Analytical and Numerical Comparisons of Biogeography-Based Optimization and Genetic Algorithms

Dan Simon,* Rick Rarick, Mehmet Ergezer, and Dawei Du
Cleveland State University
Department of Electrical and Computer Engineering
Stilwell Hall Room 332
2121 Euclid Avenue
Cleveland, Ohio 44115

March 26, 2010

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 benchmark simulations. We show that the unique selection pressure provided by BBO generally results in better optimization results for a set of standard benchmark problems.

Key Words – evolutionary algorithm, Markov model, biogeography-based optimization, genetic algorithm, global uniform recombination, benchmark.

* Corresponding author, email address d.j.simon@csuohio.edu. This work was supported by NSF Grant 0826124 in the CMMI Division of the Engineering Directorate.
1 Introduction

Mathematical models of biogeography describe the migration of species between islands, along with their speciation and extinction [22, 46]. Biogeography-based optimization (BBO) was first presented in [36] 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 [36], power system optimization [31, 34], groundwater detection [21], mechanical gear train design [35], and satellite image classification [29].

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 [25], we use linear migration curves like those shown in Figure 1 for the sake of simplicity. Figure 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 [36], which is called partial immigration-based BBO in [38]. 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 probabilistically selected based on fitness (e.g., using roulette wheel selection). This gives the algorithm shown in Figure 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
Figure 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.

BBO algorithm shown in Figure 2. Borrowing from GA terminology [44], we therefore say that Figure 2 depicts a generational BBO algorithm.

This paper is based on a previously-developed Markov model for BBO [39]. Markov models have been developed for other EAs also, including simple genetic algorithms [41, 42] and simulated annealing [23].

A Markov chain is a random process which has a set of $T$ possible states [17, 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

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

where each row $p_{ss}$ of $P(\infty)$ is the same. The $i$th 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.
For each solution $y_k$, $k \in [1, N]$, 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 $\lambda_k$ 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

$y \leftarrow z$

---

Figure 2: One generation of the BBO algorithm. $y$ is the entire population of solutions, $y_k$ is the $k$th solution, and $y_k(s)$ is the $s$th feature of $y_k$. Similarly, $z$ is the temporary population of solutions, $z_k$ is the $k$th temporary solution, and $z_k(s)$ is the $s$th feature of $z_k$.

A Markov state in this paper represents a BBO population distribution. Each state describes how many individuals of each point in the search-space there are in the population. The probability $P_{ij}$ is the probability that the population transitions from the $i$th distribution to the $j$th distribution from one generation to the next. If the mutation rate is nonzero, this probability is 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 [39] 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 benchmark simulations. We provide concluding remarks and directions for future work in Section 5.
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 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 Figure 3. Comparing Figures 2 and 3, it can be seen that BBO reduces to a specific type of GA/GUR if, instead of setting $\lambda_k = 1 - \mu_k$ in the BBO algorithm of Figure 2, we set $\lambda_k = 1$ for all $k$.

---

For each solution $y_k$, $k \in [1, N]$, define parent probability $\mu_k \propto \text{fitness of } y_k$, $\mu_k \in [0, 1]$ 
$z \leftarrow y$
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
$y \leftarrow z$

---

Figure 3: One generation of a GA with global uniform recombination (GA/GUR). $y$ is the entire population of solutions, $y_k$ is the $k$th solution, and $y_k(s)$ is the $s$th feature of $y_k$. Similarly, $z$ is the temporary population of solutions, $z_k$ is the $k$th temporary solution, and $z_k(s)$ is the $s$th feature of $z_k$. Compare with Figure 2.
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) [40, 30]. 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 [19] 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 \( \phi_1 \) 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) [26]. If a \((\mu, \lambda)\) 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 \( \mu \) and \( \lambda \) that are used in ES notation are not related to the \( \mu \) and \( \lambda \) 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 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 [25], species populations (including mortality and reproduction), predator/prey relationships, species mobilities, directional momentum during migration, habitat area and isolation, and many others [22, 46].

3 Markov Models for BBO and GA/GUR

In [39] 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 [39] 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 Figure 2. Second, a solution can emigrate a feature to itself. Third, the migration rates $\lambda$ and $\mu$ are independent of the population distribution; that is, absolute fitness values are used to obtain $\lambda$ and $\mu$, as opposed to a rank-based fitness [45].

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 $v$ to denote the population vector, where the component $v_i$ is the number of $x_i$ individuals in the population. We see that

$$\sum_{i=1}^{n} v_i = N. \quad (2)$$

We use $y_k$ to denote the $k$th individual in the population, where the $y_k$’s are ordered to
group identical individuals. The population of the search algorithm can thus be depicted as

$$\text{Population} = \{y_1, \ldots, y_N\} = \{x_1, x_1, \ldots, x_1, x_2, x_2, \ldots, x_n, x_n, \ldots, x_n\}$$

(3)

where the $y_k$’s are put in the same order as the $x_i$’s. We use $\mu_i$ to denote the emigration probability of $x_i$, which is proportional to the fitness of $x_i$. We use $\lambda_i$ to denote the immigration probability of $x_i$, which decreases with the fitness of $x_i$. We use the notation $x_i(s)$ to denote the $s$th bit of solution $x_i$. For $i \in [1, n]$ and $s \in [1, q]$ we use the notation $\mathcal{J}_i(s)$ to denote the set of search space indices $j$ such that $x_j(s) = x_i(s)$:

$$\mathcal{J}_i(s) = \{j \in [1, n] : x_j(s) = x_i(s)\}.$$  

(4)

Note that the cardinality of $\mathcal{J}_i(s)$ is $n/2$ for all $i$ and $s$. See [39] for an example that shows how $\mathcal{J}_i(s)$ is constructed. From (3) we see that

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

(5)

This can be written more compactly as

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

(6)

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

$$m(k) = \min r \text{ such that } \sum_{i=1}^{r} v_i \geq k.$$  

(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 $s$th bit of the $k$th individual at generation $t$.

