Information Sciences 181 (2011) 1224–1248

Contents lists available at ScienceDirect

Information Sciences

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

Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms

Dan Simon *, Rick Rarick, Mehmet Ergezer, Dawei Du

Cleveland State University, Department of Electrical and Computer Engineering, Stilwell Hall Room 332, 2121 Euclid Avenue, Cleveland, OH 44115, United States

ARTICLE INFO

Article history:
Received 26 March 2010
Received in revised form 27 October 2010
Accepted 16 December 2010
Available online 23 December 2010

Keywords:
Evolutionary algorithm
Markov model
Biogeography-based optimization
Genetic algorithm
Global uniform recombination
Benchmark

ABSTRACT

We show that biogeography-based optimization (BBO) is a generalization of a genetic algorithm with global uniform recombination (GA/GUR). Based on the common features of BBO and GA/GUR, we use a previously-derived BBO Markov model to obtain a GA/GUR Markov model. One BBO characteristic which makes it distinctive from GA/GUR is its migration mechanism, which affects selection pressure (i.e., the probability of retaining certain features in the population from one generation to the next). We compare the BBO and GA/GUR algorithms using results from analytical Markov models and continuous optimization benchmark problems. We show that the unique selection pressure provided by BBO generally results in better optimization results for a set of standard benchmark problems. We also present comparisons between BBO and GA/GUR for combinatorial optimization problems, include the traveling salesman, the graph coloring, and the bin packing problems.

1. Introduction

Mathematical models of biogeography describe the migration of species between islands, along with their speciation and extinction [27,57]. Biogeography-based optimization (BBO) was first presented in [45] and is an example of how a natural process can be generalized to solve optimization problems. Since its introduction, it has been applied to a variety of problems, including sensor selection [45], power system optimization [38,42], groundwater detection [24], mechanical gear train design [43], and satellite image classification [36].

Like other evolutionary algorithms (EAs), BBO is based on the idea of probabilistically sharing information between candidate solutions (individuals) based on their fitness values. Suppose we have a population of candidate solutions to an optimization problem. Each individual is comprised of a set of features. When a copy of feature s from individual x replaces one of the features in individual y, we say that s has *emigrated* from x and *immigrated* to y. The probability that a given individual shares its features increases with fitness, and the probability that a given individual receives features from other individuals decreases with fitness.

Although more complicated and life-like migration curves can give better optimization results [30], we use linear migration curves like those shown in Fig. 1 for the sake of simplicity. Fig. 1 illustrates two individuals in BBO. S_1 represents a poor solution and S_2 represents a good solution. The immigration probability for S_1 will therefore be higher than the immigration probability for S_2. The emigration probability for S_1 will be lower than the emigration probability for S_2.

There are several different ways to implement the details of BBO, but in this paper we use the original BBO formulation [45], which is called partial immigration-based BBO in [47]. In this approach, for each feature in each solution we probabilistically decide whether or not to immigrate. If immigration is selected for a given feature, then the emigrating solution is

* Corresponding author.
E-mail address: d.j.simon@csuohio.edu (D. Simon).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.12.006

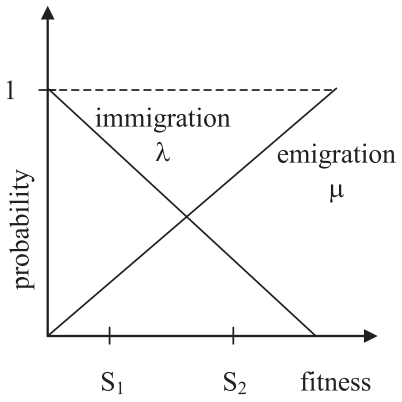

Fig. 1. Illustration of two candidate solutions to some problem using symmetric immigration and emigration curves. S_1 is a relatively poor solution and S_2 is a relatively good solution. S_1 is likely to receive features from other individuals, but unlikely to share features with other individuals. S_2 is unlikely to receive features from other individuals, but likely to share features with other individuals.

probabilistically selected based on fitness (e.g., using roulette wheel selection). This gives the algorithm shown in Fig. 2 as a conceptual description of one generation. Migration and mutation of each individual in the current generation occurs before any of the individuals are replaced in the population, which requires the use of the temporary population z in the BBO algorithm shown in Fig. 2. Borrowing from GA terminology [55], we therefore say that Fig. 2 depicts a *generational* BBO algorithm.

This paper is based on a previously-developed Markov model for BBO [48]. Markov models have been developed for other EAs also, including simple genetic algorithms [50,51] and simulated annealing [28].

A Markov chain is a random process which has a set of T possible states [19, Chapter 11]. The probability that the system transitions from state i to state 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. If $w(t)$ is a column vector containing the probabilities that the system is in each state at time t, then $w(t+1) = Pw(t)$ describes how the probabilities change from one time step to the next. A Markov chain and its transition matrix are *regular* if some power of the transition matrix has only positive elements. The fundamental limit theorem for regular Markov chains says that if P is regular, then

$$\lim_{n\to\infty} P^n = P(\infty) \tag{1}$$

where each row p_{ss} of $P(\infty)$ is the same. The ith element of p_{ss} is the probability that the Markov chain is in state i after an infinite number of transitions, and p_{ss} is independent of the initial state.

A Markov state in this paper represents a BBO population distribution. Each state describes how many of each individual of the search space there are in the population. The probability P_{ij} is the probability that the population transitions from the ith distribution to the jth distribution from one generation to the next. If the mutation rate is nonzero, this probability is

---

For each solution y_k, $k \in \{1, \ldots, N\}$, define emigration probability $\mu_k \propto$ fitness of y_k, $\mu_k \in [0, 1]$
For each solution y_k define immigration probability $\lambda_k = 1 - \mu_k$
$z \leftarrow y$
For each solution z_k
    For each solution feature s
        Use λ_k to probabilistically decide whether to immigrate to z_k
        If immigrating then
            Use $\{\mu_i\}$ to probabilistically select the emigrating solution y_j
            $z_k(s) \leftarrow y_j(s)$
        end if
    next solution feature
    Probabilistically mutate z_k
next solution
$y \leftarrow z$

Fig. 2. One generation of the BBO algorithm. N is the population size, y is the entire population of solutions, y_k is the kth solution, and $y_k(s)$ is the sth feature of y_k. Similarly, z is the temporary population of solutions, z_k is the kth temporary solution, and $z_k(s)$ is the sth feature of z_k.

greater than zero for all i and j, which means that the transition matrix is regular. This means that there is a unique, nonzero, limiting probability for each possible population distribution as the number of generations approaches infinity.

In Section 2, we compare and contrast BBO and GA with global uniform recombination (GA/GUR) from an algorithmic perspective. In Section 3 we review our previously-derived BBO Markov model [48] and use it to obtain the limiting distribution of the populations. We compare the BBO Markov model with the Markov models of GA/GUR and GA with single-point crossover (GA/SP). In Section 4 we use the BBO and GA/GUR Markov models to make some analytical comparisons, which we then support with simulations of parametric benchmark optimization problems. In Section 5 we empirically compare BBO and GA/GUR on combinatorial benchmarks, including traveling salesman, graph coloring, and bin packing problems. We provide concluding remarks and directions for future work in Section 6.

2. Biogeography-based optimization and genetic algorithms

The BBO migration strategy is conceptually similar to a combination of two ideas from the GA literature: *global recombination* and *uniform crossover*. The first idea, global recombination, originated with evolutionary strategies (ES) and means that many parents can contribute features to a single offspring [2,3]. This idea has also been applied with the names multi-parent recombination [12,13] and scanning crossover [14], and was suggested as early as 1966 [6]. Global recombination strays from the biological foundation of GAs because individuals in nature cannot have more than two parents. There are several choices to be made when implementing global recombination in GAs. For example, how many individuals should be in the pool of potential parents? How should individuals be chosen for the pool? Once the pool has been determined, how should parents be selected from the pool?

The second idea, uniform crossover, was first proposed in [1]. Uniform crossover means that each solution feature in an offspring is generated independently from every other solution feature. If we combine global recombination and uniform crossover, we obtain *global uniform recombination*. If in addition we use the entire population as potential contributors to the next generation, and we also use fitness-based selection for each solution feature in each offspring, we obtain the algorithm shown in Fig. 3. Comparing Figs. 2 and 3, we see that BBO is a generalization of a specific type of GA/GUR. If rather than setting $\lambda_k = 1 - \mu_k$ in the BBO algorithm of Fig. 2, we instead set $\lambda_k = 1$ for all k, then we obtain the GA/GUR algorithm of Fig. 3.

It is not too surprising that BBO is similar to GA/GUR, because many EAs can be expressed in terms of each other. For example, consider differential evolution (DE) [49,37]. DE involves the selection of three random individuals from the population, denoted r_1, r_2, and r_3, and the generation of a random integer n between 1 and the population size. If, however, r_1 is selected on the basis of fitness, r_2 is replaced with r_1, r_3 is replaced with r_2 and is selected on the basis of fitness, and $n = 1$, then DE is equivalent to a continuous GA with intermediate global recombination [21] in which the first parent is chosen deterministically and the second parent is chosen based on fitness.

As another example of the similarity between EAs, consider particle swarm optimization (PSO) [8]. If a particle's velocity at each generation is independent of its previous velocity, the random proportionality constant ϕ_1 is set equal to 0, and the neighborhood's best position p_g is probabilistically selected based on fitness, then PSO, like DE, is equivalent to a continuous GA with intermediate fitness-based global recombination.

The final example that we note here is evolution strategy (ES) [32]. If a (μ, λ) ES is implemented with $\lambda = \mu$, with fitness-based parent selection, with uniform recombination, and with a constant (nonadaptive) mutation parameter, then it is equivalent to a GA with fitness-based global uniform recombination. (Note that the μ and λ that are used in ES notation are not related to the μ and λ that are used in biogeography and BBO notation.)

Since GA/GUR can be viewed as either BBO, DE, PSO, or ES under special conditions, it follows that all of these EAs function identically under these special conditions. However, this identical functionality occurs only under special conditions, and

---

For each solution y_k, $k \in \{1, \ldots, N\}$, define parent probability $\mu_k \propto$ fitness of y_k, $\mu_k \in [0, 1]$
$z \leftarrow y$
For each solution z_k
    For each solution feature s
        Use $\{\mu_i\}$ to probabilistically select the parent solution y_j
        $z_k(s) \leftarrow y_j(s)$
    next solution feature
    Probabilistically mutate z_k
next solution
$y \leftarrow z$

Fig. 3. One generation of a GA with global uniform recombination (GA/GUR). N is the population size, y is the entire population of solutions, y_k is the kth solution, and $y_k(s)$ is the sth feature of y_k. Similarly, z is the temporary population of solutions, z_k is the kth temporary solution, and $z_k(s)$ is the sth feature of z_k. Compare with Fig. 2.

each EA still has its own particular features and parameters that give it a unique flexibility that other EAs do not have. It is therefore useful to retain the distinction between these EAs because of their differences.

Another reason that it is useful to retain the distinction between EAs is their unique biological motivations. For example, retaining the biological foundation of GAs stimulates the incorporation of features from biology in GAs, which makes the study of GAs richer and more flexible. Some of these features include gender, niching, crowding, aging, diploidy, co-evolution, and ontogony [3].

Similarly, it is advantageous to retain BBO as a distinctive EA rather than viewing it as a generalized GA. Unifying various EAs is instructive, but retaining BBO as a separate algorithm stimulates the incorporation of behaviors from natural biogeography into the BBO algorithm, and this opens up many areas of further research. Some of these behaviors include the effect of geographical proximity on migration rates, nonlinear migration curves [30], species populations (including mortality and reproduction), predator/prey relationships, species mobilities, directional momentum during migration, habitat area and isolation, and many others [27,57].

3. Markov models for BBO and GA/GUR

In [48] a Markov model for BBO is derived. In this section we review that model and show how it also applies to GA/GUR. This allows us to make analytical comparisons between BBO and GA/GUR at the end of this section and support those results with simulations.

3.1. Markov model for biogeography-based optimization

The BBO Markov model derived in [48] makes three assumptions. First, all of the new BBO solutions are created before any solutions are replaced in the population; that is, we use a generational BBO algorithm rather than a steady-state BBO algorithm. This is clear from the use of the temporary population z in Fig. 2. Second, a solution can emigrate a feature to itself. Third, the migration rates λ and μ are independent of the population distribution; that is, absolute fitness values are used to obtain λ and μ, as opposed to a rank-based fitness [56].

Suppose that we have a problem whose solutions 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 $n = 2^q$. We use N to denote the population size, and we use $v = [v_1, \ldots, v_n]^T$ to denote the population vector, where the component $v_i \in \{0, \ldots, N\}$ is the number of x_i individuals in the population. We see that

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

We use y_k to denote the kth individual in the population, where the elements of y_k are ordered to group identical individuals. The population of the search algorithm can thus be depicted as

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

We use μ_i to denote the emigration probability of x_i, which is proportional to the fitness of x_i. We use λ_i to denote the immigration probability of x_i, which decreases with the fitness of x_i. Note that we used μ_i and λ_i, for $i \in \{1, \ldots, N\}$, in Fig. 2 to refer to the migration probabilities of y_i; now we are using μ_i and λ_i, for $i \in \{1, \ldots, n\}$, to refer to the migration probabilities of x_i.

