\end{aligned} \tag{4}$$

where $\mathbf{1}_0$ is the indicator function on the set $\{0\}$, and

$$y_k = x_{m(k)} \text{ for } k = 1, \cdots, N \tag{5}$$

where $m(k)$ is defined as

$$m(k) = \min\ r \text{ such that } \sum_{i=1}^{r} v_i \geqslant k \tag{6}$$

The relationship between $y_k$ and $x_{m(k)}$ is depicted in (3). The notation $\varsigma_i(s)$ in (4) denotes the set of candidate solution indices $j$ such that the $s$th bit of $x_j$ is equal to the $s$th bit of $x_i$. That is,

$$\varsigma_i(s) = \{j : x_j(s) = x_i(s)\} \tag{7}$$

We know from (4) that the total migration probability includes two parts: the probability that immigration did not occur, and the probability that immigration did occur. If the $s$th feature of $y_k$ is not selected for immigration during generation $t$, then

$$y_k(s)_{t+1} = x_{m(k)}(s), \quad (\text{if no immigration to } y_{k,t}) \tag{8}$$

If the $s$th feature of $y_k$ is selected for immigration during generation $t$, the probability that $y_k(s)_{t+1}$ is equal to $x_i(s)$ is proportional to the sum of the emigration rates of all individuals whose $s$th feature is equal to $x_i(s)$. This probability can be written as

$$\Pr(y_k(s)_{t+1} = x_i(s)|\text{immigration}) = \frac{\sum_{j\in\varsigma_i(s)} v_j\mu_j}{\sum_{j=1}^{n} v_j\mu_j} \tag{9}$$

Eqs. (8) and (9) are combined with the fact that the probability of immigration to $y_k(s)$ is equal to $\lambda_{m(k)}$ to obtain (4).

$P_{ki}(v)$ is used to denote the probability that immigration results in $y_{k,t+1} = x_i$, given that the population distribution is described by the population count vector $v$. $P_{ki}(v)$ can be written as

$$P_{ki}(v) = \Pr(y_{k,t+1} = x_i) = \prod_{s=1}^{q}\left[(1-\lambda_{m(k)})1_0(x_{m(k)}(s) - x_i(s)) + \lambda_{m(k)}\frac{\sum_{j\in\varsigma_i(s)} v_j\mu_j}{\sum_{j=1}^{n} v_j\mu_j}\right] \tag{10}$$

where $q$ is the number of bits in each solution. Note that $P_{ki}(v)$ can be computed for each $k \in [1,N]$ and each $i \in [1,n]$ in order to form the $N \times n$ matrix $P(v)$. The $k$th row of $P(v)$ corresponds to the $k$th iteration of the outer loop in Algorithm 1 (there are $N$ iterations of the outer loop in Algorithm 1). The $i$th column of $P(v)$ corresponds to the probability of obtaining solution $x_i$ during each outer loop iteration; that is, $P_{ki}(v)$ gives the probability of obtaining the $i$th outcome on the $k$th immigration trial.

Only migration is included in (10). Next we include the possibility of mutation. $U$ is used to denote the $n \times n$ mutation matrix, where $U_{ji}$ is the probability that $x_j$ mutates to $x_i$. The probability that the $k$th immigration trial, followed by mutation, results in $x_i$ is denoted as $P^{(2)}_{ki}(v)$. This can be written as

$$\begin{aligned} P^{(2)}_{ki}(v) &= \sum_{j=1}^{n} P_{kj}(v) U_{ji} \\ P^{(2)}(v) &= P(v)U \end{aligned} \tag{11}$$

where the elements of $P(v)$ are given in (10). $P^{(2)}(v)$ contains the probabilities when both migration and mutation are considered. $u$ is defined as the population count vector after migration and mutation are completed for a given generation, where the component $u_i$ is the number of solutions $x_i$ in the population. The transition probability $\Pr(u|v)$ that we obtain a population count vector $u$ after one generation, given that we started with a population count vector $v$, can be obtained from the multinomial theorem [30] as

$$\Pr(u|v) = \sum_{J \in Y} \prod_{k=1}^{N} \prod_{i=1}^{n} \left[ P^{(2)}_{ki}(v) \right]^{J_{ki}}, \text{where } Y = \left\{ J \in R^{N \times n} : J_{ki} \in \{0,1\}, \sum_{i=1}^{n} J_{ki} = 1 \text{ for all } k, \sum_{k=1}^{N} J_{ki} = u_i \text{ for all } i \right\} \tag{12}$$

Eq. (12) can be used to find the transition matrix for partial immigration-based BBO with migration and mutation.

The Markov transition matrix, which we denote as $Q$, is obtained by computing (12) for each possible $v$ and each possible $u$. Each element of $Q$ will give the transition probability from one population count vector $v$ to another population count vector $u$ after one generation. Note that $Q$ is a $T \times T$ matrix, where $T$ is the total number of possible populations. $T$ can be calculated by several different methods [30]. Once we calculate the transition matrix $Q$, a wealth of Markov tools [13] can be applied to find statistical properties of BBO, including the limiting probability (as the generation count approaches infinity) of each possible BBO population.

### 3.2. Markov analysis of variations of BBO

The above subsection summarized the partial immigration-based BBO Markov chain model of Algorithm 1, which considers the immigration of each solution feature as separate probabilistic trials. This subsection derives new Markov models for three variations of BBO.

#### 3.2.1. Total immigration-based BBO

Total immigration-based BBO (Algorithm 2) bases migration on the immigration rate for each solution, and probabilistically decides whether or not to immigrate all solution features to a given solution. This is different from partial immigration-based BBO (Algorithm 1), which considers immigration of one solution feature at a time. For total immigration-based BBO, given that the population distribution at generation $t$ is equal to $v$, the probability $P_{ki}(v)$ that immigration results in $y_{k,t+1} = x_i$ at generation $t+1$ can be obtained as

$$\begin{aligned} P_{ki}(v) &= \Pr(y_{k,t+1} = x_i) = \Pr(\text{no immigration to } y_{k,t}) \Pr(y_{k,t+1} \\ &\quad = x_i | \text{no immigration}) + \Pr(\text{immigration to } y_{k,t}) \Pr(y_{k,t+1} = x_i | \text{immigration}) \\ &= (1 - \lambda_{m(k)}) 1_0(x_{m(k)} - x_i) + \lambda_{m(k)} \prod_{s=1}^{q} \frac{\sum_{j \in \varsigma_i(s)} v_j \mu_j}{\sum_{j=1}^{n} v_j \mu_j} \end{aligned} \tag{13}$$

Note that the first term of the right side of Eq. (13) denotes the probability when immigration does not occur; that is, when $y_k$ is not selected for immigration. The second term on the right side of Eq. (13) denotes the probability when immigration occurs, and it is proportional to the product of the summed emigration rates of all solutions whose bits are equal to those of $x_i$.

Now suppose that the mutation probability is the same as that in partial immigration-based BBO. That is, $P^{(2)}_{ki}(v)$ is defined as in (11), except that we use $P_{ki}(v)$ from (13) instead of $P_{ki}(v)$ from (10). Then the transition matrix for total immigration-based BBO is calculated as shown in (12).