Let $\text{im}(t)$ be the event that immigration occurs at the $t$-th 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 [39] that for fixed $i$, $k$, $s$, and $t$,
\[
\Pr(y_k(s)_{t+1} = x_i(s)) = \Pr(\overline{\text{im}}(t)) \Pr(y_k(s)_{t+1} = x_i(s) | \overline{\text{im}}(t)) + \\
\Pr(\text{im}(t)) \Pr(y_k(s)_{t+1} = x_i(s) | \text{im}(t))
\]

\[
= \begin{cases} 
(1 - \lambda_{m(k)}) + \lambda_{m(k)} \frac{\sum_{j \in J_i(s)} v_j \mu_j}{\sum_{j=1}^{n} v_j \mu_j}, & \text{if } x_{m(k)}(s) = x_i(s) \\
\lambda_{m(k)} \frac{\sum_{j \in J_i(s)} v_j \mu_j}{\sum_{j=1}^{n} v_j \mu_j}, & \text{if } x_{m(k)}(s) \neq x_i(s) 
\end{cases}
\]

\[
= (1 - \lambda_{m(k)}) \mathbf{1}_0(x_{m(k)}(s) - x_i(s)) + \lambda_{m(k)} \frac{\sum_{j \in J_i(s)} v_j \mu_j}{\sum_{j=1}^{n} v_j \mu_j}
\]

\[
= (1 - \lambda_{m(k)}) \mathbf{1}_0(x_{m(k)}(s) - x_i(s)) + \lambda_{m(k)} f_i(s)
\]

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 \( s \)th 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 \( \mu_k \) in Figure 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 \( i \),

\[
\Pr(y_{k,1,t+1} = x_i(1), \ldots, y_{k,q,t+1} = x_i(q)) = \prod_{s=1}^{q} \Pr(y_k(s)_{t+1} = x_i(s)).
\]

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

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

\( P_{ki}(v) \) can be computed for each \( k \in [1,N] \) and each \( i \in [1,n] \) to form the \( N \times n \) matrix \( P(v) \). The \( k \)th row of \( P(v) \) corresponds to the \( k \)th iteration of the outer loop in Figure 2. The \( i \)th 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 Figure 2), where the probability of the $i$th outcome on the $k$th trial 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 $u = [ u_1 \cdots u_n ]^T$. Then the probability that we obtain population vector $u$ at the $(t + 1)$st generation, given that we have population vector $v$ at the $t$-th generation, can be calculated with the generalized multinomial theorem [39, 4] as

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

(11)

where

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

An example of how to construct $Y(u)$ is given in [39].

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

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

(13)

from which we obtain

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

(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

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

(15)

Equation 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 [33] 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 [28] the value of $T$ is expressed using the notation for combinations:

$$T = C(n + N - 1, N).$$

(16)

Other methods for calculating $T$ are discussed in [39].

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 $\lambda_i$ is replaced with 1. In BBO, the immigration rate $\lambda_i$ decreases with the fitness of $x_i$. In GA/GUR, $\lambda_i = 1$ for all $i$. 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 [33, 28, 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 3,876 possible population vectors as calculated from (16).

We investigate 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 [37]. The crossover probability used in GA/SP was 0.9.

<table>
<thead>
<tr>
<th>Mutation Rate</th>
<th>Population Vector</th>
<th>GA/SP</th>
<th>GA/GUR</th>
<th>BBO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>All Optimal</td>
<td>0.0084</td>
<td>0.0079</td>
<td>0.0044</td>
</tr>
<tr>
<td></td>
<td>No Optima</td>
<td>0.5826</td>
<td>0.5623</td>
<td>0.5111</td>
</tr>
<tr>
<td>0.01</td>
<td>All Optimal</td>
<td>0.2492</td>
<td>0.2513</td>
<td>0.3484</td>
</tr>
<tr>
<td></td>
<td>No Optima</td>
<td>0.5436</td>
<td>0.5372</td>
<td>0.2128</td>
</tr>
<tr>
<td>0.001</td>
<td>All Optimal</td>
<td>0.4029</td>
<td>0.4034</td>
<td>0.7616</td>
</tr>
<tr>
<td></td>
<td>No Optima</td>
<td>0.5696</td>
<td>0.5690</td>
<td>0.1679</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Mutation Rate</th>
<th>Population Vector</th>
<th>GA/SP</th>
<th>GA/GUR</th>
<th>BBO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>All Optimal</td>
<td>0.0119</td>
<td>0.0106</td>
<td>0.0066</td>
</tr>
<tr>
<td></td>
<td>No Optima</td>
<td>0.5006</td>
<td>0.4939</td>
<td>0.4370</td>
</tr>
<tr>
<td>0.01</td>
<td>All Optimal</td>
<td>0.3675</td>
<td>0.3701</td>
<td>0.4715</td>
</tr>
<tr>
<td></td>
<td>No Optima</td>
<td>0.4139</td>
<td>0.4079</td>
<td>0.1450</td>
</tr>
<tr>
<td>0.001</td>
<td>All Optimal</td>
<td>0.5655</td>
<td>0.5670</td>
<td>0.8502</td>
</tr>
<tr>
<td></td>
<td>No Optima</td>
<td>0.4069</td>
<td>0.4053</td>
<td>0.0968</td>
</tr>
</tbody>
</table>

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.
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 individuals (85% to 56%), and in its lower probability of obtaining a population with no optimal individuals (10% to 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 [25], 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 the simple problem

\[
\begin{align*}
\mu &= \{2/3, 1/3, \ldots, 1/3\} \\
\lambda &= \{1/3, 2/3, \ldots, 2/3\}. 
\end{align*}
\]

That is, assuming the linear migration rates of Figure 1, \( x_1 \) is twice as fit as \( x_i \) for \( i \in [2, n] \) 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
\[
v_1 = 0, \quad m(k) \neq 1, \quad \lambda_{m(k)} = 2/3, \quad \mu_{m(k)} = 1/3, \quad \text{for all } k \in [1, N]. \tag{18}
\]
Therefore,
\[
\sum_{j \in J_1(s)} v_j \mu_j = \frac{1}{3} \sum_{j \in J_1(s)} v_j. \tag{19}
\]
The search space \( \{x_i\} \) consists of \( n \) possible solutions. We have \( N \) random variables \( y_k \), each having the identical probability mass function
\[
\Pr(y_k = x_i) = \begin{cases} 0, & \text{if } i = 1 \\ 1/(n - 1), & \text{if } i \in [2, n]. \end{cases} \tag{20}
\]
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
\[
X = 1_0(x_{m(k)}(s) - x_1(s)) \in \{0, 1\}. \quad \text{We will continue to write } m(k) \text{ instead of } m(k, v) \text{ for brevity of notation. Recall from (4) that the cardinality of } J_1(s) \text{ is } n/2 \text{ for all } i \text{ and } s, \text{ so there are } n/2 \text{ search space indices } m(k) \text{ for which } x_{m(k)}(s) = x_1(s), \text{ or equivalently, for which } X = 1. \text{ However, since there are no } x_1 \text{ individuals in the population, there are only } (n/2 - 1) \text{ unique population indices } k \text{ such that } X = 1. \text{ This fact, along with the fact from (20) that there are } (n - 1) \text{ search space indices } i \text{ such that } y_k = x_i, \text{ gives}
\]
\[
E[X] = E \left[ 1_0(x_{m(k)}(s) - x_1(s)) \right] = (0)\Pr[X = 0] + (1)\Pr[X = 1]
= \frac{n/2 - 1}{n - 1}. \tag{21}
\]
A formal proof of (21) is in Appendix 1. An argument similar to that given above can be used to show that
\[
E \left[ 1_0(x_{m(k)}(s) - x_1(s))1_0(x_{m(k)}(r) - x_1(r)) \right] = \left( \frac{n/2 - 1}{n - 1} \right)^2
\tag{22}
\]
for all \( r \neq s \). Therefore, \( 1_0(x_{m(k)}(s) - x_1(s)) \) and \( 1_0(x_{m(k)}(r) - x_1(r)) \), 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 \( \sum_{j \in J_1(s)} v_j \) with respect to the random variable \( v \). For the sake of this calculation, we temporarily re-index the components of \( v \) so that
\[
\sum_{j \in J_1(s)} v_j = v_1 + v_2 + \cdots + v_{n/2}. \tag{23}
\]
Recall from (18) that \( v_1 = 0 \). Recall from (20) 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