We use the notation $x_i(s)$ to denote the sth bit of solution x_i. For $i \in \{1, \ldots, n\}$ and $s \in \{1, \ldots, q\}$ we use the notation $\mathcal{J}_i(s)$ to denote the set of search space indices j such that $x_j(s) = x_i(s)$:

$$\mathcal{J}_i(s) = \{j \in \{1, \ldots, n\} : x_j(s) = x_i(s)\} \tag{4}$$

Note that the cardinality of $\mathcal{J}_i(s)$ is $n/2$ for all i and s. See [48] for an example that shows how $\mathcal{J}_i(s)$ is constructed. From (3) we see that

$$y_k = \begin{cases} x_1 & \text{for } k = 1, \ldots, v_1 \\ x_2 & \text{for } k = v_1 + 1, \ldots, v_1 + v_2 \\ x_3 & \text{for } k = v_1 + v_2 + 1, \ldots, v_1 + v_2 + v_3 \\ \vdots & \vdots \\ x_n & \text{for } k = \sum_{i=1}^{n-1} v_i + 1, \ldots, N \end{cases} \tag{5}$$

This can be written more compactly as

$$y_k = x_{m(k)} \quad \text{for } k = 1, \ldots, N \tag{6}$$

where $m(k) \in \{1, \ldots, n\}$ is defined as

$$m(k) = \min r \quad \text{such that} \quad \sum_{i=1}^{r} v_i \geqslant k \tag{7}$$

We use an additional subscript to denote the generation number of the algorithm. For example, $y_k(s)_t$ is the value of the sth bit of the kth individual at generation t.

Let $\text{im}(t)$ be the event that immigration occurs at the tth generation, and let $\overline{\text{im}}(t)$ be its negation. With these definitions, and viewing $y_k(s)_{t+1}$ as a random variable, it is shown in [48] that for fixed i, k, s, and t,

$$\begin{aligned}\Pr(y_k(s)_{t+1} = x_i(s)) &= \Pr(\overline{\text{im}}(t))\Pr(y_k(s)_{t+1} = x_i(s)|\overline{\text{im}}(t)) + \Pr(\text{im}(t))\Pr(y_k(s)_{t+1} = x_i(s)|\text{im}(t)) \\ &= \begin{cases} (1-\lambda_{m(k)}) + \lambda_{m(k)} \dfrac{\sum_{j\in\mathcal{I}_i(s)} v_j\mu_j}{\sum_{j=1}^n v_j\mu_j}, & \text{if } x_{m(k)}(s) = x_i(s) \\ \lambda_{m(k)} \dfrac{\sum_{j\in\mathcal{I}_i(s)} v_j\mu_j}{\sum_{j=1}^n v_j\mu_j}, & \text{if } x_{m(k)}(s) \neq x_i(s) \end{cases} = (1-\lambda_{m(k)})\mathbf{1}_0(x_{m(k)}(s) - x_i(s)) + \lambda_{m(k)} \frac{\sum_{j\in\mathcal{I}_i(s)} v_j\mu_j}{\sum_{j=1}^n v_j\mu_j} \\ &= (1-\lambda_{m(k)})\mathbf{1}_0(x_{m(k)}(s) - x_i(s)) + \lambda_{m(k)} f_i(s) \end{aligned} \tag{8}$$

where $\mathbf{1}_0$ is the indicator function on the set $\{0\}$, and $f_i(s)$ is defined by the above equation. We call $f_i(s)$ the fitness-weighted abundance of $x_i(s)$ bits in the population. Note that $f_i(s)$ is equal to $\Pr(y_k(s)_{t+1} = x_i(s)|\text{im}(t))$. This is the ratio of the sum of the fitness values of those individuals whose sth bit is equal to $x_i(s)$, to the sum of the fitness values of the entire population. Since we use fitness-proportional selection to choose the emigrating individual, as shown in the use of μ_k in Fig. 2, the ratio of sums gives the desired probability.

For each k and each individual $y_{k,t+1}$, we have q independent random variables, $y_k(1)_{t+1}, \ldots, y_k(q)_{t+1}$. Therefore, for a fixed i,

$$\Pr(y_k(1)_{t+1} = x_i(1), \ldots, y_k(q)_{t+1} = x_i(q)) = \prod_{s=1}^{q} \Pr(y_k(s)_{t+1} = x_i(s)) \tag{9}$$

Given this fact and the fact that the population at the t-th generation is described by the vector v, the probability that $y_{k,t+1} = x_i$ is denoted as $P_{ki}(v)$ and can be written as

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

$P_{ki}(v)$ can be computed for each $k \in \{1,\ldots,N\}$ and each $i \in \{1,\ldots,n\}$ to form the $N \times n$ matrix $P(v)$. The kth row of $P(v)$ corresponds to the kth iteration of the outer loop in Fig. 2. The ith column of $P(v)$ corresponds to the probability of obtaining x_i during each outer loop iteration.

The BBO algorithm entails N trials (i.e., N iterations of the outer loop in Fig. 2), where the probability of the ith outcome on the kth trial (that is, $y_k \leftarrow x_i$) is given as $P_{ki}(v)$. We use u_i to denote the total number of times that outcome i occurs after all N trials have been completed, and define $u = [u_1 \quad \cdots \quad u_n]^T$. Then the probability that we obtain population vector u at the $(t+1)$ st generation, given that we have population vector v at the tth generation, can be calculated with the generalized multinomial theorem [48,4] as

$$\Pr(u|v) = \sum_{J\in Y(u)} \prod_{k=1}^{N} \prod_{i=1}^{n} P_{ki}^{J_{ki}}(v) \tag{11}$$

where

$$Y(u) = \left\{ J \in \mathbf{R}^{N\times n} : J_{ki} \in \{0,1\},\ \sum_{i=1}^{n} J_{ki} = 1 \text{ for all } k,\ \sum_{k=1}^{N} J_{ki} = u_i \text{ for all } i \right\} \tag{12}$$

An example of how to construct $Y(u)$) is given in [48].

To include the possibility of mutation, we use U to denote the $n \times n$ mutation matrix, where U_{ij} is the probability that x_j mutates to x_i. The probability that $y_{k,t+1} = x_i$ with both migration and mutation considered is denoted as $P_{ki}^{(m)}(v)$ and is given by

$$P_{ki}^{(m)}(v) = \sum_{j=1}^{n} U_{ij} P_{kj}(v) \tag{13}$$

from which we obtain

$$P^{(m)}(v) = \left[P_{ki}^{(m)}(v) \right] = P(v)U^T \tag{14}$$

where the elements of $P(v)$ are given in (10). $P(v)$ is the $N \times n$ matrix containing the probabilities of obtaining each of n possible individuals at each of N immigration trials, if mutation is not considered. $P^{(m)}(v)$ contains those probabilities if both

migration and mutation are considered. In this case we can write the probability of transitioning from population vector v to population vector u after one generation as

$$\Pr^{(m)}(u|v) = \sum_{J \in Y(u)} \prod_{k=1}^{N} \prod_{i=1}^{n} [P_{ki}^{(m)}(v)]^{J_{ki}} \tag{15}$$

Eq. (15) can be used to obtain the transition matrix elements for the Markov model of BBO if both migration and mutation are considered. In Section 3.3 we use standard Markov tools [41] with the transition matrix to find the limiting distribution of the BBO population for a particular problem.

The Markov transition matrix is obtained by computing (15) for each possible v vector and each possible u vector. The transition matrix is therefore a $T \times T$ matrix, where T is the total number of possible population distribution vectors v; that is, T is the number of $n \times 1$ integer vectors whose elements sum to N, and each of whose elements is in $[0, N]$. This number can be calculated several different ways. In [35] the value of T is expressed using the notation for combinations:

$$T = C(n + N - 1, N) \tag{16}$$

Other methods for calculating T are discussed in [48].

3.2. Markov model for genetic algorithm with global uniform recombination

Since BBO reduces to GA/GUR if $\lambda_i = 1$ for all i, the equations in the preceding section all apply to GA/GUR if λ_i is replaced with 1. In BBO, the immigration rate λ_i decreases with the fitness of x_i. In GA/GUR, $\lambda_i = 1$ for all i; in this case, (10) becomes

$$P_{ki}(v) = \Pr(y_{k,t+1} = x_i) = \prod_{s=1}^{q} \left[\frac{\sum_{j \in \mathcal{J}_i(s)} v_j \mu_j}{\sum_{j=1}^{n} v_j \mu_j} \right] \tag{17}$$

while (11)–(16) remain the same. The following section shows that this simple change in selection pressure can make a significant difference in optimization performance between the two algorithms.

3.3. Markov model comparisons

In this section we compare Markov model results for GA with single-point crossover (GA/SP), GA/GUR, and BBO. The Markov model for GA/GUR and BBO is presented in the previous sections. The Markov model for GA/SP is presented in [41,35,9,10]. Due to the factorial increase of the transition matrix dimension that is associated with an increase in population and search space size, we limit our investigation to four-bit problems ($n = 16$) with a population size of four ($N = 4$). This results in 3876 possible population vectors as calculated from (16).

We examine three problems in this section. The first problem is the unimodal one-max problem in which the fitness of each bit string is equal to the number of ones in the bit string. The second problem is a multimodal problem; its fitness values are equal to those of the one-max problem, except that the bit string consisting of all zeros has the same fitness as the bit string consisting of all ones. The third problem is a deceptive problem; its fitness values are equal to those of the one-max problem, except it is a unimodal problem in which the bit string consisting of all zeros has the highest fitness.

Tables 1–3 show comparisons between Markov model results for GA/SP, GA/GUR, and BBO. 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. The mutation rates shown in Tables 1–3 are applied to each bit in each individual. The models have been previously supported with simulation results for various benchmark functions as shown in [46].

We used a crossover probability $p_c = 0.9$ to generate the GA/SP data in Tables 1–3. Performance improves as p_c increases, but the improvement is very small. If we re-create Tables 1–3 using different values of p_c, the numbers in the GA/SP columns change by an average of less than 2% as p_c increases from 0.5 to 1.

Table 1
Unimodal problem optimization results. The results were obtained using Markov models and were supported with simulation results. The best performance is in **bold font** in each row.

Mutation rate	Population vector	Probability		
		GA/SP	GA/GUR	BBO
0.1	All optimal	**0.0084**	0.0079	0.0044
	No optima	0.5826	0.5623	**0.5111**
0.01	All optimal	0.2492	0.2513	**0.3484**
	No optima	0.5436	0.5372	**0.2128**
0.001	All optimal	0.4029	0.4034	**0.7616**
	No optima	0.5696	0.5690	**0.1679**

Table 2
Multimodal problem optimization results. The results were obtained using Markov models and were supported with simulation results. The best performance is in **bold font** in each row.

Mutation rate	Population vector	Probability		
		GA/SP	GA/GUR	BBO
0.1	All optimal	**0.0119**	0.0106	0.0066
	No optima	0.5006	0.4939	**0.4370**
0.01	All optimal	0.3675	0.3701	**0.4715**
	No optima	0.4139	0.4079	**0.1450**
0.001	All optimal	0.5655	0.5670	**0.8502**
	No optima	0.4069	0.4053	**0.0968**

Table 3
Deceptive problem optimization results. The results were obtained using Markov models and were supported with simulation results. The best performance is in **bold font** in each row.

Mutation rate	Population vector	Probability		
		GA/SP	GA/GUR	BBO
0.1	All optimal	**0.01314**	0.0109	0.0120
	No optima	0.8120	0.8325	**0.7954**
0.01	All optimal	0.4601	0.4760	**0.6506**
	No optima	0.4308	0.4103	**0.1915**
0.001	All optimal	0.6230	0.6383	**0.9074**
	No optima	0.3638	0.3482	**0.0730**

3.4. Discussion

Several things are notable about the results in Tables 1–3. We see that in each table, as the mutation rate decreases, performance improves; that is, the probability of obtaining a population of all optimal individuals increases, and the probability of obtaining no optimal individuals decreases. This is true for all three algorithms in all three tables.

GA/SP is the best algorithm only when the mutation rate is high (10% per bit), and only insofar as the probability of obtaining a population of all optimal individuals is slightly higher in GA/SP than in GA/GUR and BBO. In every other performance comparison in the tables, GA/GUR performs slightly better than GA/SP, and BBO performs significantly better than both GA/SP and GA/GUR. This is especially true when the mutation rate is low (0.1% per bit), in which case BBO performs better than GA/SP and GA/GUR in its higher probability of obtaining a population with all optimal indivuals (85 vs. 56%), and in its lower probability of obtaining a population with no optimal individuals (10 vs. 41%).

The best performance in each table is obtained by BBO with a 0.1% mutation rate. The best GA/SP and GA/GUR performance in each table is much worse than the best BBO performance in each table. In the unimodal problem results in Table 1, the lowest BBO probability of no optimal individuals is 17% while the lowest GA probability is 54%. In the multimodal problem results in Table 2, the lowest BBO probability of no optimal individuals is 10% while the lowest GA probability is 41%. In the deceptive problem results in Table 3, the lowest BBO probability of no optimal individuals is 7% while the lowest GA probability is 35%.

Previous papers have compared the optimization performance of multi-parent EAs and have reported similar results; in particular, ES performance has been seen to generally improve as the number of parents increases [15]. However, we see in this section that the difference between GA/SP and GA/GUR is relatively small, while the improved performance that comes with BBO is significant. This is because a BBO individual uses its own fitness before deciding how likely it is to accept features from other solutions. This simple and intuitive idea does not have an analogy in genetics, but is motivated by biogeography. The results in this section are also consistent with [30], which shows that BBO with a constant immigration rate of $\lambda = 1$ (which we have shown reduces BBO to GA/GUR) gives much worse performance than standard BBO ($\lambda_k = 1 - \mu_k$) for a wide range of benchmarks.