#### 3.2.2. Partial emigration-based BBO

Partial emigration-based BBO (Algorithm 3) bases migration on emigration rates for each solution, and probabilistically decides whether or not to emigrate each solution feature. If emigration is selected, the immigrating solution is probabilistically selected based on immigration rates. For all BBO variations, the probability that $y_k$ is equal to some specific value includes the probability that immigration does not occur and the probability that immigration occurs. Suppose $r$ denotes the current emigration trial (there are $N$ emigration trials of the outer loop in Algorithm 3). If the $s$th feature of $y_k$ is not selected for immigration during generation $t$, then

$$y_k(s)_{t+1} = x_{m(k)}(s) \quad \text{(if no immigration to } y_{k,\,t}(s) \text{ on the } r\text{th emigration trial)} \tag{14}$$

To calculate the probability that immigration does not occur, first consider the emigration probability of $x_{m(r)}$ on the $r$th emigration trial, which can be written as

$$\Pr(\text{emigration on the } r\text{th emigration trial}) = \mu_{m(r)} \tag{15}$$

where the meaning of $m(r)$ is similar to $m(k)$ in (6). The immigration probability of $y_k$ is proportional to its immigration rate. So the probability that $x_{m(r)}(s)$ immigrates to $y_k(s)$ on the $r$th emigration trial can be written as

$$\begin{aligned}&\Pr(\text{immigration to } y_k \text{ on the } r\text{th emigration trial})\\ &= \Pr(\text{emigration on the } r\text{th emigration trial})\Pr(\text{immigration to } y_k|\text{emigration on the } r\text{th emigration trial})\\ &= \mu_{m(r)}\frac{\lambda_{m(k)}}{\sum_{j=1}^{n} v_j\lambda_j}\end{aligned} \tag{16}$$

The probability that immigration does not occur for $y_k(s)$ on the $r$th emigration trial can be written as

$$\begin{aligned}\Pr(\text{no immigration to } y_k \text{ on the } r\text{th emigration trial}) &= 1\\ &\quad - \Pr(\text{immigration to } y_k \text{ on the } r\text{th emigration trial})\\ &= 1 - \mu_{m(r)}\frac{\lambda_{m(k)}}{\sum_{j=1}^{n} v_j\lambda_j}\end{aligned} \tag{17}$$

However, if the $s$th feature of $y_k$ is selected for immigration on the $r$th emigration trial during generation $t$, then in order to have $y_k(s) = x_i(s)$, the $s$th feature of the emigrating solution $x_{m(r)}$ must be equal to the $s$th feature of $x_i$; that is,

$$y_k(s)_{t+1} = x_{m(r)}(s) = x_i(s) \quad \text{(if immigration to } y_{k,t}(s) \text{ on the } r\text{th emigration trial)} \tag{18}$$

Eqs. (14)–(17) are combined to obtain the total migration probability of one bit of $y_k$ after $R$ emigration trials. Here use $R$ to denote the total number of emigration trials; this differs from $r$, which indicates a specific emigration trial. So the total migration probability can be written as

$$\begin{aligned}\Pr(y_{k,t+1}(s) = x_i(s) \text{ after } R \text{ emigration trials}) &= \Pr(\text{no immigration to } y_{k,t} \text{ on } R\text{th emigration trial})\Pr(y_{k,t+1}(s)\\ &= x_i(s) \text{ after } R-1 \text{ emigration trials}) + \Pr(\text{immigration to } y_{k,t} \text{ on } R\text{th emigration trial})\Pr(y_{k,t+1}(s)\\ &= x_i(s)|\text{immigration on } R\text{th emigration trial}) = \left(1 - \mu_{m(R)}\frac{\lambda_{m(k)}}{\sum_{j=1}^{n} v_j\lambda_j}\right)\Pr(x_{m(k)}(s)\\ &= x_i(s) \text{ after } (R-1) \text{ emigration trials}) + \left(\mu_{m(R)}\frac{\lambda_{m(k)}}{\sum_{j=1}^{n} v_j\lambda_j}\right)1_0(x_{m(R)}(s) - x_i(s))\end{aligned} \tag{19}$$

Note that the above equation applies to one bit in one individual after $R$ emigration trials. For example, consider the simple case of one emigration trial ($R = 1$). In this case, the probability that $y_k(s) = x_i(s)$ at generation $t + 1$ can be written as

$$\begin{aligned}\Pr(y_{k,t+1}(s) = x_i(s) \text{ after 1emigration trial}) &= \Pr(\text{no immigration to } y_{k,t}\text{on 1st emigration trial})\Pr(y_{k,t+1}(s)\\ &= x_i(s)|\text{no immigration}) + \Pr(\text{immigration to } y_{k,t} \text{ on 1st emigration trial})\Pr(y_{k,t+1}(s)\\ &= x_i(s)|\text{immigration on 1st emigration trial})\\ &= \left(1 - \mu_{m(1)}\frac{\lambda_{m(k)}}{\sum_{j=1}^{n} v_j\lambda_j}\right)1_0(x_{m(k)}(s) - x_i(s)) + \left(\mu_{m(1)}\frac{\lambda_{m(k)}}{\sum_{j=1}^{n} v_j\lambda_j}\right)1_0(x_{m(1)}(s) - x_i(s))\end{aligned} \tag{20}$$

To express this in a more compact form, we introduce the notation $\eta_{m(k)}$ for $\lambda_{m(k)}/\sum_{j=1}^{n} v_j\lambda_j$ in the next equations. So after two emigration trials we obtain

$$\begin{aligned}\Pr(y_{k,t+1}(s) = x_i(s) \text{ after 2 emigration trials}) &= \Pr(\text{no immigration to } y_{k,t} \text{ on 2nd emigration trial})\Pr(y_{k,t+1}(s)\\ &= x_i(s) \text{ after 1 emigration trial}) + \Pr(\text{immigration to } y_{k,t} \text{ on 2nd emigration trial})\Pr(y_{k,t+1}(s)\\ &= x_i(s)|\text{immigration on 2nd emigration trial}) = (1 - \mu_{m(2)}\eta_{m(k)})\Pr(x_{m(k)}(s)\\ &= x_i(s)\text{after 1 emigration trial}) + (\mu_{m(2)}\eta_{m(k)})1_0(x_{m(2)}(s) - x_i(s))\\ &= (1 - \mu_{m(2)}\eta_{m(k)})((1 - \mu_{m(1)}\eta_{m(k)})1_0(x_{m(k)}(s) - x_i(s)) + (\mu_{m(1)}\eta_{m(k)})1_0(x_{m(1)}(s) - x_i(s)))\\ &\quad + (\mu_{m(2)}\eta_{m(k)})1_0(x_{m(2)}(s) - x_i(s))\\ &= (1 - \mu_{m(2)}\eta_{m(k)})(1 - \mu_{m(1)}\eta_{m(k)})1_0(x_{m(k)}(s) - x_i(s)) + (1 - \mu_{m(2)}\eta_{m(k)})(\mu_{m(1)}\eta_{m(k)})1_0(x_{m(1)}(s) - x_i(s))\\ &\quad + (\mu_{m(2)}\eta_{m(k)})1_0(x_{m(2)}(s) - x_i(s))\end{aligned} \tag{21}$$