\[
\{ y_1, \ldots, y_N \} = \{ x_2, x_2, \ldots, x_2, \underline{x_3, x_3, \ldots, x_3}, \ldots, x_n, x_n, \ldots, x_n \},
\]

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

\[
E \left[ \sum_{j \in \mathcal{J}_1(s)} v_j \right] = E[v_1] + E[v_2] + \cdots + E[v_{n/2}]
\]

\[
= 0 + \frac{N}{n - 1} + \cdots + \frac{N}{n - 1}
\]

\[
= \frac{N(n/2 - 1)}{n - 1}.
\]

(25)

(26)

A formal proof of (26) can be constructed along the lines of the proof of (21). An argument similar to that given above can be used to show that

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

(27)

for all \( r \neq s \). Therefore, \( \sum_{j \in \mathcal{J}_1(s)} v_j \) and \( \sum_{j \in \mathcal{J}_1(r)} v_j \), 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 \( \sum_{j \in \mathcal{J}_1(s)} v_j \) and \( \sum_{j \in \mathcal{J}_1(r)} v_j \) 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
\[
E[\Pr(y_{k,t+1} = x_1)] = \prod_{s=1}^{q} \left[ (1 - \lambda_{m(k)})E[\mathbf{1}_0(x_{m(k)}(s) - x_1(s))] + \lambda_{m(k)} \frac{E\left[ \sum_{j \in \mathcal{J}_1(s)} v_j \mu_j \right]}{\sum_{j=2}^{n} v_j \mu_j} \right] \\
= \prod_{s=1}^{q} \left[ \left( \frac{1}{3} \right) \left( \frac{n/2 - 1}{n - 1} \right) + \left( \frac{2}{3} \right) \left( \frac{1}{3} E\left[ \sum_{j \in \mathcal{J}_1(s)} v_j \right] \right) \right] \\
= \prod_{s=1}^{q} \left[ \left( \frac{1}{3} \right) \left( \frac{n/2 - 1}{n - 1} \right) + \left( \frac{2}{3} \right) \left( \frac{1}{3} \frac{N(n-2)}{2(n-1)} \right) \right] \\
= \left( \frac{n-2}{2(n-1)} \right)^q \approx 2^{-q} = 1/n \text{ for large } n. \tag{28}
\]

The above result shows that the probability of finding the optimal solution, averaged over all population distributions given by (20) and all \( k \in [1, N] \), 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 (28) 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 (28) 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 \( \mu_k \) values shown in (17), but \( \lambda_{m(k)} = 1 \) for all \( k \). In this case (8) gives

\[
E[\Pr(y_{k,t+1} = x_1)] = \prod_{s=1}^{q} \left[ (1 - \lambda_{m(k)})E[\mathbf{1}_0(x_{m(k)}(s) - x_1(s))] + \lambda_{m(k)} \frac{E\left[ \sum_{j \in \mathcal{J}_1(s)} v_j \mu_j \right]}{\sum_{j=2}^{n} v_j \mu_j} \right] \\
= \prod_{s=1}^{q} \left( \frac{1}{3} \frac{N(n-2)}{2(n-1)} \cdot \frac{N/3}{N/3} \right) \\
= \left( \frac{n-2}{2(n-1)} \right)^q \approx 2^{-q} = 1/n \text{ for large } n. \tag{29}
\]
Comparing (28) and (29), 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 simple problem

\[ \mu = \{2/3, 1/3, \ldots, 1/3\} \]
\[ \lambda = \{1/3, 2/3, \ldots, 2/3\}. \]  

(30)

That is, assuming the linear migration rates of Figure 1, \(x_1\) is twice as fit as \(x_i\) for \(i \in [2, n]\) 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

\[ v_1 = 1, \ m(k_0) = 1, \ \lambda_{m(k_0)} = 1/3, \ \mu_{m(k_0)} = 2/3. \]  

(31)

Therefore,

\[
\sum_{j \in J_1(s)} v_j \mu_j = \sum_{j=1}^{1} v_j \mu_j + \sum_{\substack{j \in J_1(s) \ j \neq 1}} v_j \mu_j
= 2/3 + 1/3 \sum_{\substack{j \in J_1(s) \ j \neq 1}} v_j. \]  

(32)

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

\[ \Pr(y_k = x_i) = \begin{cases} 
0, & \text{if } i = 0 \\
1/(n - 1), & \text{if } i \in [2, n]. 
\end{cases} \]  

(33)

Given this random population distribution, we calculate the expected value of the sum on the right side of (32) with respect to the random variable \(v\). Recall from (4) that the cardinality of \(J_i(s)\) is \(n/2\) for all \(i\) and \(s\). However, since the sum on the right side of (32) 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

\[
E \left[ \sum_{j \in \mathcal{J}_1(s)} v_j \right] = (N - 1) \Pr(y_k(s) = x_1(s) : k \neq k_0)
\]

\[
= \frac{(N - 1)(n/2 - 1)}{n - 1}.
\]

(34)

As with (21) and (26), a formal derivation similar to that in Appendix 1 can also be used to obtain (34).