4. Probabilities of obtaining and retaining optimal solutions

In Section 4.1 we use the BBO and GA/GUR Markov models to determine the probability of finding an optimal solution in one generation, assuming that the individuals in the population have been randomly and uniformly selected from the search space, and assuming that no mutation occurs. We expect that this probability gives an indication of the relative difficulty of finding an optimal solution over many generations. In Section 4.2 we approximate the probability of retaining an optimal solution in the population under the same conditions. For these analyses, we use the probability of (8) in which the mutation probability is assumed to be zero. In Section 4.3 we discuss our assumptions in more detail and compare BBO and GA/GUR probabilities. In Section 4.4 we support our analyses with benchmark simulations.

4.1. Probability of obtaining an optimum

In this section we first determine the probability that an optimal solution is found by the BBO algorithm in one generation, and then we do the same for GA/GUR. To obtain tractable and specific results for comparison between BBO and GA/GUR, we consider a relatively simple problem with a population denoted as in (3). $\{x_1,\ldots,x_n\}$ is the search space, and we specify the μ and λ values of each x_i as

$$\begin{aligned}\mu &= \{2/3, 1/3, \ldots, 1/3\}\\ \lambda &= \{1/3, 2/3, \ldots, 2/3\}\end{aligned} \tag{18}$$

That is, assuming the linear migration curves of Fig. 1, x_1 is twice as fit as x_i for $i \in \{2,\ldots,n\}$ and is therefore the optimal solution. Suppose that there are no x_1 individuals in the population. Since $y_k \neq x_1$ for all k, we know that

$$v_1 = 0, \quad m(k) \neq 1, \quad \lambda_{m(k)} = 2/3, \quad \mu_k = 1/3, \quad \text{for all } k \in \{1,\ldots,N\} \tag{19}$$

Therefore,

$$\sum_{j\in\mathcal{J}_1(s)} v_j\mu_j = \frac{1}{3}\sum_{j\in\mathcal{J}_1(s)} v_j \tag{20}$$

The search space $\{x_i\}$ consists of n possible solutions. We have N random variables y_k, each having the identical probability mass function

$$\Pr(y_k = x_i) = \begin{cases} 0, & \text{if } i = 1\\ 1/(n-1), & \text{if } i \in \{2,\ldots,n\}\end{cases} \tag{21}$$

Given this random population distribution, and viewing $m(k)$ as a function of the random population vector v as well as k, we calculate the expected value of the random variable $X = \mathbf{1}_0(x_{m(k)}(s) - x_1(s)) \in \{0,1\}$. We will continue to write $m(k)$ instead of $m(k,v)$ for brevity of notation. Recall from (4) that the cardinality of $\mathcal{J}_i(s)$ is $n/2$ for all i and s, so there are $n/2$ search space indices $m(k)$ for which $x_{m(k)}(s) = x_1(s)$, or equivalently, for which $X = 1$. However, since there are no x_1 individuals in the population, there are only $(n/2-1)$ unique population indices k such that $X = 1$. This fact, along with the fact from (21) that there are $(n-1)$ search space indices i such that $y_k = x_i$, gives

$$E[X] = E[\mathbf{1}_0(x_{m(k)}(s) - x_1(s))] = (0)\Pr[X=0] + (1)\Pr[X=1] = \frac{n/2-1}{n-1} \tag{22}$$

A formal proof of (22) is in Appendix A. An argument similar to that given above can be used to show that

$$E\left[\mathbf{1}_0(x_{m(k)}(s) - x_1(s))\mathbf{1}_0(x_{m(k)}(r) - x_1(r))\right] = \left(\frac{n/2-1}{n-1}\right)^2 \tag{23}$$

for all $r \neq s$; that is, the expected value of the product is equal to the product of the expected values. Therefore, $\mathbf{1}_0(x_{m(k)}(s) - x_1(s))$ and $\mathbf{1}_0(x_{m(k)}(r) - x_1(r))$, considered as functions of the random variables v and k, are uncorrelated random variables for $r \neq s$.

We next calculate the expected value of $\sum_{j\in\mathcal{J}_1(s)} v_j$ with respect to the random variable v. For the sake of this calculation, we temporarily re-index the components of v so that

$$\sum_{j\in\mathcal{J}_1(s)} v_j = v_1 + v_2 + \cdots + v_{n/2} \tag{24}$$

Recall from (19) that $v_1 = 0$. Recall from (21) that there are N individuals $y_k = x_{m(k)}$ in the population chosen randomly and uniformly from the remaining $(n-1)$ elements of the diminished search space $\{x_2,\ldots,x_n\}$. Therefore, on average there are $\eta = N/(n-1)$ copies of each search space element x_j $(j \neq 1)$ in the population. We can depict this average population as

$$\{y_1,\ldots,y_N\} = \left\{\underbrace{x_2, x_2, \ldots, x_2}_{v_2=\eta}, \ldots, \underbrace{x_n, x_n, \ldots, x_n}_{v_n=\eta}\right\} \tag{25}$$

This means that each of the components of v other than v_1 has an average value given by $E[v_j] = N/(n-1)$, $j \neq 1$. So we have

$$\begin{aligned} E\left[\sum_{j\in\mathcal{J}_1(s)} v_j\right] &= E[v_1] + E[v_2] + \cdots + E[v_{n/2}]\\ &= 0 + \frac{N}{n-1} + \cdots + \frac{N}{n-1} \qquad (26)\\ &= \frac{N(n/2-1)}{n-1} \qquad (27)\end{aligned}$$

A formal proof of (27) can be constructed along the lines of the proof of (22). An argument similar to that given above can be used to show that

$$E\left[\sum_{j\in\mathscr{J}_1(s)} v_j \sum_{j\in\mathscr{J}_1(r)} v_j\right] = \left(\frac{N(n/2-1)}{n-1}\right)^2 \tag{28}$$

for all $r \neq s$. Therefore, $\sum_{j\in\mathscr{J}_1(s)} v_j$ and $\sum_{j\in\mathscr{J}_1(r)} v_j$, considered as functions of the random variable v, are uncorrelated random variables for $r \neq s$.

Now we use the above results with (8) to calculate the expected value of $\Pr(y_{k,t+1} = x_1)$ with respect to the random variables v and k. We use the fact that the random variables $\sum_{j\in\mathscr{J}_1(s)} v_j$ and $\sum_{j\in\mathscr{J}_1(r)} v_j$ are uncorrelated for $r \neq s$, and $\mathbf{1}_0(x_{m(k)}(s) - x_1(s))$ and $\mathbf{1}_0(x_{m(k)}(r) - x_1(r))$ are uncorrelated for $r \neq s$, to obtain

$$\begin{aligned}
E[\Pr(y_{k,t+1} = x_1)] &= \prod_{s=1}^{q}\left[(1-\lambda_{m(k)})E[\mathbf{1}_0(x_{m(k)}(s) - x_1(s))] + \lambda_{m(k)}\frac{E\left[\sum_{j\in\mathscr{J}_1(s)} v_j\mu_j\right]}{\sum_{j=2}^{n} v_j\mu_j}\right] \\
&= \prod_{s=1}^{q}\left[\left(\frac{1}{3}\right)\left(\frac{n/2-1}{n-1}\right) + \left(\frac{2}{3}\right)\left(\frac{\frac{1}{3}E\left[\sum_{j\in\mathscr{J}_1(s)} v_j\right]}{\frac{1}{3}\sum_{j=2}^{n} v_j}\right)\right] = \prod_{s=1}^{q}\left[\left(\frac{1}{3}\right)\left(\frac{n/2-1}{n-1}\right) + \left(\frac{2}{3}\right)\left(\frac{\left(\frac{1}{3}\right)\frac{N(n-2)}{2(n-1)}}{N/3}\right)\right] \\
&= \left(\frac{n-2}{2(n-1)}\right)^q \approx 2^{-q} = 1/n \quad \text{for large } n.
\end{aligned} \tag{29}$$

The above result shows that the probability of finding the optimal solution, averaged over all population distributions given by (21) and all $k \in \{1,\ldots,N\}$, decreases exponentially with the number of bits in the search space. In addition, the average probability of finding the optimal solution is not a function of the population size N. At first this seems nonintuitive, but the average probability of (29) is averaged over all individuals and all possible population distributions. The average probability of keeping an optimal bit $x_1(s)$ in a random individual y_k is the same as the average probability of migrating an optimal bit into y_k. When applied to real-world problems, EAs need to have a large enough population size N to reasonably cover the search space. The performance of EAs in real problems is therefore highly dependent on N. However, the result of (29) gives the probability of finding an optimal solution averaged over *all possible* populations, assuming that the optimal solution does not yet exist in the population. This average probability is not a function of N.

Now consider the GA/GUR algorithm. We still have the μ values shown in (18), but $\lambda_{m(k)} = 1$ for all k. In this case (8) gives

$$\begin{aligned}
E[\Pr(y_{k,t+1} = x_1)] &= \prod_{s=1}^{q}\left[(1-\lambda_{m(k)})E\left[\mathbf{1}_0(x_{m(k)}(s) - x_1(s))\right] + \lambda_{m(k)}\frac{E\left[\sum_{j\in\mathscr{J}_1(s)} v_j\mu_j\right]}{\sum_{j=2}^{n} v_j\mu_j}\right] = \prod_{s=1}^{q}\left(\frac{\left(\frac{1}{3}\right)\frac{N(n-2)}{2(n-1)}}{N/3}\right) \\
&= \left(\frac{(n-2)}{2(n-1)}\right)^q \approx 2^{-q} = 1/n \quad \text{for large } n
\end{aligned} \tag{30}$$

Comparing (29) and (30), we see that BBO and GA/GUR have the same average probability of obtaining an optimal individual.

4.2. Probability of retaining an optimum

In this section we first determine the probability that an optimum is retained in the BBO algorithm when elitism is not used, and then we do the same for GA/GUR. This analysis is important in EA implementations in which elitism is not used, which will be discussed further in Section 4.3. As above, we consider the relatively simple problem

$$\begin{aligned}
\mu &= \{2/3, 1/3, \ldots, 1/3\} \\
\lambda &= \{1/3, 2/3, \ldots, 2/3\}
\end{aligned} \tag{31}$$

That is, assuming the linear migration rates of Fig. 1, x_1 is twice as fit as x_i for $i \in \{2,\ldots,n\}$ and is therefore the optimal solution. In contrast to the previous section, we now suppose that there is exactly one x_1 individual in the population; that is, $y_{k_0} = x_1$ for some k_0, and $y_k \neq x_1$ for all $k \neq k_0$. This gives

$$v_1 = 1, \quad m(k_0) = 1, \quad \lambda_{m(k_0)} = 1/3, \quad \mu_{m(k_0)} = 2/3 \tag{32}$$

Therefore,

$$\sum_{j\in\mathscr{J}_1(s)} v_j\mu_j = \sum_{j=1}^{1} v_j\mu_j + \sum_{\substack{j\in\mathscr{J}_1(s) \\ j\neq 1}} v_j\mu_j = \frac{2}{3} + \frac{1}{3}\sum_{\substack{j\in\mathscr{J}_1(s) \\ j\neq 1}} v_j \tag{33}$$

The search space $\{x_i\}$ consists of n possible solutions. Since $y_{k_0} = x_1$, it is not random, but the $(N-1)$ individuals $y_k (k \neq k_0)$ are random, each with the identical probability mass function

$$\Pr(y_k = x_i) = \begin{cases} 0, & \text{if } i = 0 \\ 1/(n-1), & \text{if } i \in \{2,\ldots,n\} \end{cases} \tag{34}$$

Given this random population distribution, we calculate the expected value of the sum on the right side of (33) with respect to the random variable v. Recall from (4) that the cardinality of $\mathscr{J}_i(s)$ is $n/2$ for all i and s. However, since the sum on the right

side of (33) does not include the count of x_1 individuals (that is, $j \neq 1$), there are only $(n/2-1)$ terms in the sum. This fact, along with the fact that there are $(n-1)$ search space indices j such that $j \neq 1$, and the fact that $\sum_{j=2}^{n} v_j = N-1$ (since $v_1 = 1$), gives

$$E\left[\sum_{\substack{j\in\mathcal{J}_1(s)\\ j\neq 1}} v_j\right] = (N-1)\Pr(y_k(s) = x_1(s) : k\neq k_0) = \frac{(N-1)(n/2-1)}{n-1} \tag{35}$$

As with (22) and (27), a formal derivation similar to that in Appendix A can also be used to obtain (35).

Now, given that $y_{k_0,t} = x_1$, we use the above results with (8) to calculate the expected value of $\Pr(y_{k_0,t+1} = x_1)$ with respect to the probability density function of v:

$$\begin{aligned}
E[\Pr(y_{k_0,t+1} = x_1)] &= \prod_{s=1}^{q}\left[(1-\lambda_{m(k_0)})E[\mathbf{1}_0(x_{m(k_0)}(s) - x_1(s))] + \lambda_{m(k_0)}\frac{E\left[\sum_{j\in\mathcal{J}_1(s)} v_j\mu_j\right]}{\sum_{j=1}^{n} v_j\mu_j}\right] \\
&= \prod_{s=1}^{q}\left[\left(\frac{2}{3}\right)(1) + \left(\frac{1}{3}\right)\left(\frac{\frac{2}{3} + \frac{(N-1)(n-2)}{6(n-1))}}{\frac{2}{3}+\frac{N-1}{3}}\right)\right] \\
&= \left(\frac{(5N+7)n - 2(3N+1)}{6(n-1)(N+1)}\right)^q \\
&\approx \left(\frac{5N+7}{6N+6}\right)^q \quad \text{for large } n \\
&\approx \left(\frac{5}{6}\right)^q \quad \text{for large } N
\end{aligned} \tag{36}$$

This result shows that if we have an optimum in the population, the probability of keeping that optimum from one generation to the next decreases exponentially with the number of bits in the search space. However, the rate of decrease is not nearly as severe as it is for the probability of finding an optimum in the first place (compare (36) with (29)).