Note that the first term of the right side of the above equation denotes the probability when immigration does not occur on either of the two emigration trials, the second term denotes the probability when immigration occurs on the first emigration trial but not on the second emigration trial, and the third term denotes the probability when immigration occurs on the second emigration trial. After $N$ emigration trials (recall that the population size is $N$), we can use induction to see that the probability can be written as

$$\begin{aligned}
\Pr(y_{k,t+1}(s) = x_i(s)\text{after } N \text{ emigration trials}) &= \Pr(\text{no immigration to } y_{k,t} \text{ on } N\text{th emigration trial})\Pr(y_{k,t+1}(s) \\
&= x_i(s)\text{after } N-1 \text{ emigration trials}) + \Pr(\text{immigration to } y_{k,t} \text{ on } N\text{th emigration trial})\Pr(y_{k,t+1}(s) \\
&= x_i(s)|\text{immigration on } N\text{th emigration trial}) = (1-\mu_{m(N)}\eta_{m(k)})\Pr(x_{m(k)}(s) \\
&= x_i(s)\text{after}(N-1) \text{ emigration trials}) + (\mu_{m(N)}\eta_{m(k)})1_0(x_{m(N)}(s) - x_i(s)) \\
&= (1-\mu_{m(N)}\eta_{m(k)})(1-\mu_{m(N-1)}\eta_{m(k)})\cdots(1-\mu_{m(3)}\eta_{m(k)})(1-\mu_{m(2)}\eta_{m(k)})(1-\mu_{m(1)}\eta_{m(k)})1_0(x_{m(k)}(s) \\
&\quad - x_i(s)) + (1-\mu_{m(N)}\eta_{m(k)})(1-\mu_{m(N-1)}\eta_{m(k)})\cdots(1-\mu_{m(3)}\eta_{m(k)})(1-\mu_{m(2)}\eta_{m(k)})(\mu_{m(1)}\eta_{m(k)})1_0(x_{m(1)}(s) \\
&\quad - x_i(s)) + (1-\mu_{m(N)}\eta_{m(k)})(1-\mu_{m(N-1)}\eta_{m(k)})\cdots(1-\mu_{m(3)}\eta_{m(k)})(\mu_{m(2)}\eta_{m(k)})1_0(x_{m(2)}(s) - x_i(s))\cdots \\
&\quad + (\mu_{m(N)}\eta_{m(k)})1_0(x_{m(N)}(s) - x_i(s))
\end{aligned} \tag{22}$$

Note that the first term of the right side of the above equation denotes the probability when immigration does not occur on any of the $N$ emigration trials, the second term denotes the probability when immigration occurs on the first emigration trial but none of the later emigration trials, the third term denotes the probability when immigration occurs on the second emigration trial but none of the later trials, and so on. A more compact form of (22) is

$$\begin{aligned}
\Pr(y_{k,t+1}(s) &= x_i(s) \text{ after } N \text{ emigration trials}) \\
&= \prod_{l=1}^{N}(1-\mu_{m(l)}\eta_{m(k)})1_0(x_{m(k)}(s) - x_i(s)) + \sum_{L=1}^{N-1}\left[\prod_{l=L}^{N-1}(1-\mu_{m(l+1)}\eta_{m(k)})(\mu_{m(L)}\eta_{m(k)})1_0(x_{m(L)}(s) - x_i(s))\right] \\
&\quad + (\mu_{m(N)}\eta_{m(k)})1_0(x_{m(N)}(s) - x_i(s))
\end{aligned} \tag{23}$$

We again use $P_{ki}(v)$ to denote the probability that immigration results in $y_{k,t+1} = x_i$, given that the population distribution at generation $t$ is equal to $v$. Recalling that there are $q$ bits in each solution, this can be written as

$$P_{ki}(v) = \Pr(y_{k,t+1} = x_i) = \prod_{s=1}^{q}(\Pr(y_{k,t+1}(s) = x_i(s) \text{ after N emigration trials})) \tag{24}$$

The incorporation of mutation probability is the same as for partial immigration-based BBO, so $P_{ki}^{(2)}(v)$ is defined the same as in (11), except that we use $P_{ki}(v)$ from (24). Then the transition matrix for total immigration-based BBO is calculated as shown in (12).

#### 3.2.3. Total emigration-based BBO

Total emigration-based BBO (Algorithm 4) bases migration on the emigration rate for each solution, and probabilistically decides whether or not to emigrate all solution features from each solution. This differs from partial emigration-based BBO, which considers emigration of one solution feature at a time. But in total emigration-based BBO, the immigration probability of one bit of $y_k$ after $R$ emigration trials is the same as in partial emigration-based BBO. This can be seen by carefully comparing Algorithms 3 and 4; the probability of immigrating to $z_k(s)$ is the same for both algorithms. Therefore, the Markov transition matrix is the same for total emigration-based BBO as it is for partial emigration-based BBO.

## 4. Simulation results

This section first looks at the performance of the four BBO algorithms of Section 2 using the Markov chain models derived in Section 3. This Markov-based comparison is presented in Section 4.1. Then we compare the performance of the four BBO algorithms on a set of commonly used benchmark functions in Section 4.2. Finally we compare BBO with several other optimization algorithms on some real-world optimization problems in Section 4.3.

### 4.1. Theoretical comparison

This subsection theoretically compares the original BBO with its variations. Most importantly in this section, we use simulation results to confirm the Markov models of the previous section. Of secondary importance, the simulation results provide some preliminary guidelines for when to use different BBO variations.

The four BBO algorithms were presented in Section 2 and include partial immigration-based BBO, total immigration-based BBO, partial emigration-based BBO, and total emigration-based BBO. The comparison in this subsection uses the Markov chain models derived in Section 3. For the remainder of this paper, partial immigration-based BBO and total

immigration-based BBO are called immigration-based BBO algorithms, and partial emigration-based BBO and total emigration-based BBO are called emigration-based BBO algorithms. Their limiting population distributions are compared using the results of Sections 3.1, 3.2.1, 3.2.2, and 3.2.3. The Markov transition matrices derived in Section 3 are used to obtain the probability, in the limit as the generation count approaches infinity, that the BBO population consists of a particular set of individuals. This approach is motivated by the fundamental limit theorem for regular Markov chains, which states that if the transition matrix is regular, there is a unique nonzero limiting probability for each state as time approaches infinity. The transition matrix of BBO has been proven to be regular when the mutation rate is nonzero [30]. Additional details about how to obtain BBO population probabilities from Markov transition matrices can be found in [29–31].

Test functions are limited to three-bit problems with a search space cardinality of eight and a population size of four due to the factorial increase of Markov matrix sizes with problem size. The three fitness functions that we examine are

$$\begin{aligned} f_1 &= (1 \quad 2 \quad 2 \quad 3 \quad 2 \quad 3 \quad 3 \quad 4) \\ f_2 &= (4 \quad 2 \quad 2 \quad 3 \quad 2 \quad 3 \quad 3 \quad 4) \\ f_3 &= (4 \quad 1 \quad 1 \quad 2 \quad 1 \quad 2 \quad 2 \quad 3) \end{aligned} \tag{25}$$

where fitness values are listed in binary order, so the first element of each fitness function corresponds to the bit string 000, the second element corresponds to the bit string 001, and so on. For example, for problem $f_1$, the bit string 000 has a fitness of 1, the bit string 001 has a fitness of 2, ..., and the bit string 111 has a fitness of 4. In (25), $f_1$ is a unimodal one-max problem in which the fitness of each bit string is proportional to the number of ones in the bit string; $f_2$ is a multimodal problem in which the fitness of each bit string is equal to those of the unimodal one-max problem, except that the bit string consisting of all zeros has the same fitness as the bit string consisting of all ones; and $f_3$ is a deceptive problem in which the fitness of each bit string is proportional to the number of ones in the bit string, except that the bit string of all zeros has the highest fitness. Note that all three problems are maximization problems. For all BBO variations, we use emigration rates $\mu_i = f_{ki}/5$ for $f_k$ ($k = 1, 2, 3$), where $f_{ki}$ is the fitness of the $i$th element of $f_k$. We use immigration rate $\lambda_i = 1 - \mu_i$, and we do not use elitism for any of the problems.

