A Markov model of biogeography-based optimization for complex systems
Dawei Du, Dan Simon
Cleveland State University
Email: d.du@csuohio.edu, d.j.simon@csuohio.edu

Abstract
Biogeography-based optimization (BBO) is an evolutionary algorithm that is inspired by nature. Because of its good performance relative to other evolutionary algorithms, it has become a popular algorithm for many different applications. Also, it has been modified in several ways to accommodate various types of problems. BBO for complex systems (BBO/Complex) is a recently-developed variation of BBO which provides a state-of-the-art algorithm for the optimization of complex systems (that is, systems with multiple related subsystems, each including multiple objectives and multiple constraints). Markov models have been used to theoretically model and analyze different versions of BBO, and to successfully predict steady-state BBO results. In this paper, we derive a Markov model for BBO/Complex. The Markov model derived here provides an exact mathematical model of BBO/Complex in the limit as the generation count becomes infinite. Simulation results are provided to confirm the new Markov model.

1. Introduction
Biogeography-based optimization (BBO) was invented in 2008. The original BBO algorithm demonstrated good performance on 14 benchmark problems as compared with seven well-known optimization algorithms – ant colony optimization (ACO), differential evolution (DE), evolution strategy (ES), genetic algorithm (GA), population-based incremental learning (PBIL), particle swarm optimization (PSO), and stud genetic algorithm (SGA) [1]. In recent years, BBO has become a popular optimization algorithm for many applications [2] [3] [4].

The environment of BBO is represented by an archipelago, which consists of islands. Each island corresponds to a possible solution to the optimization problem. Each island contains many species, each of which corresponds to an independent variable (also called feature) of the candidate solution. In BBO, each feature is also called a suitability index variable (SIV), which is terminology that is borrowed from the biogeography literature. In the original BBO algorithm, the evaluation of each candidate solution is performed with a cost function, and we use habitat suitability
index (HSI) to indicate the performance cost of candidate solution, which is another term borrowed from the biogeography literature. The evolutionary process of BBO is comprised of two techniques – migration and mutation. The purpose of migration is to share features among islands. The purpose of mutation is to add random features to the population, which can effectively prevent the population from getting stuck in local optima.

The basic procedure of the BBO algorithm is as follows [5].

1. Define the mutation probability, and the elitism parameter. These parameters are the same as in GAs and many other evolutionary algorithms.

2. Initialize the population, either randomly or based on expert knowledge.

3. Calculate the immigration rate and emigration rate for each island (that is, for each candidate solution in the population). Good solutions have high emigration rates and low immigration rates, indicating that they are likely to share their features with the rest of the population, but unlikely to import new features from the rest of the population. Poor solutions have low emigration rates and high immigration rates.

4. For the current generation, probabilistically choose the immigrating islands based on the immigration rates. Use roulette wheel selection based on the emigration rates to select the emigrating islands.

5. Migrate randomly selected SIVs (that is, independent variables) based on the selected islands in the previous step.

6. Probabilistically perform mutation based on the mutation probability for each island.

7. Calculate the performance cost of each individual island.

8. If the termination criterion is met, terminate; otherwise, go to step 3.

Heuristic algorithms can be effective optimization tools when used in conjunction with more traditional methods. They are often used in problems that are complex and too difficult for traditional solution methods. Unlike traditional methods, we cannot obtain the solution quickly by using heuristic algorithms. As typified by the basic procedure of BBO as listed above, most heuristic algorithms have a similar evolution process in their search for an optimal solution. In contrast with more traditional and analytic optimization algorithms, there is no guarantee that we can
obtain the optimal solution with heuristic algorithms. However, heuristic algorithms are effective at finding candidate solutions that are in the neighborhood of local optima.

Markov models are general tools that are used to describe the probability of transitioning from one state to another. If we can develop a Markov model for a system, the probability of the appearance of each state can be calculated mathematically. If we treat a Markov state as a distribution of individuals in a heuristic algorithm, then we can use the Markov model to calculate the probability of the appearance of any given population distribution, which means that we can calculate the probability that the optimal solution will be found by the heuristic algorithm. In this way, Markov models can be used to mathematically analyze the performance of heuristic algorithms for given optimization problems. Markov models have been successfully applied to various heuristic algorithms, such as simple genetic algorithms [6], simulated annealing [7], genitor algorithm, and CHC algorithm [8]. In 2010, a Markov model was developed for BBO, and that was the first time that the performance of BBO was analyzed mathematically and theoretically [9].

The remainder of this paper is organized as follows. In Section 2, we develop a Markov model of BBO for complex systems (BBO/Complex). Section 3 provides simulation results to verify the Markov model. Section 4 concludes the paper and provides ideas for future research directions.

### 2. A Markov Chain Model of BBO/Complex

Section 2.1 reviews the BBO/Complex algorithm, and Section 2.2 derives a Markov model of the BBO/Complex algorithm.