We also note from (36) that as the population size N increases, the probability of retaining the optimum decreases. The probability is 1 for $N = 1$, and asymptotically decreases to $(5/6)^q$ as N approaches infinity. This agrees with intuition. A larger population results in a greater chance of immigrating a nonoptimal bit (that is, the complement of $x_1(s)$) to the optimal solution. This is because the μ values are used to probabilistically select the emigrating solution, and a larger population gives a larger piece of the roulette wheel to the nonoptimal solutions.

Now consider the GA/GUR algorithm. We still have the μ values shown in (31), but $\lambda_{m(k)} = 1$. In this case we obtain

$$\begin{aligned}
E[\Pr(y_{k,t+1} = x_1)] &= \prod_{s=1}^{q}\left[(1-\lambda_{m(k)})E[\mathbf{1}_0(x_{m(k)}(s) - x_1(s))] + \lambda_{m(k)}\frac{E\left[\sum_{j\in\mathcal{J}_1(s)} v_j\mu_j\right]}{\sum_{j=1}^{n} v_j\mu_j}\right] \\
&= \prod_{s=1}^{q}\frac{4(n-1) + (N-1)(n-2)}{2(n-1)(N+1)} \\
&= \left(\frac{(N+3)n - 2(N+1)}{2(n-1)(N+1)}\right)^q \\
&\approx \left(\frac{N+3}{2N+2}\right)^q \quad \text{for large } n \\
&\approx \left(\frac{1}{2}\right)^q \quad \text{for large } N
\end{aligned} \tag{37}$$

This result shows that if we have an optimum in the population, the probability of keeping that optimum from one generation to the next decreases exponentially with the number of bits in the search space. Furthermore, the rate of decrease is more severe than that of BBO (compare with (36)). Similar to the discussion following (36), we note from (37) that as the population size N increases, the probability of retaining the optimum decreases. The probability is 1 for $N = 1$, and asymptotically decreases to $(1/2)^q$ as N approaches infinity.

4.3. Discussion of analytical results

4.3.1. Assumptions

The preceding analysis was conducted under several simplifying assumptions. We assumed a domain of q-bit individuals with a corresponding search space cardinality of $n = 2^q$, and we analyzed the behavior of the algorithms for a single generation. We assumed that all individuals in the search space had the same fitness except for the global optimum, whose fitness was twice that of the other individuals. Although many optimization problems do not satisfy this assumption, some of them do, such as "needle-in-a-haystack" problems [34]. We assumed that the population was uniformly distributed throughout the search space, which is a reasonable assumption given our assumed fitness function [11]. Although our analysis was

conducted under special conditions, our simulation results in Section 4.4 will show that it provides a correct qualitative description of BBO and GA/GUR behavior for standard benchmark problems.

We also assumed that no elitism was used. We can always implement elitism to guarantee that optima are retained in the population, but sometimes it is desirable to use nonelitist algorithms. For example, fitness evaluations for real-world problems can be very expensive [5], requiring computation, simulations, or experiments that take on the order of days or even weeks. The expense of fitness function evaluation is a common obstacle in the implementation of EAs, and has given rise to recent research in methods for reducing the number of fitness function evaluations [39] and methods for fitness function approximation [22]. It is desirable to use small population sizes for problems with expensive fitness function evaluations, which in turn makes it desirable to use nonelitist EAs [53]. Even with elitist EAs, it is often desirable to find multiple optima, or to find multiple solutions near the global optimum [29], which in turn gives importance to the probability of optimum retention.

4.3.2. Summary, comparisons, and implications

The analysis of the preceding sections can be summarized as

$$E[\Pr(\text{finding an optimum})] \approx \begin{cases} (1/2)^q & \text{BBO} \\ (1/2)^q & \text{GA/GUR} \end{cases}$$

$$E[\Pr(\text{retaining an optimum})] \approx \begin{cases} (5/6)^q & \text{BBO} \\ (1/2)^q & \text{GA/GUR} \end{cases} \tag{38}$$

With the assumptions stated at the beginning of this section, both BBO and GA/GUR have equal chances of finding an optimum, but BBO is much better than GA/GUR at retaining an optimum once it is found. This is due to BBO's immigration rate, which decreases with fitness, and which tends to preserve good solutions in the population. Futhermore, (38) shows that the advantage of BBO over GA/GUR is more pronounced with larger problems (that is, larger q). We therefore expect BBO to be particularly advantageous for problems with high dimensions.

We next consider the effect of population size on performance. Eqs. (36) and (37) are repeated here with the large n approximation, but without the large N approximation:

$$\begin{aligned} &\text{BBO} : E[\Pr(\text{retaining an optimum})] \approx \left(\frac{5N+7}{6(N+1)}\right)^q \\ &\text{GA/GUR} : E[\Pr(\text{retaining an optimum})] \approx \left(\frac{N+3}{2(N+1)}\right)^q \end{aligned} \tag{39}$$

We can use these equations to obtain the performance of BBO relative to GA/GUR as a function of population size N:

$$\frac{\text{BBO performance}}{\text{GA/GUR performance}} = \left(\frac{5N+7}{3N+9}\right)^q \tag{40}$$

This equation confirms, for the special case discussed above, that relative BBO performance is better for larger q (that is, larger problem dimensions). It also shows that relative BBO performance is better for larger N (that is, larger populations). For $N = 1$, (40) evaluates to 1; that is, BBO and GA/GUR performance are equal. As N increases, (40) asymptotically approaches $(5/3)^q$.

In summary, we expect BBO to outperform GA/GUR for all problem sizes and all population sizes. But we expect BBO to be particularly advantageous for high-dimension problems and with large populations.

4.4. Simulation results

This section supports the analysis of the preceding sections with simulations. The benchmarks that we used are representative of those published in the literature for comparison of optimization methods. We chose benchmarks that could be used with a variable number of dimensions so that we could explore the effect of changing dimensions. The functions are summarized in Table 4, which shows that they have a variety of characteristics. Multimodal functions are those which have multiple minima. An n-dimensional separable function is one which can be reduced to n independent one-dimensional functions. A regular function is one which is differentiable. More information about these functions can be found in [2,58,7].

First we compare BBO and GA/GUR for different problem dimensions. We ran 100 Monte Carlo simulations of BBO and GA/GUR for each of the 14 benchmarks using a population size of 50. We then took the minimum function value achieved among the 100 BBO runs and the mininum achieved among the 100 GA/GUR runs, and counted the number of benchmarks for which the BBO minimum was better than the GA/GUR minimum. The results are shown in Table 5, where it is seen that BBO performance improves relative to GA/GUR as the problem dimension increases. At a low problem dimension of 5, BBO performs better than GA/GUR on 4 out of 14 benchmarks. As the problem dimension increases to 30, BBO performs better than GA/GUR on 14 out of 14 benchmarks. These results support the analysis of Section 4.3. Appendix B gives a more detailed view of the data shown in Table 5, including statistical analyses.

Table 4
Benchmark function characteristics; n is the number of dimensions of the problem.

Function	Multimodal?	Separable?	Regular?	Domain
Ackley	Yes	No	Yes	$[\pm 30]^n$
Fletcher-Powell	Yes	No	No	$[\pm \pi]^n$
Griewank	Yes	No	Yes	$[\pm 600]^n$
Penalty #1	Yes	No	Yes	$[\pm 50]^n$
Penalty #2	Yes	No	Yes	$[\pm 50]^n$
Quartic	No	Yes	Yes	$[\pm 1.28]^n$
Rastrigin	Yes	Yes	Yes	$[\pm 5.12]^n$
Rosenbrock	No	No	Yes	$[\pm 2.048]^n$
Schwefel 1.2	No	No	Yes	$[\pm 65.536]^n$
Schwefel 2.21	No	No	No	$[\pm 100]^n$
Schwefel 2.22	Yes	No	No	$[\pm 10]^n$
Schwefel 2.26	Yes	Yes	No	$[\pm 512]^n$
Sphere	No	Yes	Yes	$[\pm 5.12]^n$
Step	No	Yes	No	$[\pm 200]^n$

Table 5
Number of benchmarks for which BBO performs better than GA/GUR, where the total number of benchmarks is 14. The data shows that BBO performance improves relative to GA/GUR as the problem dimension increases. Population size is 50 and results are based on 100 Monte Carlo simulations.

Problem dimension	BBO wins
5	4
10	9
20	11
30	14

Table 6
Number of benchmarks for which BBO performs better than GA/GUR, where the total number of benchmarks is 14. The data shows that BBO performance improves relative to GA/GUR as the population size increases. Problem dimension is 30 and results are based on 100 Monte Carlo simulations.

Population size	BBO wins
10	10
20	11
50	14

Next we compare BBO and GA/GUR for different population sizes. As above, we ran 100 Monte Carlo simulations of BBO and GA/GUR for each of the 14 benchmarks. We set the problem dimension to 30. We again took the minimum function value achieved among the 100 BBO runs and the mininum achieved among the 100 GA/GUR runs, and counted the number of benchmarks for which the BBO minimum was better than the GA/GUR minimum. The results are shown in Table 6, where it is seen that BBO performance improves relative to GA/GUR as the population size increases. At a small population size of 10, BBO performs better than GA/GUR on 10 out of 14 benchmarks. As the population size increases to 50, BBO performs better than GA/GUR on 14 out of 14 benchmarks. These results support the analysis of Section 4.3. Appendix C gives a more detailed view of the data shown in Table 6, including statistical analyses.

5. Combinatorial optimization

This section compares BBO and GA/GUR on combinatorial benchmark problems. We consider the traveling salesman problem (TSP) in Section 5.1, the graph coloring problem in Section 5.2, and the bin packing problem in Section 5.3.

5.1. The traveling salesman problem

BBO was first applied to the TSP in [33], where a comparison was made with ant colony optimization, genetic algorithms, particle swarm optimization, immune algorithms, and fish swarm algorithms. This section compares BBO with GA/GUR on some TSP benchmarks.

There are many different ways to adapt EAs to solve the TSP. This section is based on the inver-over operator [52], which has proven to be an effective heuristic for the TSP. The procedure of the inver-over operator is described as follows.

1. We begin with two parent individuals, p_1 and p_2, each with an ordered list of cities. The selection of the parents is what distinguishes GA/SP, GA/GUR, and BBO from each other, and this will be discussed later in this section.
2. Given two parent individuals, we randomly select a city c_1 from p_1, and then select the city immediately following c_1 as the starting city c_s.
3. We find c_1 in p_2, and select the city that follows it as the ending city c_e.
4. We find c_e in p_1. We then reverse the sequence of cities between c_s and c_e in p_1. This new ordering of cities is the offspring.

The above sequence of operations results in c_s following c_e in the offspring, just as it did in p_2. The ordering of the other cities is the same in the offspring as it was in p_1. The result is that information is copied from p_2 to p_1 to create the offspring.

Fig. 4 illustrates the inver-over operator. First we randomly select c_1 from p_1, which in this case is city 3. We see that city 2 follows city 3 in p_1, so c_s = city 2. Then we find c_1 in p_2. We see that city 1 follows c_1 in p_2, so c_e = city 1. We then go back to p_1 and find c_e in p_1. We reverse the sequence of cities between c_s and c_e in p_1, and this results in the offspring.

The difference between a GA and a BBO implementation of the inver-over operator lies in the selection of the parents. With GA/SP, both parents are selected on the basis of their fitness values (for example, using roulette-wheel selection). This results in the selection of two parents which probably have a high fitness. With GA/GUR, p_1 is randomly selected with a uniform distribution from the entire population. This corresponds to an immigration probability of $1/N$ for each individual, where N is the population size. The emigrating individual p_2 is selected on the basis of fitness, just as with GA/SP. With BBO, each individual is ranked according to fitness, assigned an emigration probability μ that increases with fitness, and an immigration probability λ that decreases with fitness. Then p_1 is selected using the immigration probabilities, and p_2 is selected using the emigration probabilities. This results in an immigrating parent that probably has a low fitness, and an emigrating parent that probably has a high fitness. These differences are summarized in Table 7.

We evaluated GA/GUR and BBO on five TSP benchmarks, all of which are available in [40]. Berlin-52 is a data set of 52 locations in Berlin, Germany. ST-70 is a 70-city problem, CH-130 is a 130-city problem, GR-202 is a 202-city problem, and RAT-575 is a 575-city problem. For both GA/GUR and BBO we used a population size of 50 and an elitism parameter of 5. Table 8 shows the best GA/GUR and BBO individual after 100 generations, averaged over 100 Monte Carlo simulations. The table also shows the t-test results of the two sets of simulations. The algorithms are still converging after 100 generations, so the numbers in Table 8 should not be compared with the best published solutions, but only with each other to measure the effect of using BBO instead of GA/GUR for parent selection. Table 8 shows that BBO is significantly better than GA/GUR for all five of the benchmarks.