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 \( v \):

\[
E \left[ \Pr(y_{k_0,t+1} = x_1) \right] = \prod_{s=1}^{q} \left[(1 - \lambda_{m(k_0)})E \left[ \text{1}_0(x_{m(k_0)}(s) - x_1(s)) \right] + \lambda_{m(k_0)} \frac{E \left[ \sum_{j \in \mathcal{J}_1(s)} v_j \mu_j \right]}{\sum_{j=1}^{n} v_j \mu_j} \right]
\]

\[
= \prod_{s=1}^{q} \left[ \left( \frac{2}{3} \right)^2 \left( 1 + \frac{1}{3} \right) \left( \frac{2}{3} + \frac{(N-1)(n-2)}{6(n-1)} \right) \right]
\]

\[
= \left( \frac{5N + 7}{6(n - 1)(N + 1)} \right)^q
\]

\[
\approx \left( \frac{5N + 7}{6N + 6} \right)^q \text{ for large } n
\]

\[
\approx \left( \frac{5}{6} \right)^q \text{ for large } N.
\]

(35)

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 (35) with (28)).

We also note from (35) 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 \( \mu \) 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 $\mu$ values shown in (30), but $\lambda_{m(k)} = 1$. In this case we obtain

$$E[\Pr(y_{k,t+1} = x_1)] = \prod_{s=1}^{q} \left[ (1 - \lambda_{m(k)}) E[1_0(x_{m(k)}(s) - x_1(s))] + \lambda_{m(k)} \frac{E\left[\sum_{j \in \mathcal{A}(s)} v_j \mu_j\right]}{\sum_{j=1}^{n} v_j \mu_j} \right]$$

$$= \prod_{s=1}^{q} \left( \frac{4(n-1) + (N-1)(n-2)}{2(n-1)(N+1)} \right)$$

$$= \left( \frac{(N+3)n - 2(N+1)}{2(n-1)(N+1)} \right)^q$$

$$\approx \left( \frac{N + 3}{2N + 2} \right)^q$$ for large $n$

$$\approx \left( \frac{1}{2} \right)^q$$ for large $N$.  \hspace{1cm} (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. Furthermore, the rate of decrease is more severe than that of BBO (compare with (35)). Similar to the discussion following (35), we 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 $(1/2)^q$ 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 [27]. We assumed that the population was uniformly distributed throughout the search space, which
is a reasonable assumption given our assumed fitness function. Although our analysis was conducted under simple conditions, our simulation results in Section 4.4 will show that it provides an accurate 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 [32] and methods for fitness function approximation [20]. 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 [43]. Even with elitist EAs, it is often desirable to find multiple optima, or to find multiple solutions near the global optimum [24], 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

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

\[
E[\Pr(\text{retaining an optimum})] \approx \begin{cases} 
(5/6)^q & \text{BBO} \\
(1/2)^q & \text{GA/GUR}.
\end{cases}
\] (37)

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. Furthermore, (37) 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. Equations (35) and (36)
are repeated here with the large \( n \) approximation, but without the large \( N \) approximation:

\[
\text{BBO: } E[\text{Pr}(\text{retaining an optimum})] \approx \left( \frac{5N + 7}{6(N + 1)} \right)^q \\
\text{GA/GUR: } E[\text{Pr}(\text{retaining an optimum})] \approx \left( \frac{N + 3}{2(N + 1)} \right)^q. \tag{38}
\]

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

\[
\frac{\text{BBO performance}}{\text{GA/GUR performance}} = \left( \frac{5N + 7}{3N + 9} \right)^q. \tag{39}
\]

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 \( N = 1 \), (39) evaluates to 1; that is, BBO and GA/GUR performance are equal. As \( N \) increases, (39) 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, 47, 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 minimum 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 2 gives a more detailed view of the data shown in Table 5.

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 minimum 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 3 gives a more detailed view of the data shown in Table 6.

<table>
<thead>
<tr>
<th>Function</th>
<th>Multimodal?</th>
<th>Separable?</th>
<th>Regular?</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ackley</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>$[-30]^n$</td>
</tr>
<tr>
<td>Fletcher-Powell</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>$[-\pi]^n$</td>
</tr>
<tr>
<td>Griewank</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>$[-600]^n$</td>
</tr>
<tr>
<td>Penalty #1</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>$[-50]^n$</td>
</tr>
<tr>
<td>Penalty #2</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>$[-50]^n$</td>
</tr>
<tr>
<td>Quartic</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>$[-1.28]^n$</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>$[-5.12]^n$</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>$[-2.048]^n$</td>
</tr>
<tr>
<td>Schwefel 1.2</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>$[-65.536]^n$</td>
</tr>
<tr>
<td>Schwefel 2.21</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>$[-100]^n$</td>
</tr>
<tr>
<td>Schwefel 2.22</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>$[-10]^n$</td>
</tr>
<tr>
<td>Schwefel 2.26</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>$[-512]^n$</td>
</tr>
<tr>
<td>Sphere</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>$[-5.12]^n$</td>
</tr>
<tr>
<td>Step</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>$[-200]^n$</td>
</tr>
</tbody>
</table>

*Table 4: Benchmark function characteristics; $n$ is the number of dimensions of the problem.*
<table>
<thead>
<tr>
<th>Problem Dimension</th>
<th>BBO Wins</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>30</td>
<td>14</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Population Size</th>
<th>BBO Wins</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>50</td>
<td>14</td>
</tr>
</tbody>
</table>

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.

5 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 $\lambda_k$ 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; however, to be precise, BBO does not include selection pressure because it is not a reproductive algorithm.

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 allowed for the development of BBO, and it will allow for many extensions that are motivated by biogeography. These extensions should be emphasized in future BBO research and include such factors as modeling for nonlinear migration curves [25], 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 of the Markov transition matrix size with problem dimension, these results provide confidence for the application of BBO to larger, real-world problems.

We also 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 also supported with a set of benchmark comparisons.

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 to combine BBO with other EAs. Since no single algorithm can provide optimal performance for all possible problems [18], hybrid and adaptive algorithms are an important topic of current research. BBO has already been combined with opposition-based learning [16]. Future work should 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.
References


Asian Conference and Exhibition on Geospatial Information, Technology and Applications, August 2009.


Appendix 1

In this appendix we derive (21), which gives the expected value of $1_0(x_{m(k)}(s) - x(s))$. 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

$$g(K, V) = 1_0(x_{m(K,V)}(s) - x(s)) = 1_0(y_{K,V}(s) - x(s))$$

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

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

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,

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

$$= \frac{1}{N} \sum_{k=1}^{N} 1_0(y_{k,V}(s) - x(s)).$$

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

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

As shown in (16), there are a total of $C(n+N-1, N)$ possible population distributions. However, in our derivation of (21) 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 (43) can be written as

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

where

$$D = C(n + N - 2, N)$$
and \( v(j) \) is the \( j \)th 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 (21) assumes that \( y_{k,v(j)} \neq x_1 \) for all \( k, v(j) \)).