#### 2.1 BBO/Complex

Because of the popularity of BBO, many modified versions of BBO have been developed for different situations and different types of problems [10] [11] [12]. BBO/Complex is one of the successfully modified versions of BBO [13] [14], and was specifically designed for complex systems. Although BBO/Complex still belongs to the BBO family, its environment is different from the original BBO algorithm because of the structure of complex systems.

Complex systems are widely used in industry, and although they appear often, they are hard to optimize. In [15], a complex system is defined as "[a]n assembly of interacting members that is difficult to understand as a whole." Based on [16], a complex system contains the following properties: (1) it is comprised of a large
number of elements; (2) the interactions between the elements are rich; and (3) the interactions between the elements have special but difficult characteristics, such as nonlinearity.

Based on the definition and properties of a complex system, the original BBO algorithm is not sufficient to optimize this type of system, and a special modification is needed. The fundamental characteristic of standard BBO is its environment – an archipelago. An archipelago consists of islands, and each island represents a candidate solution to the optimization problem. But a complex system consists of multiple subsystems, each of which can be considered partially independent from the others, although the subsystems do interact with each other. Because of their organizational independence, each subsystem can be treated, to a large extent, as an independent subsystem. In order to adapt BBO to complex systems, the environment of BBO needs to be changed to a multi-archipelago environment.

The term “complex system” indicates not only the complexity of the system as a whole, but also its multi-subsystem structure, and its multiple and related optimization objectives and constraints. Usually, a complex system contains multi-objectives and multi-constraints, characteristics that are not directly handled by the original BBO algorithm. BBO/Complex is discussed in detail in [14] and can be summarized as follows.

1. Define the control parameters: population size, stopping criteria, mutation probability, and elitism parameter. A typical setup for BBO is that population size is 100, stopping criteria is 100,000 cost function calls, mutation probability is 0.05, and elitism parameter is 1.

2. Initialize the population. This is usually done with randomly-generated islands (candidate solutions).

3. Calculate the constraint and objective similarity levels between each pair of subsystems.

4. Calculate the rank of each island in its subsystem, where rank considers both the cost function values and the constraint violations.

5. Within-subsystem migration: Probabilistically choose the immigrating islands based on the island ranks. Use roulette wheel selection based on the emigration rates to select the emigrating islands. Emigration rates are linearly related to the island ranks. After an emigrating island is selected for each immigrating island, perform within-subsystem migration. Each SIV (independent variable) in each immigrating island has a chance to be replaced by an SIV from an emigrating island.
6. Cross-subsystem migration: Probabilistically choose the immigrating islands based on the island ranks. Find suitable pairs of subsystems based on similarity levels of subsystem constraints and objectives. Calculate distances between each pair of islands from different subsystems. Use roulette wheel selection based on partial distances to select the emigrating islands. Then perform cross-subsystem migration between immigrating islands and emigrating islands. Each SIV (independent variable) in each immigrating island has a chance to be replaced by an SIV from an emigrating island.

7. Probabilistically perform mutation on each island based on the mutation probability.

8. Save the best islands in each subsystem as elite islands. Replace the worst islands in the population with the previous generation’s elite islands.

9. If the termination criterion is not met, go to step 4; otherwise, terminate.

### 2.2 A Markov model of BBO/Complex

Markov models describe the probability that a system transitions from one state to another. It is a discrete-time random process on a finite state space. Assume that there are \( T \) possible states in some system. Then a \( T \times T \) transition matrix can be created to describe the probability of transitioning between each pair of states. We call this transition matrix \( P \). The probability that state \( S_i \) transits to state \( S_j \) is given by \( P_{ij} \), which is also called the transition probability. If a Markov model can transition from any state to any other state, then \( P \) does not include any zero entries, and the transition matrix \( P \) of the Markov chain is called regular. If \( P \) is regular, we can obtain the steady state transition matrix \( P_{ss} \) as follows [9] [16]:

\[
\lim_{n \to \infty} P^n = P_{ss}
\]  
(2.1)

Equation (2.1) gives the transition matrix after an infinite number of transitions. Each row in \( P_{ss} \) is the same as every other row, and the \( i \)-th element in each row is the limiting probability of the occurrence of state \( i \) as the number of transitions approaches infinity.

If BBO/Complex is implemented on a system with discrete independent variables, then it has a finite number of population distributions, and we can derive a Markov model for it. Each population distribution represents a state in the Markov model. As shown in Equation (2.1), \( P_{ss} \) is independent from the initial state. In BBO, this means that the final population distribution is independent of the initial population. This
result is of great importance in this paper. We only need the transition matrix to predict the final population distribution (in the limit as the generation count approaches infinity), and this limiting distribution is independent from the initial population. For the simulations that will be used to verify the Markov model later in this paper (Section 3), we do not need to be particular about the initial population – all initial populations will eventually lead to the same final population distribution.