5.2. Graph coloring

In the graph coloring problem, we are given a set of countries (vertices) on a map. Each country has some neighbors. Neighboring countries are represented as vertices that are connected with an edge. Our goal is to find the minimum number of colors with which we can color the vertices, under the constraint that connected vertices cannot share the same color. The minimum number of colors is denoted as the chromatic number, $\chi(G)$ [31].

Fig. 5 illustrates a graph coloring problem with eight vertices and 12 edges, along with its solution. The minimum number of colors needed to solve the problem is $\chi(G) = 3$. We have used different shapes instead of different colors in Fig. 5 for the

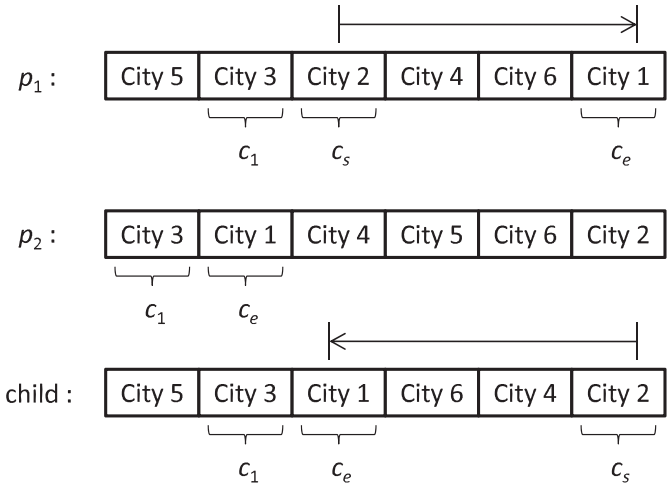

Fig. 4. Illustration of the inver-over operator for the TSP. Two parent individuals combine to form a child individual. See the text for a detailed description.

Table 7
TSP parent selection probabilities used by GA/SP, GA/GUR, and BBO. p_1 is the immigrating parent that receives information from the emigrating parent p_2. N is the population size, μ is normalized fitness, and $\lambda = 1 - \mu$.

	GA/SP	GA/GUR	BBO
Parent p_1	μ	$1/N$	μ
Parent p_2	μ	λ	λ

Table 8
Comparison of the cost (traveling distance) between GA/GUR and BBO for traveling salesman benchmarks. The numbers show the average of the best performance of 100 Monte Carlo simulations after 100 generations. The "Prob." column shows the probability that the GA/GUR and BBO results are from the same distribution.

TSP problem	GA/GUR	BBO	Prob.
Berlin-52	17,294	15,916	0.1667
ST-70	2377	2170	0.0585
CH-130	36,175	34,250	0.0132
GR-202	2701	2579	0.0001
RAT-575	104,210	102,952	0.0043

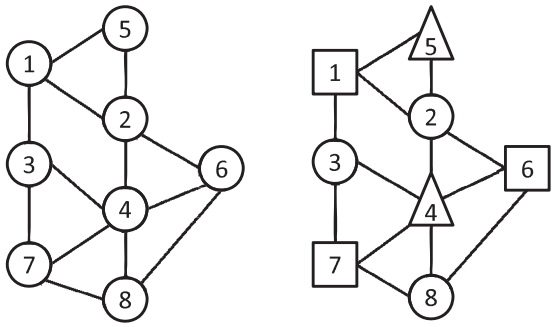

Fig. 5. Illustration of a graph coloring problem with eight vertices and 12 edges. The unsolved graph is shown on the left, and the solved graph is shown on the right, where we have used shapes to indicate colors. This problem can be solved with a minimum of three colors.

sake of illustration. Note from the right side of Fig. 5 that none of the circles are connected to each other, none of the squares are connected to each other, and none of the triangles are connected to each other.

One way to solve the graph coloring problem with an EA is to combine a greedy algorithm with the TSP inver-over operator. With this approach, each individual in the EA population stores an ordered list of vertices as its solution features. Information about the order of the vertices is shared among individuals using the inver-over operator. The greedy algorithm of Fig. 6 assigns colors to the vertices. The number of colors assigned by the greedy algorithm to a given individual is the cost of that individual, and is used to assign emigration and immigration rates.

We evaluated three graph coloring benchmarks from [54]. The first benchmark is the Leighton graph from [25] and includes 450 vertices and 17,343 edges. The second benchmark is based on the Mycielski transformation and includes 191 vertices and 2360 edges. The third benchmark is flat (that is, it has a uniform distribution of edges), and includes 300 vertices and 21,695 edges.

$C =$ indices of available colors
For each vertex v_i
    $N_i \leftarrow \{\text{neighbors of } v_i\}$
    $C(N_i) \leftarrow \{\text{colors of } x : x \in N_i\}$
    $c_i \leftarrow \min\{k \in C : k \notin C(N_i)\}$
    Assign c_i as the color of v_i
Next vertex

Fig. 6. The greedy graph coloring algorithm. For each vertex, this algorithm finds the next available color that is not used by a neighbor. The colors assigned to the vertices depend on the order of the vertices.

Table 9
Comparison of cost (number of colors) between GA/GUR and BBO for graph coloring benchmarks. The numbers show the average of the best performance of 30 Monte Carlo simulations after 100 generations. The "Prob." column shows the probability that the GA/GUR and BBO results are from the same distribution.

Problem	GA/GUR	BBO	Prob.
Leighton	93.1	92.1	0.1053
Mycielski	26.1	26.1	1.0000
Flat	137.9	135.4	0.0060

For both GA/GUR and BBO we used a population size of 50 and an elitism parameter of 3. Table 9 shows the best GA/GUR and BBO individual after 100 generations, averaged over 30 Monte Carlo simulations. The table also shows the *t*-test results of the two sets of simulations. We see from Table 9 that the improvement of BBO over GA/GUR is statistically significant for the Leighton and Flat graph coloring problems, but BBO and GA/GUR have the same performance for the Mycielski problem. Further research could focus on determining the types of graph coloring problems for which BBO performs better than GA/GUR, and the reasons.

5.3. Bin packing

In the bin packing problem, we are given a list of items with various weights. Our problem is to pack the items in the fewest bins, each of which have the same capacity [26].

One way to solve the bin packing problem with an EA is to combine a greedy algorithm with the TSP inver-over operator. Each individual in the EA population stores an ordered list of items as its solution features. Information about the order of the items is shared between individuals using the inver-over operator. Given an ordered list of items, the greedy algorithm of Fig. 7 or Fig. 8 places the items in bins. The number of bins used by the greedy algorithm is the cost of that individual, and is used to assign emigration and immigration rates.

We evaluated six bin packing benchmarks. The first three benchmarks are taken from [17,23]. The first benchmark is called Binpack-1 and includes 120 items with weights uniformly distributed between 20 and 100, to be packed into bins with a capacity of 150. The second benchmark is called Binpack-4; it is the same as Binpack-1 except that it includes 1000 items. The third benchmark is called Binpack-8 and includes 501 items and bin capacities of 1000. The item weights are carefully selected so that one big item and two small items in each bin is optimal.

The other three bin packing benchmarks are called Hard-0, Hard-1, and Hard-9 [44]. Each of these benchmarks have 200 items with weights drawn from a uniform distribution on $[20,35] \times 10^3$, to be packed into bins of capacity 10^5.

For both GA/GUR and BBO we used a population size of 50 and an elitism parameter of 3. Table 10 shows the best GA/GUR and BBO individual after 100 generations, averaged over 30 Monte Carlo simulations. The table also shows the *t*-test results of the two sets of simulations. We see from Table 10 that there is not much difference between GA/GUR and BBO for the bin packing benchmarks. GA/GUR is better for three of the benchmarks, while BBO is better for the other three benchmarks. The only problem for which the performance difference is statistically significant is the Binpack-4 problem, which BBO solves better than GA/GUR. Table 10 can be interpreted to mean that the combination of the inver-over operator with greedy bin packing algorithms is not a promising BBO strategy, and it needs to be modified for better results. This is left for future work.

$B = \{b\}$ = indices of available bins
$C(b)$ = residual capacity of bin b
$I = \{i\}$ = indices of items
$W(i)$ = weight of item i
For each item i
    $\beta \leftarrow \min\{b \in B : C(b) \geq W(i)\}$
    Place item i in bin β
    $C(\beta) \leftarrow C(\beta) - W(i)$
Next item

Fig. 7. The first-fit greedy bin packing algorithm. Each item is placed into the first bin in which it fits. The bins into which each item is placed depends on the order of the items.

```
B = {b} = indices of available bins
C(b) = residual capacity of bin b
I = {i} = indices of items
W(i) = weight of item i
For each item i
    Sort B from least to greatest C(b)
    For each bin b
        If C(b) ≥ W(i)
            Place item i in bin b
            C(b) ← C(b) − W(i)
        End if
    Next bin
Next item
```

Fig. 8. The best-fit greedy bin packing algorithm. Bins with a smaller residual capacity are filled before bins with a larger residual capacity. This is an attempt to fill the bins more efficiently than the first-fit greedy algorithm. The bins into which each item is placed depends on the order of the items.

Table 10
Comparison of cost (number of bins) between GA/GUR and BBO for bin packing benchmarks. The numbers show the average of the best performance of 30 Monte Carlo simulations after 100 generations. The "Prob." column shows the probability that the GA/GUR and BBO results are from the same distribution.

Problem	GA/GUR	BBO	Prob.
Binpack-1	49.6	49.7	0.18
Binpack-4	414.0	413.7	0.04
Binpack-8	179.4	179.2	0.23
Hard-0	58.9	58.8	0.33
Hard-1	59.8	59.9	0.72
Hard-9	59.0	59.1	0.56

6. Conclusion

We have explored the similarities, differences, and relationships between BBO and GA/GUR conceptually, analytically, and through simulation. We have seen that if the immigration rate λ_k is 1 for all BBO individuals, then BBO reduces to a special case of GA/GUR. BBO is therefore a generalization of GA/GUR. Due to its non-uniform immigration rate, BBO can be viewed as including additional "selection pressure" that is missing from GA/GUR. BBO does not include selection pressure in the sense that the term is typically used in the context of reproduction algorithms, but BBO does include selection pressure in a broader sense.

On one hand, the similarity between BBO and GA/GUR is not surprising because we see similarities between many EAs. On the other hand, the similarity between BBO and GA/GUR is surprising in view of the fact that their biological motivations are so different. The view of natural biogeography as an optimization process has guided the development of BBO, and it can motivate many extensions also. These extensions should be emphasized in future BBO research and include such factors as modeling for nonlinear migration curves [30], species populations, predator/prey relationships, species mobilities, direction momentum during migration, habitat area, and habitat isolation.

We have summarized a Markov model of BBO and compared it with Markov models for GA/SP and GA/GUR. Results from the Markov models provide theoretical evidence of strong differences in performance between BBO, GA/SP, and GA/GUR. We have seen that for both unimodal and multimodal problems, the probability of obtaining a population of all optimal individuals in GA/SP is slightly better than BBO when a high mutation rate is used (10% per bit). However, BBO performs significantly better than both GA/SP and GA/GUR for lower mutation rates. Although our theoretical Markov results are limited to small problem dimensions due to the factorial increase in the size of the Markov transition matrix with problem dimension, these results provide confidence for the successful application of BBO to larger, real-world problems.

We used the BBO and GA/GUR Markov models to approximate the probability of finding and retaining an optimal solution in the population in one generation. Based on our analytical results, BBO's performance matches or exceeds GA/GUR in these areas, and its improved performance becomes more pronounced as the problem dimension and the population size becomes larger. Although the theoretical results in this comparison were only approximate, they were supported with a set of benchmark comparisons.

We compared BBO and GA/GUR on combinatorial benchmarks. We found that BBO consistently performs much better on TSP benchmarks, usually performs better on graph coloring benchmarks, and sometimes performs better on bin packing benchmarks. The graph coloring and bin packing studies are preliminary, and more research is required to develop a robust and widely applicable BBO algorithm for these problems.

For future work we see several important directions. The first is to extend BBO in the many directions indicated by natural biogeography theory, as mentioned earlier in this conclusion. Another direction for future work is the development of hybrid BBO algorithms. Since no single algorithm can provide optimal performance for all possible problems [20], hybrid and adaptive algorithms are an important topic of current research. BBO has already been combined with opposition-based learning [16] and differential evolution [18]. Additional work in this area could focus on combining BBO with other algorithms, and using benchmarks and real-world optimization problems to evaluate the performance of these hybrid algorithms.

After BBO is combined with other EAs, theoretical approaches such as Markov theory should be used to explore the characteristics of the combined algorithms. With analytical results such as those provided by Markov theory, we do not rely on the stochastic nature of simulation studies to draw conclusions about performance, but we use simulations to support theoretical results and to probe the limits of the theory.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 0826124.