Equation (44) can then be written as

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

\[
= \frac{1}{ND} \sum_{i=2}^{n/2} \sum_{j=1}^{D} \sum_{k=1}^{N} \mathbf{1}_0(y_{k,v(j)} - x_i).  
\] (46)

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

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

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 \( j \)th population that are equal to \( x_i \), which is equal to \( v_i(j) \), as shown in (3). Equation (47) therefore becomes

\[
E_V \{ E_K[g(K,V)] \} = \frac{1}{ND} \sum_{i=2}^{n/2} \sum_{j=1}^{D} \sum_{v_i(j) \in [1, N]} v_i(j).  
\] (48)

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 \( \alpha \) 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. Equation (48) can therefore be written as
\[
E[V | g(K, V)] = \frac{1}{ND} \sum_{i=2}^{n/2} \sum_{\alpha=1}^{N} \alpha C(n + N - \alpha - 3, N - \alpha)
\]
\[
= \frac{n/2 - 1}{ND} \sum_{\alpha=0}^{N-1} (N - \alpha)C(n + \alpha - 3, \alpha)
\]
\[
= \frac{n/2 - 1}{n - 1}
\]
(49)

where we have used Theorem 1 (see below) to take the final step and obtain (21) as desired.

**Theorem 1** Suppose that \(n, N \in \mathbb{N}\) are natural numbers with \(n \geq 3\) and \(N \geq 1\). Then
\[
\sum_{j=0}^{N-1} (N - j)C(n + j - 3, j) = \frac{N}{n - 1} C(n + N - 2, N).
\]
(50)

**Proof:** We let \(n \in \mathbb{N}, n \geq 3\) be an arbitrary but fixed natural number. We apply the principle of mathematical induction (PMI) to the proposition \(P(N)\) of (50), which we rewrite as
\[
P(N) : \sum_{j=0}^{N-1} (N - j)\frac{(n + j - 3)!}{j!(n - 3)!} = \frac{N}{n - 1} \frac{(n + N - 2)!}{N!(n - 2)!}
\]
\[
= \frac{(n + N - 2)!}{(N - 1)!(n - 1)!}.
\]
(51)

It is straightforward to show that \(P(1)\) is true; substituting \(N = 1\) in (51) results in the identity \(1 = 1\). Next, assuming that \(P(N)\) is true for some \(N \geq 1\), we write
\[
\sum_{j=0}^{N} (N + 1 - j)\frac{(n + j - 3)!}{j!(n - 3)!} = \sum_{j=0}^{N-1} (N + 1 - j)\frac{(n + j - 3)!}{j!(n - 3)!} + \frac{(n + N - 3)!}{N!(n - 3)!}
\]
\[
= \sum_{j=0}^{N-1} (N - j)\frac{(n + j - 3)!}{j!(n - 3)!} + \sum_{j=0}^{N-1} \frac{(n + j - 3)!}{j!(n - 3)!} + \frac{(n + N - 3)!}{N!(n - 3)!}.
\]
(52)

Applying the inductive hypothesis (51) to the first term on the right side of (52) gives
\[
\sum_{j=0}^{N} (N + 1 - j)\frac{(n + j - 3)!}{j!(n - 3)!} = \frac{(n + N - 2)!}{(N - 1)!(n - 1)!} + \sum_{j=0}^{N-1} \frac{(n + j - 3)!}{j!(n - 3)!} + \frac{(n + N - 3)!}{N!(n - 3)!}.
\]
(53)
Applying Lemma 1, which follows this proof, to the second term on the right side of (53) gives

\[
\sum_{j=0}^{N} \frac{(N + 1 - j) (n + j - 3)!}{j! (n - 3)!} = \frac{N(n + N - 2)!}{N! (n - 1)!} + \frac{N(n + N - 3)!}{N! (n - 2)!} + \frac{(n + N - 3)!}{N! (n - 3)!} \\
= \frac{N(n + N - 2)!}{N! (n - 1)!} + \frac{N(n - 1)(n + N - 3)!}{N! (n - 1)!} + \frac{(n - 2)(n - 1)(n + N - 3)!}{N! (n - 1)!} \\
= \frac{1}{N! (n - 1)!} \left[ N(n + N - 2)! + N(n - 1)(n + N - 3)! + (n - 2)(n - 1)(n + N - 3)! \right] \\
= \frac{1}{N! (n - 1)!} \left[ N(n + N - 2)! + (n - 1)(n + N - 2)! \right] \\
= \frac{1}{N! (n - 1)!} \left[ N(n + N - 2)! + (n - 1)(n + N - 2)! \right] \\
= \frac{(n + N - 2)!}{N! (n - 1)!} (n + N - 1) \\
= \frac{(n + N - 1)!}{N! (n - 1)!}.
\]

According to the PMI, \( P(N) \) is true for all \( N \geq 1 \). Since \( n \geq 3 \) is arbitrary, (50) is true for all \( n \geq 3 \) and for all \( N \geq 1 \).

**QED**

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

\[
Q(N) : \sum_{j=0}^{N-1} \frac{(n + j - 3)!}{j! (n - 3)!} = \frac{N(n + N - 3)!}{N! (n - 2)!}.
\]

**Proof:** We need the constraints on \( n \) and \( N \) so that the arguments of the factorials are well-defined. Let \( n \in \mathbb{N}, n \geq 3 \) be an arbitrary but fixed natural number. We apply the PMI to the proposition \( Q(N) \) of (55). It is straightforward to show that \( Q(1) \) is true; substituting \( N = 1 \) in (55) results in the identity \( 1 = 1 \). Next, we write

\[
\sum_{j=0}^{N} \frac{(n + j - 3)!}{j! (n - 3)!} = \sum_{j=0}^{N-1} \frac{(n + j - 3)!}{j! (n - 3)!} + \frac{(n + N - 3)!}{N! (n - 3)!}.
\]
Assuming that $Q(N)$ is true for some $N \geq 1$ and applying the inductive hypothesis (55) to the first term on the right side of (56) yields

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

$$
= \frac{N(n+N-3)!}{N! (n-2)!} + \frac{(n-2)(n+N-3)!}{N! (n-2)!}
$$

$$
= \frac{N(n+N-3)! + (n-2)(n+N-3)!}{N! (n-2)!}
$$

$$
= \frac{(n+N-3)! (n+N-2)}{N! (n-2)!}
$$

$$
= \frac{(n+N-2)!}{N! (n-2)!}.
$$

(57)

According to the PMI, $Q(N)$ is true for all $N \geq 1$. Since $n \geq 3$ is arbitrary, (55) is true for all $n \geq 3$ and for all $N \geq 1$. QED