To confirm the BBO Markov chain models, we use simulation in this subsection with the number of generations equal to 20,000 to approximate an infinite number of generations. In addition, the parameters of all BBO variations used in the simulations are as follows: population size equal to 50, and 100 Monte Carlo simulations for each problem. Tables 1–3 show

**Table 1**
Optimization results for the three-bit unimodal one-max problem $f_1$. The table shows the probabilities of obtaining an all-optimal population and the probabilities of obtaining a no-optimal population using the BBO Markov models and simulations. In addition, CPU times for simulations are shown in the last row of the table. The best Markov performance is in **bold font** in each row.

| Mutation rate | Population count vector | Probability | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Partial immigration BBO (original BBO) | | Total immigration BBO | | Partial emigration BBO | | Total emigration BBO | |
| | | Markov | Simulation | Markov | Simulation | Markov | Simulation | Markov | Simulation |
| 0.1 | All optimal | 0.02452 | 0.02516 | 0.02583 | 0.02651 | **0.06302** | 0.06164 | **0.06302** | 0.06318 |
| | No optimal | 0.29985 | 0.29701 | 0.31083 | 0.32324 | **0.23619** | 0.23584 | **0.23619** | 0.23126 |
| 0.01 | All optimal | 0.53439 | 0.53142 | 0.53301 | 0.52443 | **0.75518** | 0.75962 | **0.75518** | 0.75104 |
| | No optimal | 0.11344 | 0.13093 | 0.11759 | 0.11609 | **0.03283** | 0.03425 | **0.03283** | 0.03628 |
| 0.001 | All optimal | 0.86053 | 0.85905 | 0.86012 | 0.85876 | **0.95428** | 0.95472 | **0.95428** | 0.95271 |
| | No optimal | 0.09232 | 0.09752 | 0.09278 | 0.09002 | **0.02216** | 0.02302 | **0.02216** | 0.02672 |
| CPU time (s) | | 35.61 | | 32.85 | | 39.89 | | 34.38 | |

**Table 2**
Optimization results for the three-bit multimodal problem $f_2$. The table shows the probabilities of obtaining an all-optimal population and the probabilities of obtaining a no-optimal population using the BBO Markov models and simulations. In addition, CPU times for simulations are shown in the last row of the table. The best Markov performance is in **bold font** in each row.

| Mutation rate | Population count vector | Probability | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Partial immigration BBO (original BBO) | | Total immigration BBO | | Partial emigration BBO | | Total emigration BBO | |
| | | Markov | Simulation | Markov | Simulation | Markov | Simulation | Markov | Simulation |
| 0.1 | All optimal | 0.04850 | 0.04727 | **0.05061** | 0.05461 | 0.04876 | 0.04731 | 0.04876 | 0.04313 |
| | No optimal | 0.18195 | 0.18255 | **0.17801** | 0.17162 | 0.20908 | 0.20607 | 0.20908 | 0.20382 |
| 0.01 | All optimal | 0.68721 | 0.69371 | 0.68631 | 0.68133 | **0.70573** | 0.71952 | **0.70573** | 0.71725 |
| | No optimal | **0.04842** | 0.04181 | 0.04995 | 0.04746 | 0.08468 | 0.08595 | 0.08468 | 0.08251 |
| 0.001 | All optimal | **0.93527** | 0.93096 | 0.93503 | 0.93629 | 0.90709 | 0.89833 | 0.90709 | 0.90463 |
| | No optimal | **0.03370** | 0.03232 | 0.03393 | 0.03185 | 0.07016 | 0.08577 | 0.07016 | 0.06934 |
| CPU time (s) | | 38.42 | | 34.51 | | 42.53 | | 37.16 | |

**Table 3**
Optimization results for the three-bit deceptive problem $f_3$. The table shows the probabilities of obtaining an all-optimal population and the probabilities of obtaining a no-optimal population using the BBO Markov models and simulations. In addition, CPU times for simulations are shown in the last row of the table. The best Markov performance is in **bold font** in each row.

| Mutation rate | Population count vector | Probability | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Partial immigration BBO (original BBO) | | Total immigration BBO | | Partial emigration BBO | | Total emigration BBO | |
| | | Markov | Simulation | Markov | Simulation | Markov | Simulation | Markov | Simulation |
| 0.1 | All optimal | 0.03151 | 0.03183 | **0.03280** | 0.03282 | 0.03135 | 0.03402 | 0.03135 | 0.03237 |
| | No optimal | **0.37516** | 0.37655 | 0.38376 | 0.38896 | 0.41166 | 0.42616 | 0.41166 | 0.42014 |
| 0.01 | All optimal | **0.62062** | 0.62098 | 0.61859 | 0.61421 | 0.56959 | 0.55109 | 0.56959 | 0.57042 |
| | No optimal | **0.13629** | 0.13331 | 0.13991 | 0.13832 | 0.22503 | 0.22492 | 0.22503 | 0.21961 |
| 0.001 | All optimal | **0.87712** | 0.87703 | 0.87665 | 0.87534 | 0.78179 | 0.77922 | 0.78179 | 0.78423 |
| | No optimal | **0.09378** | 0.09275 | 0.09428 | 0.09527 | 0.19413 | 0.19174 | 0.19413 | 0.18965 |
| CPU time (s) | | 37.02 | | 33.83 | | 40.94 | | 35.49 | |

comparisons between theoretical (Markov) and simulated results for the four BBO algorithms with mutation rates of 0.1, 0.01, and 0.001 per bit per generation. The tables show the probability of obtaining a population in which all individuals are optimal, and the probability of obtaining a population in which no individuals are optimal. In general, the higher the probability of obtaining an all-optimal population and the lower the probability of obtaining a no-optimal population, the better the optimization performance.

Several things are notable about the results in Table 1. First, the mutation rate affects the performance for all four BBO algorithms. For all three problems, the performance of the four algorithms improves as the mutation rate decreases; that is, the probability of obtaining an all-optimal population increases, and the probability of obtaining a no-optimal population decreases. Tables 1–3 show that a high mutation rate of 0.1 per bit results in too much exploration, so the probability of obtaining an all-optimal population is low, and the probability of obtaining a no-optimal population is relatively high. However, as the mutation rate decreases to the values of 0.01 and 0.001, the probability of obtaining an all-optimal population significantly increases, and the probability of obtaining a no-optimal population significantly decreases. The higher the mutation rate, the lower the probability that the optimum is found and kept for the next generation, which gives worse performance, as shown in Table 1.

Second, for the unimodal one-max problem $f_1$ in Table 1, emigration-based BBO algorithms outperform immigration-based BBO algorithms for all mutation rates; that is, emigration-based BBO algorithms have a higher probability of obtaining an all-optimal population, and a lower probability of obtaining a no-optimal population. For example, for a mutation rate of 0.001 per bit, the best performance is obtained by emigration-based BBO algorithms in their high probability of obtaining an all-optimal population (95.43%), and in their low probability of obtaining a no-optimal population (2.22%). Partial immigration-based BBO and total immigration-based BBO probabilities are 86.05% and 86.01% respectively for obtaining an all-optimal population, and 9.23% and 9.28% respectively for obtaining a no-optimal population.

Third, for the multimodal problem $f_2$ in Table 2, the probability of obtaining an all-optimal population and the probability of obtaining a no-optimal population are very similar for all four BBO algorithms. Specifically, total immigration-based BBO outperforms the other three algorithms when the mutation rate is 0.1 per bit, and partial immigration-based BBO outperforms the other three algorithms when the mutation rate is 0.001 per bit, but the emigration-based BBO algorithms outperform the immigration-based BBO algorithms when the mutation rate is 0.01 per bit.