Appendix A

In this appendix we derive (22), which gives the expected value of $\mathbf{1}_0(x_{m(k)}(s) - x_1(s))$. The expected value is taken with respect to the random population distribution V (of which v is a realization) and the random population index K (of which k is a realization). For ease of notation we define

$$g(K,V) = \mathbf{1}_0(x_{m(K,V)}(s) - x_1(s)) = \mathbf{1}_0(y_{K,V}(s) - x_1(s)) \tag{41}$$

where we have used (6). Then the expected value of $g(K,V)$ with respect to K is

$$E_K[g(K,V)] = \sum_{k=1}^{N} g(k,V)p_K(k) \tag{42}$$

where $p_K(k)$ is the probability mass function of the random variable K. Since K has a uniform distribution, we have $p_K(k) = 1/N$ for $k = 1,\ldots,N$. Therefore,

$$E_K[g(K,V)] = \frac{1}{N}\sum_{k=1}^{N} g(k,V) = \frac{1}{N}\sum_{k=1}^{N} \mathbf{1}_0(y_{k,V}(s) - x_1(s)) \tag{43}$$

Next we take the expected value of $E_K[g(K,V)]$ with respect to the random population distribution V:

$$E_V\{E_K[g(K,V)]\} = \frac{1}{N}E_V\left\{\sum_{k=1}^{N} \mathbf{1}_0(y_{k,V}(s) - x_1(s))\right\} \tag{44}$$

As shown in (16), there are a total of $C(n+N-1,N)$ possible population distributions. However, in our derivation of (22) we assume that there are no x_1 individuals in the population, so we have a total of $C(n+N-2,N)$ possible population distributions. Each population distribution has an equal probability of occurrence, so (44) can be written as

$$E_V\{E_K[g(K,V)]\} = \frac{1}{ND}\sum_{j=1}^{D}\sum_{k=1}^{N} \mathbf{1}_0(y_{k,v(j)}(s) - x_1(s)) \tag{45}$$

where

$$D = C(n+N-2,N) \tag{46}$$

and $v(j)$ is the jth possible population distribution. Without loss of generality, we assume that x_1 is the binary string containing all zeros, that $s = 1$, that the bits are indexed beginning with the left-most bit, and that the x_i's are in natural binary order. Then $y_{k,v(j)}(s) = x_1(s)$ if and only if $y_{k,v(j)}(1) = 0$, which is true if and only if $y_{k,v(j)} = x_i$ for some $i \in [2, n/2]$. (Recall that the derivation of (22) assumes that $y_{k,v(j)} \neq x_1$ for all k, $v(j)$.) Eq. (45) can then be written as

$$E_V\{E_K[g(K,V)]\} = \frac{1}{ND}\sum_{j=1}^{D}\sum_{k=1}^{N}\sum_{i=2}^{n/2} \mathbf{1}_0(y_{k,v(j)} - x_i) = \frac{1}{ND}\sum_{i=2}^{n/2}\sum_{j=1}^{D}\sum_{k=1}^{N} \mathbf{1}_0(y_{k,v(j)} - x_i) \tag{47}$$

The summation over k is nonzero only if $v_i(j)$ is nonzero, that is, only if at least one of the $y_{k,v(j)}$ individuals is equal to x_i. This means that the summation over j can be restricted to those values which have a nonzero $v_i(j)$. Since our population size is N, the summation over v can thus be restricted to those populations for which $v_i(j) \in [1,N]$, resulting in

$$E_V\{E_K[g(K,V)]\} = \frac{1}{ND}\sum_{i=2}^{n/2}\sum_{\substack{j=1\\ v_i(j)\in[1,N]}}^{D}\sum_{k=1}^{N}\mathbf{1}_0(y_{k,v(j)} - x_i) \tag{48}$$

Now note that $\sum_{k=1}^{N}\mathbf{1}_0(y_{k,v(j)} - x_i)$ is simply equal to the total number of individuals in the jth population that are equal to x_i, which is equal to $v_i(j)$, as shown in (3). Eq. (48) therefore becomes

$$E_V\{E_K[g(K,V)]\} = \frac{1}{ND}\sum_{i=2}^{n/2}\sum_{\substack{j=1\\ v_i(j)\in[1,N]}}^{D} v_i(j) \tag{49}$$

Recall that $\sum_{i=1}^{n} v_i(j) = N$, as shown in (2). Suppose that $v_i(j) = \alpha$ for some i, j, and for some $\alpha \in [1,N]$. This means that besides the α individuals that are equal to x_i, there are $(N-\alpha)$ additional individuals which are distributed across the $(n-2)$ remaining search space indices. Generalizing (16), we see that there are a total of $C((n-2)+(N-\alpha)-1, N-\alpha) = C(n+N-\alpha-3, N-\alpha)$ ways to accomplish this distribution. Eq. (49) can therefore be written as

$$E_V\{E_K[g(K,V)]\} = \frac{1}{ND}\sum_{i=2}^{n/2}\sum_{\alpha=1}^{N}\alpha C(n+N-\alpha-3, N-\alpha) = \frac{n/2-1}{ND}\sum_{\alpha=0}^{N-1}(N-\alpha)C(n+\alpha-3,\alpha) = \frac{n/2-1}{n-1} \tag{50}$$

where we have used Theorem 1 (see below) to take the final step and obtain (22) as desired.

Theorem 1. *Suppose that $n, N \in \mathbb{N}$ are natural numbers with $n \geqslant 3$ and $N \geqslant 1$. Then*

$$\sum_{j=0}^{N-1}(N-j)C(n+j-3,j) = \frac{N}{n-1}C(n+N-2,N) \tag{51}$$

Proof. We let $n \in \mathbb{N}, n \geqslant 3$ be an arbitrary but fixed natural number. We apply the principle of mathematical induction (PMI) to the proposition $P(N)$ of (51), which we rewrite as

$$P(N): \sum_{j=0}^{N-1}(N-j)\frac{(n+j-3)!}{j!(n-3)!} = \frac{N}{n-1}\frac{(n+N-2)!}{N!(n-2)!} = \frac{(n+N-2)!}{(N-1)!(n-1)!} \tag{52}$$

It is straightforward to show that $P(1)$ is true; substituting $N = 1$ in (52) results in the identity $1 = 1$. Next, assuming that $P(N)$ is true for some $N \geqslant 1$, we write

$$\begin{aligned}\sum_{j=0}^{N}(N+1-j)\frac{(n+j-3)!}{j!(n-3)!} &= \sum_{j=0}^{N-1}(N+1-j)\frac{(n+j-3)!}{j!(n-3)!} + \frac{(n+N-3)!}{N!(n-3)!}\\ &= \sum_{j=0}^{N-1}(N-j)\frac{(n+j-3)!}{j!(n-3)!} + \sum_{j=0}^{N-1}\frac{(n+j-3)!}{j!(n-3)!} + \frac{(n+N-3)!}{N!(n-3)!}\end{aligned} \tag{53}$$

Applying the inductive hypothesis (52) to the first term on the right side of (53) gives

$$\sum_{j=0}^{N}(N+1-j)\frac{(n+j-3)!}{j!(n-3)!} = \frac{(n+N-2)!}{(N-1)!(n-1)!} + \sum_{j=0}^{N-1}\frac{(n+j-3)!}{j!(n-3)!} + \frac{(n+N-3)!}{N!(n-3)!} \tag{54}$$

Applying Lemma 1, which follows this proof, to the second term on the right side of (54) gives

$$\begin{aligned}\sum_{j=0}^{N}(N+1-j)\frac{(n+j-3)!}{j!(n-3)!} &= \frac{N(n+N-2)!}{N!(n-1)!} + \frac{N(n+N-3)!}{N!(n-2)!} + \frac{(n+N-3)!}{N!(n-3)!}\\ &= \frac{N(n+N-2)!}{N!(n-1)!} + \frac{N(n-1)(n+N-3)!}{N!(n-1)!} + \frac{(n-2)(n-1)(n+N-3)!}{N!(n-1)!}\\ &= \frac{1}{N!(n-1)!}[N(n+N-2)! + N(n-1)(n+N-3)! + (n-2)(n-1)(n+N-3)!]\\ &= \frac{1}{N!(n-1)!}[N(n+N-2)! + (n-1)(n+N-3)!(n+N-2)]\\ &= \frac{1}{N!(n-1)!}[N(n+N-2)! + (n-1)(n+N-2)!] = \frac{(n+N-2)!}{N!(n-1)!}(n+N-1)\\ &= \frac{(n+N-1)!}{N!(n-1)!}\end{aligned} \tag{55}$$

According to the PMI, $P(N)$ is true for all $N \geqslant 1$. Since $n \geqslant 3$ is arbitrary, (51) is true for all $n \geqslant 3$ and for all $N \geqslant 1$. □

Table 11
Comparison between BBO and GA/GUR performance on 5-dimensional benchmark problems with a population size of 50. The "Prob." column shows the probability that the BBO and GA/GUR results are from the same distribution.

	BBO best	GA best	BBO ave.	GA ave.	BBO σ	GA σ	Prob.
Ackley	0.036	0	2.9	3.6	1.2	1.7	2.7e−8
Fletcher	3	0.72	2.6e+3	5.2e+3	3.1e+2	6.9e+2	1.3e−3
Griewank	1.0003	1.0032	1.020	1.037	0.022	0.027	6.2e−6
Penalty #1	1.9e−6	0.00024	0.34	0.27	0.04	0.068	3.5e−4
Penalty #2	3.4e−5	0.012	0.59	1.0	0.11	0.19	3.4e−8
Quartic	0	0	6.2e−22	6.1e−7	6.2e−24	1e−8	2.0e−1
Rastrigin	0	0	0.9	2	0.033	0.16	2.2e−3
Rosenbrock	0	0	3.8	7.9	1.8	2.0	2.4e−2
Schwefel 1.2	0.34	2.8	1.5e+2	2.9e+2	16	61	7.8e−13
Schwefel 2.21	0.26	0.068	4.1	5	1.7	2.3	2.4e−5
Schwefel 2.22	0	0	0.29	0.48	0.048	0.1	5.5e−5
Schwefel 2.26	0.26	0.44	30	40	5.8	9.6	1.5e−5
Sphere	0	0	0.0064	0.024	0.00022	0.0018	6.4e−5
Step	0	0	15	14	2.5	4.1	5.8e−6
BBO wins	57%		85%		92%		-

Table 12
Comparison between BBO and GA/GUR performance on 10-dimensional benchmark problems with a population size of 50. The "Prob." column shows the probability that the BBO and GA/GUR results are from the same distribution.

	BBO best	GA best	BBO ave.	GA ave.	BBO σ	GA σ	Prob.
Ackley	0.75	0.82	4.1	4.9	2.6	3.1	6.6e−9
Fletcher	3.5e+2	7.8e+2	1.5e+4	2.7e+4	4.2e+3	6.6e+3	1.2e−4
Griewank	1.000	1.016	1.6	1.8	1.2	1.3	1.0e−3
Penalty #1	0.015	0.088	1.8	1.4	0.36	0.51	2.9e−5
Penalty #2	0.14	0.39	3.7	5.5	1.2	2.0	6.2e−14
Quartic	0	0	1.9e−7	0.00019	2e−9	2.2e−6	2.0e−1
Rastrigin	0	0	3	3	0.28	0.75	1.1e−9
Rosenbrock	0	5.5	82	74	14	20	1.0e−2
Schwefel 1.2	55	89	1.2e+3	2.8e+3	3.8e+2	7.9e+2	2.6e−12
Schwefel 2.21	3.8	3.5	15	23	8.2	10	2.5e−2
Schwefel 2.22	0	0	0.64	0.96	0.22	0.41	2.4e−6
Schwefel 2.26	5.5	13	1.8e+2	1.5e+2	48	70	1.6e−9
Sphere	0	0	0.11	0.17	0.0077	0.023	3.7e−3
Step	5	8	74	1.4e+2	25	38	1.5e−8
BBO wins	90%		77%		100%		–

Table 13
Comparison between BBO and GA/GUR performance on 20-dimensional benchmark problems with a population size of 50. The "Prob." column shows the probability that the BBO and GA/GUR results are from the same distribution.

	BBO best	GA best	BBO ave.	GA ave.	BBO σ	GA σ	Prob.
Ackley	3.1	3.0	6.0	7.3	4.5	5.2	3.4e−14
Fletcher	1.5e+4	1.5e+4	1.1e+5	1.6e+5	4.5e+4	5.5e+4	1.3e−6
Griewank	1.6	2.3	7.5	6.7	3.3	4.0	3.1e−13
Penalty #1	1.5	2.3	61	2.8e+2	5.3	11	3.6e−1
Penalty #2	4.6	7.7	2.4e+4	3.1e+4	1.1e+3	2.6e+3	7.6e−3
Quartic	0	0	0.00061	0.0026	0.00002	0.00011	5.5e−2
Rastrigin	0	0.013	6.8	10	2.7	3.8	1.3e−15
Rosenbrock	12	18	1.5e+2	1.9e+2	68	96	1.8e−1
Schwefel 1.2	1.0e+3	1.4e+3	6.4e+3	9.5e+3	3.1e+3	5.3e+3	0
Schwefel 2.21	11	17	38	41	25	31	8.3e−14
Schwefel 2.22	0.22	0.96	3.6	5.8	1.8	2.6	1.2e−9
Schwefel 2.26	1.4e+2	1.6e+2	7.9e+2	8.4e+2	3.6e+2	4.5e+2	4.5 e−11
Sphere	0.0061	0.012	1.3	1.9	0.30	0.63	4.0e−8
Step	82	1.3e+2	4.3e+2	6.8e+2	2.5e+2	3.5e+2	2.6e−7
BBO wins	92%		93%		100%		–

Table 14
Comparison between BBO and GA/GUR performance on 30-dimensional benchmark problems with a population size of 50. The "Prob." column shows the probability that the BBO and GA/GUR results are from the same distribution.