**Appendix 2**

Tables 7–10 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 indicate which algorithm can find the best solution over multiple runs. The numbers in the “Ave.” column show the average performance of 100 Monte Carlo simulations and indicate which algorithm performs best on average. The numbers in the $\sigma$ column 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.
<table>
<thead>
<tr>
<th>Function</th>
<th>BBO Best</th>
<th>GA Best</th>
<th>BBO Ave.</th>
<th>GA Ave.</th>
<th>BBO $\sigma$</th>
<th>GA $\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ackley</td>
<td>0.036</td>
<td>0</td>
<td>2.9</td>
<td>3.6</td>
<td>1.2</td>
<td>1.7</td>
</tr>
<tr>
<td>Fletcher</td>
<td>3</td>
<td>0.72</td>
<td>2.6e+003</td>
<td>5.2e+003</td>
<td>3.1e+002</td>
<td>6.9e+002</td>
</tr>
<tr>
<td>Griewank</td>
<td>1</td>
<td>1</td>
<td>1.1</td>
<td>1.1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Penalty #1</td>
<td>1.9e-006</td>
<td>0.00024</td>
<td>0.34</td>
<td>0.27</td>
<td>0.04</td>
<td>0.068</td>
</tr>
<tr>
<td>Penalty #2</td>
<td>3.4e-005</td>
<td>0.012</td>
<td>0.59</td>
<td>1</td>
<td>0.11</td>
<td>0.19</td>
</tr>
<tr>
<td>Quartic</td>
<td>0</td>
<td>0</td>
<td>6.2e-022</td>
<td>6.1e-007</td>
<td>6.2e-024</td>
<td>1e-008</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>0</td>
<td>0</td>
<td>0.9</td>
<td>2</td>
<td>0.033</td>
<td>0.16</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>0</td>
<td>0</td>
<td>3.8</td>
<td>7.9</td>
<td>1.8</td>
<td>2</td>
</tr>
<tr>
<td>Schwefel 1.2</td>
<td>0.34</td>
<td>2.8</td>
<td>1.5e+002</td>
<td>2.9e+002</td>
<td>16</td>
<td>61</td>
</tr>
<tr>
<td>Schwefel 2.21</td>
<td>0.26</td>
<td>0.068</td>
<td>4.1</td>
<td>5</td>
<td>1.7</td>
<td>2.3</td>
</tr>
<tr>
<td>Schwefel 2.22</td>
<td>0</td>
<td>0</td>
<td>0.29</td>
<td>0.48</td>
<td>0.048</td>
<td>0.1</td>
</tr>
<tr>
<td>Schwefel 2.26</td>
<td>0.26</td>
<td>0.44</td>
<td>30</td>
<td>40</td>
<td>5.8</td>
<td>9.6</td>
</tr>
<tr>
<td>Sphere</td>
<td>0</td>
<td>0</td>
<td>0.0064</td>
<td>0.024</td>
<td>0.00022</td>
<td>0.0018</td>
</tr>
<tr>
<td>Step</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>14</td>
<td>2.5</td>
<td>4.1</td>
</tr>
<tr>
<td><strong>BBO Wins</strong></td>
<td>57%</td>
<td>85%</td>
<td>92%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7: Comparison between BBO and GA/GUR performance on 5-dimensional benchmark problems with a population size of 50.

<table>
<thead>
<tr>
<th>Function</th>
<th>BBO Best</th>
<th>GA Best</th>
<th>BBO Ave.</th>
<th>GA Ave.</th>
<th>BBO $\sigma$</th>
<th>GA $\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ackley</td>
<td>0.75</td>
<td>0.82</td>
<td>4.1</td>
<td>4.9</td>
<td>2.6</td>
<td>3.1</td>
</tr>
<tr>
<td>Fletcher</td>
<td>3.5e+002</td>
<td>7.8e+002</td>
<td>1.5e+004</td>
<td>2.7e+004</td>
<td>4.2e+003</td>
<td>6.6e+003</td>
</tr>
<tr>
<td>Griewank</td>
<td>1.000</td>
<td>1.016</td>
<td>1.6</td>
<td>1.8</td>
<td>1.2</td>
<td>1.3</td>
</tr>
<tr>
<td>Penalty #1</td>
<td>0.015</td>
<td>0.088</td>
<td>1.8</td>
<td>1.4</td>
<td>0.36</td>
<td>0.51</td>
</tr>
<tr>
<td>Penalty #2</td>
<td>0.14</td>
<td>0.39</td>
<td>3.7</td>
<td>5.5</td>
<td>1.2</td>
<td>2</td>
</tr>
<tr>
<td>Quartic</td>
<td>0</td>
<td>0</td>
<td>1.9e-007</td>
<td>0.00019</td>
<td>2e-009</td>
<td>2.2e-006</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0.28</td>
<td>0.75</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>0</td>
<td>5.5</td>
<td>82</td>
<td>74</td>
<td>14</td>
<td>20</td>
</tr>
<tr>
<td>Schwefel 1.2</td>
<td>55</td>
<td>89</td>
<td>1.2e+003</td>
<td>2.8e+003</td>
<td>3.8e+002</td>
<td>7.9e+002</td>
</tr>
<tr>
<td>Schwefel 2.21</td>
<td>3.8</td>
<td>3.5</td>
<td>15</td>
<td>23</td>
<td>8.2</td>
<td>10</td>
</tr>
<tr>
<td>Schwefel 2.22</td>
<td>0</td>
<td>0</td>
<td>0.64</td>
<td>0.96</td>
<td>0.22</td>
<td>0.41</td>
</tr>
<tr>
<td>Schwefel 2.26</td>
<td>5.5</td>
<td>13</td>
<td>1.8e+002</td>
<td>1.5e+002</td>
<td>48</td>
<td>70</td>
</tr>
<tr>
<td>Sphere</td>
<td>0</td>
<td>0</td>
<td>0.11</td>
<td>0.17</td>
<td>0.0077</td>
<td>0.023</td>
</tr>
<tr>
<td>Step</td>
<td>5</td>
<td>8</td>
<td>74</td>
<td>1.4e+002</td>
<td>25</td>
<td>38</td>
</tr>
<tr>
<td><strong>BBO Wins</strong></td>
<td>90%</td>
<td>77%</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8: Comparison between BBO and GA/GUR performance on 10-dimensional benchmark problems with a population size of 50.
<table>
<thead>
<tr>
<th></th>
<th>BBO Best</th>
<th>GA Best</th>
<th>BBO Ave.</th>
<th>GA Ave.</th>
<th>BBO σ</th>
<th>GA σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ackley</td>
<td>3.1</td>
<td>3.0</td>
<td>6.0</td>
<td>7.3</td>
<td>4.5</td>
<td>5.2</td>
</tr>
<tr>
<td>Fletcher</td>
<td>1.5e+004</td>
<td>1.5e+004</td>
<td>1.1e+005</td>
<td>1.6e+005</td>
<td>4.5e+004</td>
<td>5.5e+004</td>
</tr>
<tr>
<td>Griewank</td>
<td>1.6</td>
<td>2.3</td>
<td>7.5</td>
<td>6.7</td>
<td>3.3</td>
<td>4.0</td>
</tr>
<tr>
<td>Penalty #1</td>
<td>1.5</td>
<td>2.3</td>
<td>61</td>
<td>2.8e+002</td>
<td>5.3</td>
<td>11</td>
</tr>
<tr>
<td>Penalty #2</td>
<td>4.6</td>
<td>7.7</td>
<td>2.4e+004</td>
<td>3.1e+004</td>
<td>1.1e+003</td>
<td>2.6e+003</td>
</tr>
<tr>
<td>Quartic</td>
<td>0</td>
<td>0</td>
<td>0.00061</td>
<td>0.0026</td>
<td>0.00002</td>
<td>0.00011</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>0</td>
<td>0.013</td>
<td>6.8</td>
<td>10</td>
<td>2.7</td>
<td>3.8</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>12</td>
<td>18</td>
<td>1.5e+002</td>
<td>1.9e+002</td>
<td>68</td>
<td>96</td>
</tr>
<tr>
<td>Schwefel 1.2</td>
<td>1.0e+003</td>
<td>1.4e+003</td>
<td>6.4e+003</td>
<td>9.5e+003</td>
<td>3.1e+003</td>
<td>5.3e+003</td>
</tr>
<tr>
<td>Schwefel 2.21</td>
<td>11</td>
<td>17</td>
<td>38</td>
<td>41</td>
<td>25</td>
<td>31</td>
</tr>
<tr>
<td>Schwefel 2.22</td>
<td>0.22</td>
<td>0.96</td>
<td>3.6</td>
<td>5.8</td>
<td>1.8</td>
<td>2.6</td>
</tr>
<tr>
<td>Schwefel 2.26</td>
<td>1.4e+002</td>
<td>1.6e+002</td>
<td>7.9e+002</td>
<td>8.4e+002</td>
<td>3.6e+002</td>
<td>4.5e+002</td>
</tr>
<tr>
<td>Sphere</td>
<td>0.0061</td>
<td>0.012</td>
<td>1.3</td>
<td>1.9</td>
<td>0.30</td>
<td>0.63</td>
</tr>
<tr>
<td>Step</td>
<td>82</td>
<td>1.3e+002</td>
<td>4.3e+002</td>
<td>6.8e+002</td>
<td>2.5e+002</td>
<td>3.5e+002</td>
</tr>
</tbody>
</table>