Fourth, for the deceptive problem $f_3$ in Table 3, the immigration-based BBO algorithms outperform the emigration-based BBO algorithms for all mutation rates, with partial immigration-based BBO slightly better than total immigration-based BBO. For example, for a mutation rate of 0.001 per bit, the best performance is obtained by partial immigration-based BBO in its high probability of obtaining an all-optimal population (87.71%), and in its low probability of obtaining a no-optimal population (9.38%). Total immigration-based BBO and emigration-based BBO algorithms are 87.67% and 78.18% respectively for obtaining an all-optimal population, and 9.43% and 19.41% respectively for obtaining a no-optimal population.

All of these results show that different variations of BBO provide different optimization performance for different types of test problems. For the unimodal problem $f_1$, the emigration-based BBO algorithms are better than the immigration-based BBO algorithms. For the multimodal problem $f_2$, the emigration-based BBO algorithms perform similarly to the immigration-based BBO algorithms. For the deceptive problem $f_3$, the immigration-based BBO algorithms are better than

**Table 4**
Conclusions from the Markov study of the four BBO algorithms on three-bit problems. The "best algorithm" for each problem is determined from an inspection of Table 1.

| Problem | Best algorithm | Other notes |
|---|---|---|
| Unimodal problem | Emigration-based BBO | The two emigration-based BBO algorithms perform identically |
| Multimodal problem | No significant difference between BBO algorithms | All BBO algorithms perform better with low mutation rates |
| Deceptive problem | Immigration-based BBO | The two immigration-based BBO algorithms perform similarly |

the emigration-based BBO algorithms. From Tables 1–3, we further find that the performance of partial immigration-based BBO and total immigration-based BBO are similar for all test problems. These results are summarized in Table 4.

Fifth, the average running times of the four BBO variations for the three test problems are shown in the last row of Table 1. Total immigration-based BBO is the fastest algorithm. Total immigration-based BBO and total emigration-based BBO are faster than partial immigration-based BBO and partial emigration-based BBO for all test problems. The possible reason is that total immigration-based BBO and total emigration-based BBO do not need to decide whether to immigrate or emigrate each feature of each solution, which reduces computational time.

Finally, from Tables 1–3, the Markov model results and the simulation results match well for all of the test problems, which confirms the Markov models of the proposed BBO variations.

### 4.2. Benchmark results

The next experiment compares the optimization performance of the four BBO algorithms on representative benchmark functions [37]. These functions are briefly summarized in Table 5. A more detailed description of these functions can be found in the literature [37]. Functions f01–f07 are high-dimensional and unimodal, functions f08–f13 are high-dimensional and multimodal, and all benchmark functions are to be minimized. The benchmark functions are compared by implementing binary encoding of all BBO algorithms. The granularity or precision of each independent variable in each benchmark function is 0.1, except for the Quartic and generalized Rastrigin functions. Since the domains of each dimension of these two functions are ±1.28 and ±5.12 respectively, they are implemented with a granularity of 0.01.

For all four BBO algorithms, the following parameters have to be tuned: population size, maximum migration rate, and mutation rate. In the literature [17] these parameters have been discussed in detail. Here we use a reasonable set of tuning parameters, but do not make any effort at finding the best parameter settings. We use the same parameters for all four BBO algorithms: a population size of 50, a maximum immigration rate and maximum emigration rate of 1, and a Gaussian mutation rate of 0.001 which is implement as follows: $x_i'(j) = x_i(j) + N_j(0,1)$, where $x_i'$ and $x_i$ respectively denote individuals after and before mutation, $j$ is a decision variable, and $N_j(0, 1)$ indicates a normally distributed random number with a mean of 0 and a variance of 1. In addition, we use linear migration curves as suggested in [28]. The maximum number of fitness function evaluations for each simulation is 50,000. All results are computed from 25 independent simulations.

Table 6 summarizes BBO performance on the 13 benchmark functions. For unimodal functions f01–f07, total emigration-based BBO performs the best, except for function f06, for which all four algorithms attain the global optimum, and function f05, for which partial emigration-based BBO performs the best. For multimodal functions f08–f13, partial immigration-based BBO performs the best on four functions (f08, f09, f10, and f13), and total immigration-based BBO performs the best on the other two functions (f11 and f12). The results indicate that emigration-based BBO algorithms are better than immigration-based BBO algorithms for unimodal functions, and immigration-based BBO algorithms are better than emigration-based BBO algorithms for multimodal functions. These results are consistent with those discussed in the previous subsection. Note that multimodality is correlated with deceptiveness.

In Table 6, we present two-tailed $t$-test results to measure the statistical significance of the differences between partial immigration-based BBO and total immigration-based BBO, and between partial emigration-based BBO and total emigration-based BBO. The $t$-tests indicate whether the difference between groups of data is statistically significant under the assumption that the results are independent and identically normally distributed. For the immigration-based BBO algorithms, there are only five '+' superscripts in Table 6, which denotes that the differences are significant at a level of $\alpha = 0.05(95\%)$. These results mean that there is not a significant difference between partial immigration-based BBO and total immigration-based BBO; the probability that the results from these two algorithms are from the same distribution is high. Similarly, for the emigration-based BBO algorithms, there are only four '+' superscripts in Table 6, which denotes that there is not a significant difference between them.

**Table 5**
Benchmark functions. More details about these functions can be found in [37].

| Function | Name | Dimension | Domain | Minimum | Multimodal? |
|---|---|---|---|---|---|
| f01 | Sphere model | 30 | $-100 \leqslant x_i \leqslant 100$ | 0 | No |
| f02 | Schwefel's problem 2.22 | 30 | $-10 \leqslant x_i \leqslant 10$ | 0 | No |
| f03 | Schwefel's problem 1.2 | 30 | $-100 \leqslant x_i \leqslant 100$ | 0 | No |
| f04 | Schwefel's problem 2.21 | 30 | $-100 \leqslant x_i \leqslant 100$ | 0 | No |
| f05 | Generalized Rosenbrock's function | 30 | $-30 \leqslant x_i \leqslant 30$ | 0 | No |
| f06 | Step function | 30 | $-100 \leqslant x_i \leqslant 100$ | 0 | No |
| f07 | Quartic function | 30 | $-1.28 \leqslant x_i \leqslant 1.28$ | 0 | No |
| f08 | Generalized Schwefel's problem 2.26 | 30 | $-500 \leqslant x_i \leqslant 500$ | −12569.5 | Yes |
| f09 | Generalised Rastrigin's function | 30 | $-5.12 \leqslant x_i \leqslant 5.12$ | 0 | Yes |
| f10 | Ackley's function | 30 | $-32 \leqslant x_i \leqslant 32$ | 0 | Yes |
| f11 | Generalized Griewank's function | 30 | $-600 \leqslant x_i \leqslant 600$ | 0 | Yes |
| f12 | Generalized Penalized function 1 | 30 | $-50 \leqslant x_i \leqslant 50$ | 0 | Yes |
| f13 | Generalized Penalized function 2 | 30 | $-50 \leqslant x_i \leqslant 50$ | 0 | Yes |

**Table 6**
Benchmark results for four BBO algorithms. Here [$a \pm b$] indicates the mean function value and standard deviation. The best result in each row is shown in **bold font**. The '+' superscript denotes that the two-tailed $t$-test result with 24 degrees of freedom is significant at $\alpha = 0.05$ (95%). The $t$-tests are performed between the two immigration-based BBO algorithms, and between the two emigration-based BBO algorithms. In addition, CPU times (minutes) are shown in the last row of the table.