	BBO best	GA best	BBO ave.	GA ave.	BBO σ	GA σ	Prob.
Ackley	4.1	4.7	8.0	9.3	6.4	7.1	2.5e−10
Fletcher	5.9e+4	6.6e+4	3.0e+5	3.4e+5	1.6e+5	1.8e+5	1.5e−8
Griewank	4.5	5.1	16	19	9.2	12	2.8e−7
Penalty #1	8.4	10	8.7e+4	2.2e+5	1.5e+3	4.2e+3	1.5e−1
Penalty #2	2.5e+2	7.2e+2	1.0e+6	1.1e+6	6.3e+4	1.7e+5	9.4e−3
Quartic	7.7e−8	1e−5	0.049	0.37	0.0038	0.031	3.5e−2
Rastrigin	2	5	14	16	7.4	9.7	2.6e−11
Rosenbrock	38	61	3.4e+2	3.7e+2	1.7e+2	2.0e+2	2.4e−4
Schwefel 1.2	5.0e+3	6.3e+3	1.4e+4	2.7e+4	8.9e+3	1.5e+4	0
Schwefel 2.21	24	30	49	58	36	43	6.5e−14
Schwefel 2.22	1.6	3.8	10	13	5.9	7.4	0
Schwefel 2.26	4.4e+2	7.0e+2	1.6e+3	1.8e+3	9.6e+2	1.2e+3	7.5e−10
Sphere	0.41	0.85	4.2	6.6	1.9	2.8	1.6e−7
Step	3.4e+2	6.5e+2	1.5e+3	2.6e+3	9.0e+2	1.2e+3	3.6e−11
BBO wins	100%		100%		100%		–

Lemma 1. *Suppose that $n, N \in \mathbb{N}$ are natural numbers with $n \geqslant 3$ and $N \geqslant 1$. Then*

$$Q(N): \sum_{j=0}^{N-1} \frac{(n+j-3)!}{j!(n-3)!} = \frac{N(n+N-3)!}{N!(n-2)!} \tag{56}$$

Proof. We need the constraints on n and N so that the arguments of the factorials are well-defined. Let $n \in \mathbb{N},\ n \geqslant 3$ be an arbitrary but fixed natural number. We apply the PMI to the proposition $Q(N)$ of (56). It is straightforward to show that $Q(1)$ is true; substituting $N = 1$ in (56) results in the identity $1 = 1$. Next, we write

$$\sum_{j=0}^{N} \frac{(n+j-3)!}{j!(n-3)!} = \sum_{j=0}^{N-1} \frac{(n+j-3)!}{j!(n-3)!} + \frac{(n+N-3)!}{N!(n-3)!} \tag{57}$$

Assuming that $Q(N)$ is true for some $N \geqslant 1$ and applying the inductive hypothesis (56) to the first term on the right side of (57) yields

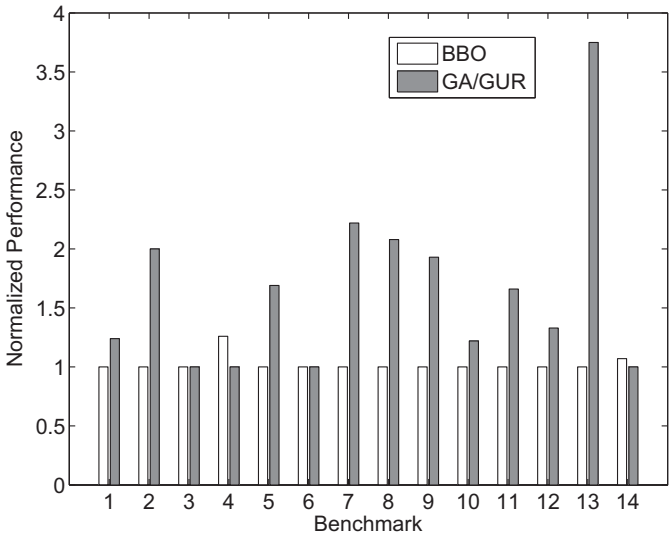

Fig. 9. Comparison between average BBO and GA/GUR performance on 5-dimensional benchmark problems with a population size of 50. Performance numbers are taken from the "Ave." columns of Table 11 and are normalized to the better (smaller) of the results of the two algorithms. The 14 categories along the horizontal axis correspond to the 14 benchmarks in Table 11.

$$\sum_{j=0}^{N}\frac{(n+j-3)!}{j!(n-3)!}=\frac{(n+N-3)!}{(N-1)!(n-2)!}+\frac{(n+N-3)!}{N!(n-3)!}=\frac{N(n+N-3)!}{N!(n-2)!}+\frac{(n-2)(n+N-3)!}{N!(n-2)!}$$
$$=\frac{N(n+N-3)!+(n-2)(n+N-3)!}{N!(n-2)!}=\frac{(n+N-3)!(n+N-2)}{N!(n-2)!}=\frac{(n+N-2)!}{N!(n-2)!}. \quad (58)$$

According to the PMI, $Q(N)$ is true for all $N \geqslant 1$. Since $n \geqslant 3$ is arbitrary, (56) is true for all $n \geqslant 3$ and for all $N \geqslant 1$. □

Appendix B

Tables 11–14 give a detailed breakdown of the four rows of Table 5 and compare BBO and GA/GUR performance for different problem dimensions. The data in each table were generated using a population size of 50, no elitism, and a mutation rate of 1%. The numbers in the "Best" columns show the best BBO and GA/GUR results after 100 Monte Carlo simulations and

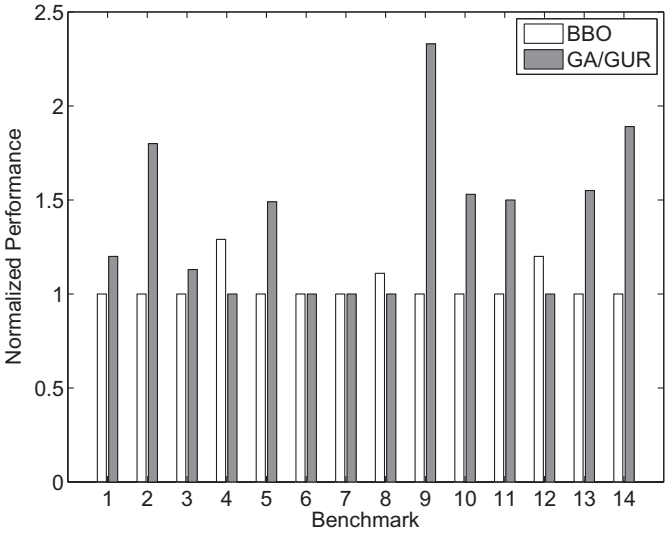

Fig. 10. Comparison between average BBO and GA/GUR performance on 10-dimensional benchmark problems with a population size of 50. Performance numbers are taken from the "Ave." columns of Table 12 and are normalized to the better (smaller) of the results of the two algorithms. The 14 categories along the horizontal axis correspond to the 14 benchmarks in Table 12.

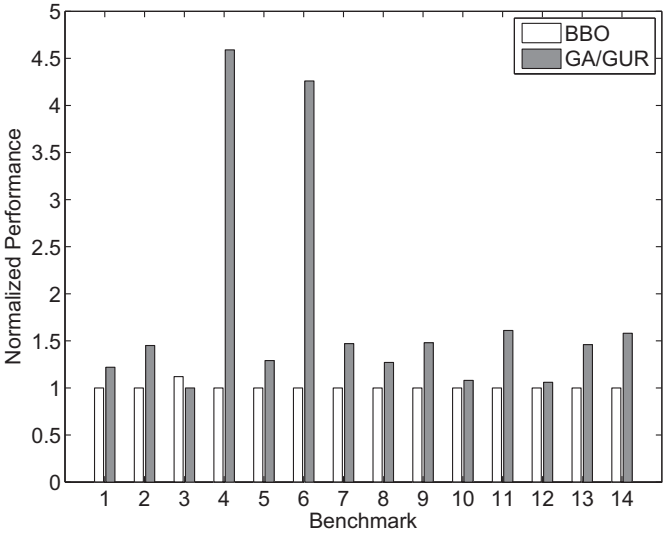

Fig. 11. Comparison between average BBO and GA/GUR performance on 20-dimensional benchmark problems with a population size of 50. Performance numbers are taken from the "Ave." columns of Table 13 and are normalized to the better (smaller) of the results of the two algorithms. The 14 categories along the horizontal axis correspond to the 14 benchmarks in Table 13.

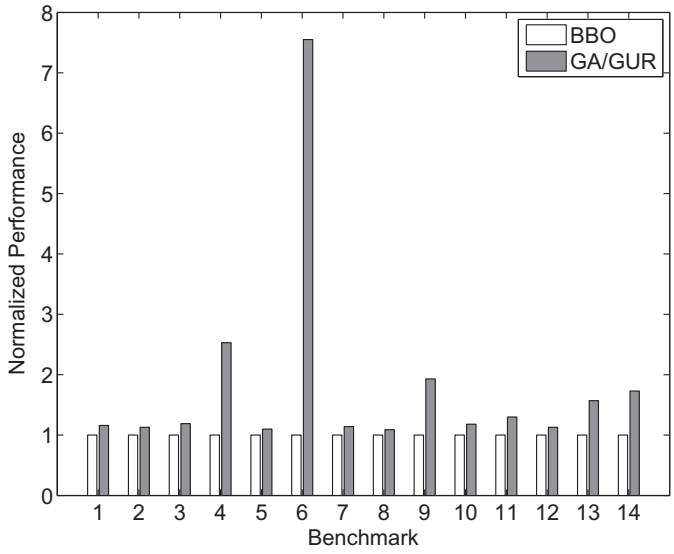

Fig. 12. Comparison between average BBO and GA/GUR performance on 30-dimensional benchmark problems with a population size of 50. Performance numbers are taken from the "Ave." columns of Table 14 and are normalized to the better (smaller) of the results of the two algorithms. The 14 categories along the horizontal axis correspond to the 14 benchmarks in Table 14.

Table 15
Comparison between BBO and GA/GUR performance on 30-dimensional benchmark problems with a population size of 10. The "Prob." column shows the probability that the BBO and GA/GUR results are from the same distribution.

	BBO best	GA best	BBO ave.	GA ave.	BBO σ	GA σ	Prob.
Ackley	9.9	9.7	15	15	12	13	5.3e−8
Fletcher	1.7e+5	1.7e+5	6.4e+5	8.3e+5	3.8e+5	4.6e+5	1.5e−3
Griewank	26	28	94	1.1e+2	57	61	2.8e−4
Penalty #1	3.6e+3	3.5e+4	1.3e+7	3.5e+7	2.3e+6	4.4e+6	5.5e−2
Penalty #2	7.4e+5	7.9e+5	4.0e+7	6.4e+7	9.6e+6	1.6e+7	7.8e−3
Quartic	0.094	0.2	18	28	3.6	5.8	9.0e−7
Rastrigin	21	23	68	82	40	50	1.1e−4
Rosenbrock	2.5e+2	2.4e+2	1.0e+3	1.6e+3	5.6e+2	6.6e+2	2.1e−2
Schwefel 1.2	9.2e+3	1.5e+4	4.7e+4	7.7e+4	1.9e+4	2.6e+4	1.5e−8
Schwefel 2.21	41	48	82	88	66	69	3.3e−3
Schwefel 2.22	18	17	37	39	28	29	2.8e−3
Schwefel 2.26	1.8e+3	2.3e+3	5.2e+3	4.9e+3	3.4e+3	3.7e+3	2.9e−3
Sphere	5.3	8.9	32	34	18	21	1.8–3
Step	2.9e+3	3.1e+3	1.2e+4	1.3e+4	6.4e+3	7.4e+3	1.6e−3
BBO wins	77%		85%		100%		–

Table 16
Comparison between BBO and GA/GUR performance on 30-dimensional benchmark problems with a population size of 20. The "Prob." column shows the probability that the BBO and GA/GUR results are from the same distribution.

	BBO best	GA best	BBO ave.	GA ave.	BBO σ	GA σ	Prob.
Ackley	7.0	7.3	12.0	12.6	9.4	10	2.2e−11
Fletcher	8.2e+4	1.5e+5	4.4e+5	6.0e+5	2.6e+5	3.1e+5	4.2e−5
Griewank	11	17	48	53	26	31	2.2e−6
Penalty #1	19	52	5.0e+6	1.3e+7	2.2e+5	6.3e+5	6.7e−3
Penalty #2	1.1e+5	3.6e+3	9.6e+6	1.6e+7	1.8e+6	2.7e+6	3.0e−3
Quartic	0.0021	0.0032	4.6	8.1	0.46	0.87	4.0e−3
Rastrigin	10	11	36	37	21	25	6.2e−7
Rosenbrock	1.3e+2	93	6.8e+2	6.2e+2	3.2e+2	3.6e+2	2.8e−6
Schwefel 1.2	6.9e+3	1.2e+4	2.3e+4	3.5e+4	1.4e+4	2.0e+4	8.1e−12
Schwefel 2.21	36	45	69	78	53	59	3.2e−13
Schwefel 2.22	7.9	9.4	24	30	16	18	3.1e−7
Schwefel 2.26	1.3e+3	1.2e+3	3.0e+3	3.3e+3	2.1e+3	2.4e+3	1.9e−6
Sphere	3.4	5.3	18	19	8.1	11	1.6e−4
Step	1.1e+3	2.0e+3	5.0e+3	7.7e+3	2.8e+3	3.8e+3	6.6e−6
BBO wins	79%		93%		100%		–

indicate which algorithm can find the best solution over multiple runs. The numbers in the "Ave." columns show the average performance of 100 Monte Carlo simulations and indicate which algorithm performs best on average. The numbers in the "σ" columns show the standard deviation of the 100 Monte Carlo results and indicate which algorithm is the most robust and consistent from one run to the next. The numbers in the "Prob." columns are from the results of a t-test, and indicate the probability that the BBO and GA/GUR results are from the same probability distribution. We obtained extremely low numbers in the "Prob." columns because we ran so many simulations (100), which gives us a high confidence that the difference between BBO and GA/GUR is statistically significant. Figs. 9–12 show the average BBO and GA/GUR results, normalized to the lowest of the two values for each benchmark.