BBO Wins | 92%        | 93%        | 100%     

Table 9: Comparison between BBO and GA/GUR performance on 20-dimensional benchmark problems with a population size of 50.

<table>
<thead>
<tr>
<th></th>
<th>BBO Best</th>
<th>GA Best</th>
<th>BBO Ave.</th>
<th>GA Ave.</th>
<th>BBO σ</th>
<th>GA σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ackley</td>
<td>4.1</td>
<td>4.7</td>
<td>8.0</td>
<td>9.3</td>
<td>6.4</td>
<td>7.1</td>
</tr>
<tr>
<td>Fletcher</td>
<td>5.9e+004</td>
<td>6.6e+004</td>
<td>3.0e+005</td>
<td>3.4e+005</td>
<td>1.6e+005</td>
<td>1.8e+005</td>
</tr>
<tr>
<td>Griewank</td>
<td>4.5</td>
<td>5.1</td>
<td>16</td>
<td>19</td>
<td>9.2</td>
<td>12</td>
</tr>
<tr>
<td>Penalty #1</td>
<td>8.4</td>
<td>10</td>
<td>8.7e+004</td>
<td>2.2e+005</td>
<td>1.5e+003</td>
<td>4.2e+003</td>
</tr>
<tr>
<td>Penalty #2</td>
<td>2.5e+002</td>
<td>7.2e+002</td>
<td>1.0e+006</td>
<td>1.1e+006</td>
<td>6.3e+004</td>
<td>1.7e+005</td>
</tr>
<tr>
<td>Quartic</td>
<td>7.7e–008</td>
<td>1e–005</td>
<td>0.049</td>
<td>0.37</td>
<td>0.0038</td>
<td>0.031</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>2</td>
<td>5</td>
<td>14</td>
<td>16</td>
<td>7.4</td>
<td>9.7</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>38</td>
<td>61</td>
<td>3.4e+002</td>
<td>3.7e+002</td>
<td>1.7e+002</td>
<td>2.0e+002</td>
</tr>
<tr>
<td>Schwefel 1.2</td>
<td>5.0e+003</td>
<td>6.3e+003</td>
<td>1.4e+004</td>
<td>2.7e+004</td>
<td>8.9e+003</td>
<td>1.5e+004</td>
</tr>
<tr>
<td>Schwefel 2.21</td>
<td>24</td>
<td>30</td>
<td>49</td>
<td>58</td>
<td>36</td>
<td>43</td>
</tr>
<tr>
<td>Schwefel 2.22</td>
<td>1.6</td>
<td>3.8</td>
<td>10</td>
<td>13</td>
<td>5.9</td>
<td>7.4</td>
</tr>
<tr>
<td>Schwefel 2.26</td>
<td>4.4e+002</td>
<td>7.0e+002</td>
<td>1.6e+003</td>
<td>1.8e+003</td>
<td>9.6e+002</td>
<td>1.2e+003</td>
</tr>
<tr>
<td>Sphere</td>
<td>0.41</td>
<td>0.85</td>
<td>4.2</td>
<td>6.6</td>
<td>1.9</td>
<td>2.8</td>
</tr>
<tr>
<td>Step</td>
<td>3.4e+002</td>
<td>6.5e+002</td>
<td>1.5e+003</td>
<td>2.6e+003</td>
<td>9.0e+002</td>
<td>1.2e+003</td>
</tr>
</tbody>
</table>

BBO Wins | 100%        | 100%        | 100%     

Table 10: Comparison between BBO and GA/GUR performance on 30-dimensional benchmark problems with a population size of 50.
Appendix 3

Tables 11–12 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, 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 10 in Appendix 2 gives a detailed breakdown of the third row of Table 6.