| Function | Partial immigration BBO | Total immigration BBO | Partial emigration BBO | Total emigration BBO |
|---|---|---|---|---|
| f01 | 2.17E−02 ± 4.54E−03 | 7.26E−02 ± 2.78E−03 | 3.57E−05 ± 3.27E−06 | **1.41E−05 ± 8.17E−06** |
| f02 | 1.84E−04 ± 6.23E−05 | 1.04E−03 ± 5.61E−04$^{+}$ | 7.86E−05 ± 4.16E−05 | **5.93E−05 ± 2.41E−05** |
| f03 | 6.33E−02 ± 1.28E−03 | 7.82E−01 ± 3.34E−02$^{+}$ | 9.16E−04 ± 2.34E−05 | **1.02E−04 ± 3.36E−05** |
| f04 | 5.68E−14 ± 7.11E−15 | 9.65E−15 ± 2.79E−16$^{+}$ | 6.45E−19 ± 8.12E−20 | **7.44E−20 ± 5.32E−21**$^{+}$ |
| f05 | 9.24E−01 ± 4.17E−02 | 3.78E−01 ± 1.25E−02 | **1.46E−02 ± 5.19E−03** | 1.85E−02 ± 8.59E−03 |
| f06 | **0.00E+00 ± 0.00E+00** | **0.00E+00 ± 0.00E+00** | **0.00E+00 ± 0.00E+00** | **0.00E+00 ± 0.00E+00** |
| f07 | 1.37E−15 ± 6.29E−16 | 5.37E−14 ± 5.21E−15$^{+}$ | 9.23E−18 ± 3.87E−19 | **4.79E−19 ± 9.16E−20**$^{+}$ |
| f08 | **2.63E−06 ± 7.24E−07** | 2.90E−06 ± 8.33E−07 | 6.14E−04 ± 2.24E−04 | 3.26E−04 ± 4.47E−05 |
| f09 | **1.55E−13 ± 8.11E−14** | 7.13E−12 ± 6.10E−13$^{+}$ | 8.91E−11 ± 1.93E−11 | 9.08E−11 ± 8.03E−11 |
| f10 | **0.00E+00 ± 0.00E+00** | 8.45E−11 ± 9.26E−12 | 2.93E−11 ± 1.18E−11 | 2.96E−11 ± 1.01E−11 |
| f11 | 7.49E−11 ± 4.24E−12 | **0.00E+00 ± 0.00E+00** | 8.24E−11 ± 3.27E−11 | 4.33E−10 ± 3.78E−11$^{+}$ |
| f12 | 2.26E−30 ± 5.17E−31 | **1.98E−30 ± 5.12E−31** | 8.75E−15 ± 1.95E−15 | 2.97E−14 ± 9.21E−15 |
| f13 | **1.28E−10 ± 6.89E−12** | 3.26E−10 ± 8.11E−11 | 3.81E−06 ± 4.42E−07 | 1.04E−05 ± 5.32E−06$^{+}$ |
| CPU time | 171.72 | 142.49 | 198.34 | 158.76 |

These results show that the immigration-based BBO algorithms perform similarly, and also that the emigration-based BBO algorithms perform similarly. This confirms the results of the BBO Markov theory (see Table 4).

In addition, the average running times of four BBO variations are shown in the last row of Table 6. Total immigration-based BBO is the fastest algorithm. Total immigration-based BBO and total emigration-based BBO require less CPU time than partial immigration-based BBO and partial emigration-based BBO for the benchmark functions. These results are consistent with those of the BBO Markov theory.

### 4.3. Real-world optimization results

To further test the performance of BBO, some real-world optimization problems from the 2011 IEEE Congress on Evolutionary Computation [6] are presented. These problems are briefly summarized in Table 7. We compare total emigration-based BBO, which generally provided better performance than the other variations of BBO in the previous sections, with stud GA (which we call SGA) [15], standard PSO 2007 (which we call SPSO 07) [4], [23], and adaptive DE (which we call ADE) [5,11,14,32,38]. We compare with SGA because SGA is an improvement of the classic GA and uses the best individual at each generation for crossover. We compare with PSO because it often offers good performance and is itself a relatively new evolutionary algorithm. We use the current standard PSO 2007, obtained from Particle Swarm Central [22]. We compare with DE because it is one of the most powerful evolutionary algorithms and has demonstrated excellent performance on many prob-

**Table 7**
Problem set descriptions. More details about these problems can be found in [6].

| Problem | Dimension | Comments |
|---|---|---|
| P01 | 6 | Parameter estimation for frequency-modulated (FM) sound waves. |
| P02 | 30 | Lennard–Jones potential problem. |
| P03 | 1 | Bifunctional catalyst blend optimal control problem. |
| P04 | 1 | Optimal control of a nonlinear stirred tank reactor. |
| P05 | 30 | Tersoff potential function minimization problem (instance 1). |
| P06 | 30 | Tersoff potential function minimization problem (instance 2). |
| P07 | 20 | Spread spectrum radar polyphase code design. |
| P08 | 7 | Transmission network expansion planning problem. |
| P09 | 126 | Large scale transmission pricing problem. |
| P10 | 12 | Circular antenna array design problem. |
| P11.1 | 120 | Dynamic economic dispatch problem (instance 1). |
| P11.2 | 216 | Dynamic economic dispatch problem (instance 2). |
| P11.3 | 6 | Static economic load dispatch problem (instance 1). |
| P11.4 | 13 | Static economic load dispatch problem (instance 2). |
| P11.5 | 15 | Static economic load dispatch problem (instance 3). |
| P11.6 | 40 | Static economic load dispatch problem (instance 4). |
| P11.7 | 140 | Static economic load dispatch problem (instance 5). |
| P11.8 | 96 | Hydrothermal scheduling problem (instance 1). |
| P11.9 | 96 | Hydrothermal scheduling problem (instance 2). |
| P11.10 | 96 | Hydrothermal scheduling problem (instance 3). |
| P12 | 26 | Spacecraft trajectory optimization problem (Messenger). |
| P13 | 22 | Spacecraft trajectory optimization problem (Cassini2). |

**Table 8**
Comparison of real-world optimization results for StudGA, SPSO 07, ADE, and BBO. Here [$a \pm b$] indicates the mean value and corresponding standard deviation. The best result in each row is shown in **bold font**. In addition, CPU times (minutes) are shown in the last row of the table.