Appendix C

Tables 15 and 16 give a detailed breakdown of the first two rows of Table 6 and compare BBO and GA/GUR performance for different population sizes. The data in each table were generated using 30-dimensional benchmark problems, no elitism,

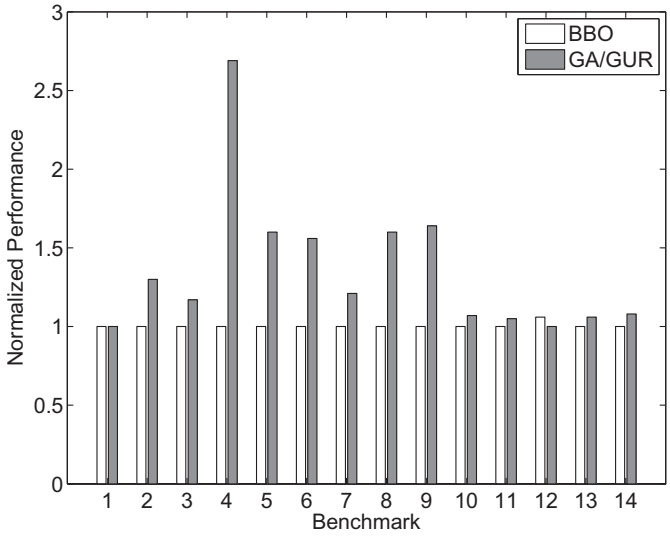

Fig. 13. Comparison between average BBO and GA/GUR performance on 30-dimensional benchmark problems with a population size of 10. Performance numbers are taken from the "Ave." columns of Table 15 and are normalized to the better (smaller) of the results of the two algorithms. The 14 categories along the horizontal axis correspond to the 14 benchmarks in Table 15.

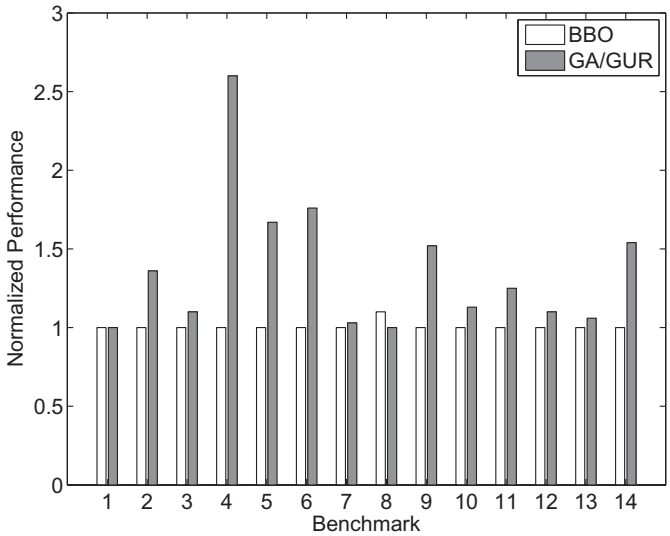

Fig. 14. Comparison between average BBO and GA/GUR performance on 30-dimensional benchmark problems with a population size of 20. Performance numbers are taken from the "Ave." columns of Table 16 and are normalized to the better (smaller) of the results of the two algorithms. The 14 categories along the horizontal axis correspond to the 14 benchmarks in Table 16.

and a mutation rate of 1%. Note that the third row of Table 6 represents the same data as the fourth row of Table 5, and so Table 14 in Appendix 2 gives a detailed breakdown of the third row of Table 6. Figs. 13 and 14 show the average BBO and GA/GUR results, normalized to the lowest of the two values for each benchmark.

References

[1] D. Ackley, A Connectionist Machine for Genetic Hillclimbing, Kluwer Academic Publishers, 1987.
[2] T. Back, Evolutionary Algorithms in Theory and Practice, Oxford University Press, 1996.
[3] T. Back, U. Hammel, H. Schwefel, Evolutionary computation: comments on the history and current state, IEEE Transactions on Evolutionary Computation (1) (1997) 3–17.
[4] N. Beaulieu, On the generalized multinomial distribution, optimal multinomial detectors, and generalized weighted partical decision detectors, IEEE Transactions on Communications (39) (1991) 193–194.
[5] M. Bhattacharya, Expensive optimization, uncertain environment: an EA-based solution, in: Genetic and Evolutionary Computation Conference, 2007, pp. 2407–2414.
[6] H. Bremermann, M. Rogson, S. Salaff, Global properties of evolution processes, in: H. Pattee, E. Edlsack, L. Fein, A. Callahan (Eds.), Natural Automata and Useful Simulations, Spartan Books, 1966, pp. 3–41.
[7] Z. Cai, Y. Wang, A multiobjective optimization-baed evoluationary algorithm for constrained optimization, IEEE Transactions on Evolutionary Computation (10) (2006) 658–675.
[8] M. Clerc, J. Kennedy, The particle swarm – Explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation (6) (2002) 58–73.
[9] T. Davis, J. Principe, A simulated annealing like convergence theory for the simple genetic algorithms, in: R. Belew, L. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kauffman, 1991, pp. 174–181.
[10] T. Davis, J. Principe, A Markov chain framework for the simple genetic algorithm, Evolutionary Computation (1) (1993) 269–288.
[11] W. Dembski, R. Marks, Conservation of information in search: measuring the cost of success, IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans (39) (2009) 1051–1061.
[12] A. Eiben, Multiparent recombination in evolutionary computing, in: A. Ghosh, S. Tsutsui (Eds.), Advances in Evolutionary Computing, Springer-Verlag, 2003, pp. 175–192.
[13] A. Eiben, Multiparent recombination, in: T. Back, D. Fogel, Z. Michalewicz (Eds.), Evolutionary Computation 1: Basic Algorithms and Operators, Institute of Physics Publishing, 2000, pp. 289–302.
[14] A. Eiben, C. Schippers, Multi-parent's niche: *n*-ary crossovers on NK-landscapes, in: Proceedings of the 4th Conference on Parallel Problem Solving from Nature, 1996, pp. 319–328.
[15] A. Eiben, T. Back, Empirical investigation of multiparent recombination operators in evolution strategies, Evolutionary Computation (5) (1998) 347–365.
[16] M. Ergezer, D. Simon, D. Du, Oppositional biogeography-based optimization, in: IEEE Conference on Systems, Man, and Cybernetics, San Antonio, Texas, 2009, pp. 1035–1040.
[17] A. Falkenauer, A hybrid grouping genetic algorithm, Journal of Heuristics (2) (1996) 5–30.
[18] W. Gong, Z. Cai, C. Ling, DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization, Soft Computing, in press. doi:10.1007/s00500-010-0591-1.
[19] C. Grinstead, J. Snell, Introduction to Probability, American Mathematical Society, 1997.
[20] Y. Ho, D. Pepyne, Simple explanation of the no-free-lunch theorem and its implications, Journal of Optimization Theory and Applications (155) (2002) 549570.
[21] F. Hoffmeister, T. Back, Genetic algorithms and evolution strategies: Similarities and differences, in: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature, 1991, pp. 455–469.
[22] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Computing (9) (2005) 3–12.
[23] A. Khanafer, One dimensional bin packing. <http://ali.khanafer.free.fr/index.php?n=Main.Benchmarks>.
[24] H. Kundra, A. Kaur, V. Panchal, An integrated approach to biogeography based optimization with case based reasoning for retrieving groundwater possibility, in: 8th Annual Asian Conference and Exhibition on Geospatial Information, Technology and Applications, August 2009.
[25] F. Leighton, A graph coloring algorithm for large scheduling problems, Journal of Research of the National Bureau of Standards (84) (1979) 489–505.
[26] K. Loh, B. Golden, E. Wasil, Solving the one-dimensional bin packing problem with a weight annealing heuristic, Computers and Operations Research (35) (2008) 2283–2291.
[27] M. Lomolino, B. Riddle, J. Brown, Biogeography, Sinauer Associates, 2009.
[28] M. Lundy, A. Mees, Convergence of an annealing algorithm, Mathematical Programming (34) (1986) 111–124.
[29] R. Lung, D. Dumitrescu, A new evolutionary model for detecting multiple optima, in: Genetic and Evolutionary Computation Conference, 2007, pp. 1296–1303.
[30] H. Ma, S. Ni, M. Sun, Equilibrium species counts and migration model tradeoffs for biogeography-based optimization, in: IEEE Conference on Decision and Control, 2009, pp. 3306–3310.
[31] D. Matula, G. Marble, J. Isaacson, Graph coloring algorithms, in: R. Read (Ed.), Graph Theory and Computing, Academic Press, 1972, pp. 109–122.
[32] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, 1998.
[33] H. Mo, L. Xu, Biogeography migration algorithm for traveling salesman problem, in: Y. Tan, Y. Shi, K.-C. Tan (Eds.), Advances in Swarm Intelligence, Springer, 2010, pp. 405–414.
[34] K. Moreland, The needles-in-haystack problem, in: International Conference on Machine Learning and Data Mining in Pattern Recognition, 2009, pp. 516–524.
[35] A. Nix, M. Vose, Modeling genetic algorithms with Markov chains, Annals of Mathematics and Artificial Intelligence (5) (1992) 79–88.
[36] V. Panchal, P. Singh, N. Kaur, H. Kundra, Biogeography based satellite image classification, International Journal of Computer Science and Information Security (6) (2009) 269–274.
[37] K. Price, R. Storn, Differential evolution, Dr. Dobb's Journal (22) (1997) 18–20. 22, 24, 78.
[38] R. Rarick, D. Simon, F. Villaseca, B. Vyakaranam, Biogeography-based optimization and the solution of the power flow problem, in: IEEE Conference on Systems, Man, and Cybernetics, 2009, pp. 1029–1034.
[39] A. Ratle, Accelerating the convergence of evolutionary algorithms by fitness landscape approximation, in: A. Eiben, T. Back, M. Schoenauer, H. Schwefel (Eds.), Parallel Problem Solving from Nature – PPSN, vol. V, Springer, 1998, pp. 87–96.
[40] G. Reinelt, TSPLIB, August 2008. <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95>.
[41] C. Reeves, J. Rowe, Genetic Algorithms: Principles and Perspectives, Kluwer, 2003.
[42] P. Roy, S. Ghoshal, S. Thakur, Biogeography-based optimization for economic load dispatch problems, Electric Power Components and Systems (38) (2010) 166–181.
[43] V. Savsani, R. Rao, D. Vakharia, Discrete optimisation of a gear train using biogeography based optimisation technique, International Journal of Design Engineering (2) (2009) 205–223.
[44] A. Scholl, R. Klein, Bin packing, September 1, 2003. <http://www.wiwi.uni-jena.de/Entscheidung/binpp/bin3dat.htm>.
[45] D. Simon, Biogeography-based optimization, IEEE Transactions on Evolutionary Computation (12) (2008) 702–713.

[46] D. Simon, M. Ergezer, D. Du, Population distributions in biogeography-based optimization algorithms with elitism, in: IEEE Conference on Systems, Man, and Cybernetics, 2009, pp. 1017–1022.
[47] D. Simon, A probabilistic analysis of a simplified biogeography-based optimization algorithm, Evolutionary Computation, in press. Available at <http://academic.csuohio.edu/simond/bbo/simplified>.
[48] D. Simon, M. Ergezer, D. Du, and R. Rarick, Markov Models for Biogeography-Based Optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part B. Available at <http://embeddedlab.csuohio.edu/BBO>.
[49] R. Storn, System design by constraint adaptation and differential evolution, IEEE Transactions on Evolutionary Computation (3) (1999) 22–34.
[50] J. Suzuki, A Markov chain analysis on simple genetic algorithms, IEEE Transactions on Systems, Man and Cybernetics, Part B (25) (1995) 655–659.
[51] 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 (28) (1998) 95–102.
[52] G. Tao, Z. Michalewicz, Inver-over operator for the TSP, in: A. Eiben, T. Back, M. Schoenauer, H. Schwefel (Eds.), Parallel Problem Solving from Nature – PPSN, vol. V, Springer, 1998, pp. 803–812.
[53] R. Torregosa, W. Kanok-Nukulchai, Weight optimization of steel frames using genetic algorithm, Advances in Structural Engineering (5) (2002) 99–111.
[54] M. Trick, Graph coloring instances. <http://mat.gsia.cmu.edu/COLOR/instances.html>.
[55] F. Vavak, T. Fogarty, Comparison of steady state and generational genetic algorithms for use in nonstationary environments, in: IEEE International Conference on Evolutionary Computation, 1996, pp. 192–195.
[56] S. Venkatraman, G. Yen, A simple elitist genetic algorithm for constrained optimization, in: IEEE Congress on Evolutionary Computation, 2004, pp. 288–295.
[57] R. Whittaker, Island Biogeography, Oxford University Press, 1998.
[58] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation (3) (1999) 82–102.