<table>
<thead>
<tr>
<th>Function</th>
<th>BBO Best</th>
<th>GA Best</th>
<th>BBO Ave.</th>
<th>GA Ave.</th>
<th>BBO σ</th>
<th>GA σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ackley</td>
<td>9.9</td>
<td>9.7</td>
<td>15</td>
<td>15</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>Fletcher</td>
<td>1.7e+005</td>
<td>1.7e+005</td>
<td>6.4e+005</td>
<td>8.3e+005</td>
<td>3.8e+005</td>
<td>4.6e+005</td>
</tr>
<tr>
<td>Griewank</td>
<td>26</td>
<td>28</td>
<td>94</td>
<td>1.1e+002</td>
<td>57</td>
<td>61</td>
</tr>
<tr>
<td>Penalty #1</td>
<td>3.6e+003</td>
<td>3.5e+004</td>
<td>1.3e+007</td>
<td>3.5e+007</td>
<td>2.3e+006</td>
<td>4.4e+006</td>
</tr>
<tr>
<td>Penalty #2</td>
<td>7.4e+005</td>
<td>7.9e+005</td>
<td>4.0e+007</td>
<td>6.4e+007</td>
<td>9.6e+006</td>
<td>1.6e+007</td>
</tr>
<tr>
<td>Quartic</td>
<td>0.094</td>
<td>0.2</td>
<td>18</td>
<td>28</td>
<td>3.6</td>
<td>5.8</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>21</td>
<td>23</td>
<td>68</td>
<td>82</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>2.5e+002</td>
<td>2.4e+002</td>
<td>1.0e+003</td>
<td>1.6e+003</td>
<td>5.6e+002</td>
<td>6.6e+002</td>
</tr>
<tr>
<td>Schwefel 1.2</td>
<td>9.2e+003</td>
<td>1.5e+004</td>
<td>4.7e+004</td>
<td>7.7e+004</td>
<td>1.9e+004</td>
<td>2.6e+004</td>
</tr>
<tr>
<td>Schwefel 2.21</td>
<td>41</td>
<td>48</td>
<td>82</td>
<td>88</td>
<td>66</td>
<td>69</td>
</tr>
<tr>
<td>Schwefel 2.22</td>
<td>18</td>
<td>17</td>
<td>37</td>
<td>39</td>
<td>28</td>
<td>29</td>
</tr>
<tr>
<td>Schwefel 2.26</td>
<td>1.8e+003</td>
<td>2.3e+003</td>
<td>5.2e+003</td>
<td>4.9e+003</td>
<td>3.4e+003</td>
<td>3.7e+003</td>
</tr>
<tr>
<td>Schwefel 2.26</td>
<td>5.3</td>
<td>8.9</td>
<td>32</td>
<td>34</td>
<td>18</td>
<td>21</td>
</tr>
<tr>
<td>Sphere</td>
<td>5.3</td>
<td>8.9</td>
<td>32</td>
<td>34</td>
<td>18</td>
<td>21</td>
</tr>
<tr>
<td>Step</td>
<td>1.8e+003</td>
<td>3.1e+003</td>
<td>1.2e+004</td>
<td>1.3e+004</td>
<td>6.4e+003</td>
<td>7.4e+003</td>
</tr>
</tbody>
</table>

Table 11: Comparison between BBO and GA/GUR performance on 30-dimensional benchmark problems with a population size of 10.
<table>
<thead>
<tr>
<th>Function</th>
<th>BBO Best</th>
<th>GA Best</th>
<th>BBO Ave.</th>
<th>GA Ave.</th>
<th>BBO $\sigma$</th>
<th>GA $\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ackley</td>
<td>7.0</td>
<td>7.3</td>
<td>12.0</td>
<td>12.6</td>
<td>9.4</td>
<td>10</td>
</tr>
<tr>
<td>Fletcher</td>
<td>8.2e+004</td>
<td>1.5e+005</td>
<td>4.4e+005</td>
<td>6.0e+005</td>
<td>2.6e+005</td>
<td>3.1e+005</td>
</tr>
<tr>
<td>Griewank</td>
<td>11</td>
<td>17</td>
<td>48</td>
<td>53</td>
<td>26</td>
<td>31</td>
</tr>
<tr>
<td>Penalty #1</td>
<td>19</td>
<td>52</td>
<td>5.0e+006</td>
<td>1.3e+007</td>
<td>2.2e+005</td>
<td>6.3e+005</td>
</tr>
<tr>
<td>Penalty #2</td>
<td>1.1e+005</td>
<td>3.6e+003</td>
<td>9.6e+006</td>
<td>1.6e+007</td>
<td>1.8e+006</td>
<td>2.7e+006</td>
</tr>
<tr>
<td>Quartic</td>
<td>0.0021</td>
<td>0.0032</td>
<td>4.6</td>
<td>8.1</td>
<td>0.46</td>
<td>0.87</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>10</td>
<td>11</td>
<td>36</td>
<td>37</td>
<td>21</td>
<td>25</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>1.3e+002</td>
<td>93</td>
<td>6.8e+002</td>
<td>6.2e+002</td>
<td>3.2e+002</td>
<td>3.6e+002</td>
</tr>
<tr>
<td>Schwefel 1.2</td>
<td>6.9e+003</td>
<td>1.2e+004</td>
<td>2.3e+004</td>
<td>3.5e+004</td>
<td>1.4e+004</td>
<td>2.0e+004</td>
</tr>
<tr>
<td>Schwefel 2.21</td>
<td>36</td>
<td>45</td>
<td>69</td>
<td>78</td>
<td>53</td>
<td>59</td>
</tr>
<tr>
<td>Schwefel 2.22</td>
<td>7.9</td>
<td>9.4</td>
<td>24</td>
<td>30</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td>Schwefel 2.26</td>
<td>1.3e+003</td>
<td>1.2e+003</td>
<td>3.0e+003</td>
<td>3.3e+003</td>
<td>2.1e+003</td>
<td>2.4e+003</td>
</tr>
<tr>
<td>Sphere</td>
<td>3.4</td>
<td>5.3</td>
<td>18</td>
<td>19</td>
<td>8.1</td>
<td>11</td>
</tr>
<tr>
<td>Step</td>
<td>1.1e+003</td>
<td>2.0e+003</td>
<td>5.0e+003</td>
<td>7.7e+003</td>
<td>2.8e+003</td>
<td>3.8e+003</td>
</tr>
</tbody>
</table>

| BBO Wins  | 79%      | 93%      | 100%     |

Table 12: Comparison between BBO and GA/GUR performance on 30-dimensional benchmark problems with a population size of 20.