| Problem | SGA | SPSO 07 | ADE | BBO |
|---|---|---|---|---|
| P01 | 7.44E−18 ± 5.11E−19 | 2.60E−02 ± 4.83E−03 | **0.00E+00 ± 0.00E+00** | 7.35E−17 ± 2.53E−19 |
| P02 | −2.62E+01 ± 1.75E+00 | −2.81E+01 ± 1.39E+00 | −2.60E+01 ± 1.36E+00 | **−2.83E+01 ± 1.27E+00** |
| P03 | **1.15E−05 ± 0.00E+00** | **1.15E−05 ± 0.00E+00** | **1.15E−05 ± 0.00E+00** | **1.15E−05 ± 0.00E+00** |
| P04 | 2.03E+01 ± 6.72E−01 | **1.37E+01 ± 4.85E−01** | 2.54E+01 ± 5.90E−01 | 1.43E+01 ± 3.98E−01 |
| P05 | −3.68E+01 ± 2.11E+00 | −3.27E+01 ± 2.01E+00 | −3.66E+01 ± 2.32E+00 | **−3.69E+01 ± 1.87E+00** |
| P06 | −2.91E+01 ± 8.64E−01 | −2.68E+01 ± 6.77E−01 | −2.91E+01 ± 7.92E−01 | **−2.92E+01 ± 1.86E−01** |
| P07 | **5.00E−01 ± 1.22E−02** | 5.08E−01 ± 7.65E−03 | 5.08E−01 ± 3.12E−03 | 9.97E−01 ± 7.45E−03 |
| P08 | **2.20E+02 ± 0.00E+00** | **2.20E+02 ± 0.00E+00** | **2.20E+02 ± 0.00E+00** | **2.20E+02 ± 0.00E+00** |
| P09 | 3.05E+02 ± 4.64E+01 | 1.60E+03 ± 9.02E+01 | **4.62E+01 ± 2.73E+00** | 1.04E+03 ± 2.55E+02 |
| P10 | −2.01E+01 ± 7.65E+00 | −2.09E+01 ± 4.18E+00 | −2.17E+01 ± 1.47E+00 | **−2.18E+01 ± 1.32E+00** |
| P11.1 | **4.79E+04 ± 5.87E+02** | 1.50E+05 ± 1.49E+04 | 5.73E+04 ± 7.11E+02 | 5.25E+04 ± 3.74E+03 |
| P11.2 | 1.81E+07 ± 7.26E+05 | 6.10E+06 ± 4.27E+05 | 1.06E+06 ± 3.44E+05 | **1.05E+06 ± 1.96E+05** |
| P11.3 | **1.54E+04 ± 2.86E+02** | 1.58E+04 ± 5.75E+02 | 1.54E+04 ± 3.11E+02 | **1.54E+04 ± 2.89E+02** |
| P11.4 | 1.80E+04 ± 4.55E+03 | 1.82E+04 ± 2.40E+03 | 1.79E+04 ± 3.64E+03 | 1.89E+04 ± 3.11E+03 |
| P11.5 | **3.26E+04 ± 7.11E+02** | 3.27E+04 ± 4.37E+02 | 3.27E+04 ± 8.19E+02 | 3.29E+04 ± 7.15E+02 |
| P11.6 | 1.21E+05 ± 3.92E+03 | 1.37E+05 ± 3.55E+03 | **1.20E+05 ± 1.75E+03** | 1.32E+05 ± 4.72E+03 |
| P11.7 | 1.95E+06 ± 3.19E+04 | 2.13E+06 ± 1.03E+05 | **1.70E+06 ± 3.75E+04** | 1.91E+06 ± 9.28E+04 |
| P11.8 | 9.54E+05 ± 1.99E+04 | 1.16E+06 ± 4.65E+05 | 9.31E+05 ± 2.01E+04 | **9.23E+05 ± 1.02E+04** |
| P11.9 | 9.39E+05 ± 4.33E+04 | 1.60E+06 ± 2.01E+04 | 1.23E+06 ± 9.75E+04 | **9.30E+05 ± 1.73E+04** |
| P11.10 | 9.51E+05 ± 6.83E+03 | 1.21E+06 ± 2.95E+04 | 9.29E+05 ± 5.49E+03 | **9.24E+05 ± 1.70E+03** |
| P12 | 7.89E+00 ± 3.12E−01 | 1.38E+01 ± 6.90E−01 | **7.07E+00 ± 3.66E−01** | 1.64E+01 ± 4.81E+00 |
| P13 | 8.65E+00 ± 7.61E−01 | 8.78E+00 ± 1.54E−01 | **8.61E+00 ± 8.17E−02** | 1.43E+01 ± 1.78E+00 |
| CPU time | 73.28 | 101.20 | 115.16 | 83.75 |

lems. We use the adaptive DE proposed by [2], where control parameter settings are gradually adapted according to the learning progress, and which uses a center based differential exponential crossover and incorporates local search to improve its efficiency.

We recognize that there are many other evolutionary algorithms, including the estimation of distribution algorithm (EDA), evolutionary strategy (ES), ant colony optimization (ACO), and their variants, which may provide better optimization performance than the algorithms we use in this paper. However, the purpose of this section is to demonstrate the optimization ability of BBO for real-world problems and to compare BBO with a few well-established algorithms. Comparisons with additional evolutionary algorithms are deferred for future research.

The parameters used in BBO in this section are the same as those in the previous subsection. For the SGA we use real coding, roulette wheel selection, single point crossover with a crossover probability of 1, and a mutation probability of 0.001. For SPSO 07 we use an inertia weight of 0.8, a cognitive constant of 0.5, a social constant for swarm interaction of 1.0, and a social constant for neighborhood interaction of 1.0. For ADE we use an adaptive scaling factor (F) with the range [0.1–0.5], and an adaptive crossover rate (CR) with the range [0.8– 0.98].

Each algorithm has a population size of 50, and a maximum of 100,000 fitness function evaluations. The granularity of each real-world optimization problem is 0.1, except for P01, P03, P10, P12 and P13, which are implemented with a granularity of 0.01. The results of solving these real-world optimization problems are given in Table 8. All results are computed from 25 independent simulations.

According to Table 8, BBO performs best on 8 problems (P02, P05, P06, P10, P11.2, P11.8, P11.9, and P11.10), ADE performs best on seven problems (P01, P09, P11.4, P11.6, P11.7, P12, and P13), SGA performs best on three problems (P07, P11.1, and P11.5), and SPSO 07 performs best on problem P04. In addition, we see that for problems P03 and P08, all four algorithms attain the same optimum, and for problem P11.3, SGA, ADE, and BBO all attain the same optimum. These results indicate that BBO performs similarly to ADE, and is significantly better than SGA and SPSO 07.

If we use more advanced versions of GA, PSO, and DE, it might be possible to obtain better results than those here. However, the same could be said for recently proposed improvements of BBO [8,9]. The purpose of these comparisons is not to tune our algorithms to obtain the best possible performance for specific problems, but rather to show that BBO is a competitive algorithm for real-world optimization problems.

The average running times of the four optimization methods are shown in the last row of Table 8. SGA is the fastest algorithm, and BBO is the second fastest.

## 5. Conclusions

This paper presented four BBO algorithms: partial immigration-based BBO, which is the original BBO algorithm, and three new algorithms. The new algorithms are called total immigration-based BBO, partial emigration-based BBO, and total emigration-based BBO. The optimization performance of these BBO variations has been explored with newly-derived Markov chain models, and new theoretical results for the four BBO algorithms have been obtained, which were confirmed with

simulation results. The results confirm that different migration methods in BBO result in significant differences in performance. Total emigration-based BBO and partial emigration-based BBO obtain better performance for three-bit unimodal problems, partial immigration-based BBO is more appropriate for three-bit deceptive problems, and all the variations have similar results for three-bit multimodal problems.

Although the theoretical results are limited to small problem dimensions due to the factorial increase of the Markov transition matrix size with problem dimension, the results provide a strong indication that migration methods can significantly affect BBO performance. To further test the performance of the four BBO algorithms, some representative benchmark functions were used, and the results confirmed the conclusions obtained by the Markov chain models. In addition, we tested BBO on real-world optimization problems. Comparisons of BBO with the stud GA, PSO, and DE, showed that BBO is a competitive algorithm for real-world optimization problems, especially total emigration-based BBO.

For future work there are several important directions. This paper bases immigration rate and emigration rate on linear migration curves. It is also of interest to combine these four BBO algorithms with nonlinear migration curves [17]. This will allow for theoretical comparisons between different types of BBO algorithms with different migration curves. The second important direction for future work is to study the optimum-hitting time of BBO with different migration methods. This paper obtains the Markov chain models of BBO algorithms, which provide the foundation for additional theoretical tools that can be used to study hitting time. The third important direction for future work is to look at the optimization ability of the proposed BBO algorithms for translated and rotated complex functions, and for high-dimensional optimization functions. The fourth direction for future work is to develop hybrid BBO algorithms, which combine BBO with other EAs, and to obtain their Markov models. The fifth direction for future work is to further develop the optimization ability of BBO by using principles from natural biogeography theory to modify the BBO algorithm.

Finally, we make the important observation that all of the Markov modeling in this paper is for a BBO algorithm that operates on discrete search spaces with a finite number of states. We obtained the simulation results in this paper by discretizing continuous search spaces, and so our Markov models do apply to the simulations in this paper. However, many EAs (including BBO) can operate on continuous domains. In one sense we can say that all continuous domains reduce to discrete domains when the EA is simulated on a digital computer. However, the number of discrete states in a 32-bit or 64-bit computer is so large that it approaches infinity for all practical purposes. It would be more accurate to obtain continuous-state-space Markov models for EAs that operate on mathematically continuous domains. The discrete Markov theory that we discussed in this paper does not directly apply to continuous state spaces. For continuous state spaces, the Markov transition matrix is replaced with a transition kernel $K$, where $K(x, A)$ is the probability of transitioning from the state $x$ to the region $A$ in state space. We can then obtain results that are analogous to those in Section 3 of this paper, but the mathematics of continuous-state-space Markov processes are more involved than those of discrete-state-space Markov processes [26]. We therefore relegate the development of the Markov model for the continuous-domain BBO algorithm to future work.

## Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0826124, the National Natural Science Foundation of China under Grant No. 61074032, the Zhejiang Provincial Natural Science Foundation of China under Grant No. Y1090866, and the Project of Science and Technology Commission of Shanghai Municipality under Grant No. 10JC1405000.

## References

[1] C. Ahn, Advances in Evolutionary Algorithms: Theory, Design and Practice, Springer Publishing, New York, 2006.
[2] M. Asafuddoula, T. Ray, R. Sarker, An adaptive differential evolution algorithm and its performance on real world optimization problems, in: IEEE Congress on Evolutionary Computation, New Orleans, LA, June 2011, pp. 1057–1062.
[3] A. Bhattacharya, P. Chattopadhyay, Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch, IEEE Transactions on Power Systems 25 (4) (2010) 1955–1964.
[4] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in: IEEE Swarm Intelligence Symposium, Honolulu, Hawaii, April 2007, pp. 120–1 27.
[5] S. Das, P.N. Suganthan, Differential evolution – a survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 4–31.
[6] S. Das, P.N. Suganthan, Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems, Technical Report, Jadavpur University, Nanyang Technological University, December 2010.
[7] T. Davis, J. Principe, A Markov chain framework for the simple genetic algorithm, Evolutionary Computation 1 (3) (1993) 269–288.
[8] D. Du, D. Simon, M. Ergezer, Biogeography-based optimization combined with evolutionary strategy and immigration refusal, in: IEEE Conference on Systems, Man, and Cybernetics, San Antonio, Texas, October 2009, pp. 1023–1028.
[9] M. Ergezer, D. Simon, D. Du, Oppositional biogeography-based optimization, in: IEEE Conference on Systems, Man, and Cybernetics, San Antonio, Texas, October 2009, pp. 1035–1040.
[10] I.D. Falco, A.D. Cioppa, D. Maisto, U. Scafuri, E. Tarantino, Biological invasion-inspired migration in distributed evolutionary algorithms, Information Sciences 207 (10) (2012) 50–65.
[11] A. Ghosh, S. Das, A. Chowdhury, R. Giri, An improved differential evolution algorithm with fitness-based adaptation of the control parameters, Information Sciences 181 (18) (2011) 3749–3765.
[12] S. Ghosh, S. Das, S. Roy, S.K. Islam, P.N. Suganthan, A differential covariance matrix adaptation evolutionary algorithm for real parameter optimization, Information Sciences 182 (1) (2012) 199–219.
[13] C. Grinstead, J. Snell, Introduction to Probability, American Mathematical Society, Providence, Rhode Island, 1997.
[14] W. Gong, Á. Fialho, Z. Cai, H. Li, Adaptive strategy selection in differential evolution for numerical optimization: an empirical study, Information Sciences 181 (24) (2011) 5364–5386.

[15] W. Khatib, P. Fleming, The stud GA: a mini revolution?, in: A. Eiben, T. Back, M. Schoenauer, H. Schwefel (Eds.), Parallel Problem Solving from Nature, Springer, New York, 1998, pp. 683–691.
[16] M. Lundy, A. Mees, Convergence of an annealing algorithm, Mathematical Programming. 34 (1) (1986) 111–124.
[17] H. Ma, An analysis of the equilibrium of migration models for biogeography-based optimization, Information Sciences 180 (18) (2010) 3444–3464.
[18] H. Ma, S. Ni, M. Sun, Equilibrium species counts and migration model tradeoffs for biogeography-based optimization, in: IEEE Conference on Decision and Control, Shanghai, China, December 2009, pp. 3306–3310.
[19] H. Ma, D. Simon, Biogeography-based optimization with blended migration for constrained optimization problems, in: Genetic and Evolutionary Computation Conference, Portland, OR, July 2010, pp. 417–418.
[20] A. Nix, M. Vose, Modeling genetic algorithms with Markov chains, Annals of Mathematics and Artificial Intelligence 5 (1) (1992) 79–88.
[21] V. Panchal, P. Singh, N. Kaur, H. Kundra, Biogeography based satellite image classification, International Journal of Computer Science and Information Security 6 (2) (2009) 269–274.
[22] Particle Swarm Central. <http://www.particleswarm.info/>.
[23] B.Y. Qu, J.J. Liang, P.N. Suganthan, Niching particle swarm optimization with local search for multi-modal optimization, Information Sciences 197 (15) (2012) 131–143.
[24] R. Rarick, D. Simon, F.E. Villaseca, B. Vyakaranam, Biogeography-based optimization and the solution of the power flow problem, in: IEEE Conference on Systems, Man, and Cybernetics, San Antonio, Texas, October 2009, pp. 1029–1034.
[25] C. Reeves, J. Rowe, Genetic Algorithms: Principles and Perspectives, Kluwer Academic Publisher, Norwell, Massachusetts, 2003.
[26] G. Rudolph, Convergence Properties of Evolutionary Algorithms, Verlag Dr. Kovac, 1997.
[27] H.P. Schwefel, Evolution and Optimum Seeking, Wiley Press, 1995.
[28] D. Simon, Biogeography-based optimization, IEEE Transactions on Evolutionary Computation 12 (6) (2008) 702–713.
[29] D. Simon, M. Ergezer, D. Du, Population distributions in biogeography-based optimization algorithms with elitism, in: IEEE Conference on Systems, Man, and Cybernetics, San Antonio, Texas, October 2009, pp. 1017–1022.
[30] D. Simon, M. Ergezer, D. Du, R. Rarick, Markov models for biogeography-based optimization, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 41 (1) (2011) 299–306.
[31] D. Simon, R. Rarick, M. Ergezer, D. Du, Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms, Information Sciences 181 (7) (2011) 1224–1248.
[32] R. Storn, K.V. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359.
[33] J. Sun, X. Wu, W. Fang, Y. Ding, H. Long, W. Xu, Multiple sequence alignment using the hidden Markov model trained by an improved quantum-behaved particle swarm optimization, Information Sciences 182 (1) (2012) 93–114.
[34] J. Suzuki, A Markov chain analysis on simple genetic algorithms, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 25 (4) (1995) 655–659.
[35] J. Suzuki, A further result on the Markov chain model of genetic algorithms and its application to a simulated annealing-like strategy, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 28 (1) (1998) 95–102.
[36] M.D. Vose, The Simple Genetic Algorithm: Foundations and Theory, The MIT press, 1999.
[37] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 82–102.
[38] D. Zaharie, Control of population diversity and adaptation in differential evolutionary algorithms, in: 9th International Conference on Soft Computing, Brno, Czech Republic, 2003, pp. 41–46.