The environment of BBO/Complex is comprised of $M$ subsystems. We assume that the independent variables of the optimization problem are binary, although we note that any discrete space can easily be mapped into a binary space. The number of bits for each island (candidate solution) in subsystem $i$ is denoted as $p_i$. The population size of BBO for subsystem $i$ is denoted as $n_i$. The total number of possible solutions in subsystem $i$ is denoted as $N_i$, and the total number of possible solutions in the entire system is denoted as $N$. $N_i$ and $N$ are calculated as follows:

$$N_i = 2^{p_i}$$

$$N = \prod_{i=1}^{M} N_i$$

(2.2)

The $j$-th island (candidate solution) in the population of subsystem $i$ is denoted as $y_{ij}$. The $j$-th point in the search space of the subsystem $i$ is denoted as $x_{ij}$. We use $v_{ij}$ to denote the total number of $x_{ij}$ islands in subsystem $i$. So the combined BBO/Complex population of the entire system can be generally represented as follows:

$$\text{Population} = \left\{ \left[ y_{11}, \ldots, y_{1N_1} \right], \left[ y_{21}, \ldots, y_{2N_2} \right], \ldots, \left[ y_{M1}, \ldots, y_{MN_M} \right] \right\}$$

$$= \left\{ \left[ x_{11}, \ldots, x_{11} \right], \left[ x_{12}, \ldots, x_{12} \right], \ldots, \left[ x_{1N_1}, \ldots, x_{1N_1} \right] \right\},$$

$$\left\{ \left[ x_{21}, \ldots, x_{21} \right], \left[ x_{22}, \ldots, x_{22} \right], \ldots, \left[ x_{2N_2}, \ldots, x_{2N_2} \right] \right\},$$

$$\vdots$$

$$\left\{ \left[ x_{M1}, \ldots, x_{M1} \right], \left[ x_{M2}, \ldots, x_{M2} \right], \ldots, \left[ x_{MN_M}, \ldots, x_{MN_M} \right] \right\}$$

(2.3)

For convenience in notation, we have ordered the $y_{ij}$ islands in the same order as the $x_{ij}$ search space points. Based on Equation (2.3), the population in subsystem $i$ can be rewritten in a more compact format as follows:
for $i = 1, \ldots, M$. In Equation (2.4), $y_{ik}$ denotes the $k$-th island in the population of subsystem $i$. We use $y_{ik}(s)$ to represent the $s$-th bit in the $k$-th island in the population of subsystem $i$. Equation (2.4) can be written as follows:

$$y_{ik} = \begin{cases} 
  x_{i1}, & \text{when } k = 1, \ldots, v_{i1} \\
  x_{i2}, & \text{when } k = v_{i1} + 1, \ldots, v_{i1} + v_{i2} \\
  \vdots \\
  x_{iN}, & \text{when } k = \sum_{l=1}^{N-1} v_{il} + 1, \ldots, \sum_{l=1}^{N} v_{il}
\end{cases}$$

(2.4)

Based on the definition of BBO/Complex, islands in different subsystems have different structures and contain different types of SIVs. For ease of notation, we use a unified format to represent each island in the BBO/Complex population. The unified format is given as follows:

$$z(k) = \min r, \text{ such that } \sum_{i=1}^{r} v_{il} > k$$

(2.5)

This unified format includes all the SIV types in the entire system. The unified format shows the value of an island’s SIV if the island contains that particular SIV type, and the unified format shows N/A if the island does not contain that particular SIV type. For example, suppose the unified format is $[\text{SIV}_1, \text{SIV}_2, \text{N/A}, \cdots, \text{SIV}_z]$, but subsystem 1 only contains types $\text{SIV}_1$, $\text{SIV}_2$, $\text{SIV}_3$ and $\text{SIV}_4$. Then an island in subsystem 1 might be represented as $[0, 1, 0, 1, \text{N/A}]$.

A Markov model describes the transitions between states. Each state in BBO/Complex is a specific population distribution. Each generation of BBO/Complex updates its population with migration and mutation. A transition between states corresponds to the evolution of the population in one generation of BBO/Complex. So in order to build a transition matrix, we need to model migration and mutation in BBO/Complex. In the following subsections, we study the migration and mutation processes, and use them to build the transition matrix for the Markov model of BBO/Complex.

### 2.2.1 Migration

Migration is the main technique that we use to share information among islands. In
the original BBO algorithm, the basic procedure of migration is to probabilistically select an immigrating island (an island that imports SIVs) and an emigrating island (an island that exports SIVs). Then we probabilistically choose some SIVs from the emigrating island, and use them to replace the SIVs in the immigrating island. Figure 1 illustrates a simple migration process.

![Migration Process Diagram]

Figure 1: An example of migration between two islands.

In BBO/Complex, the migration process is more complicated. Rather than having just one population, it contains multiple populations, one for each subsystem. Each subsystem is relatively independent from the others, which means the construction of the population of each subsystem is relatively independent from the others. Also, there are two types of migration in BBO/Complex, within-subsystem migration and cross-subsystem migration, which introduces further complexity to the Markov model development.

There are four assumptions we make in this section to develop the Markov model. They are similar (but expanded) versions of the assumptions used to develop the Markov model for the original BBO algorithm [9].

First, a BBO solution will not be replaced until the end of the generation. In other words, BBO is generational rather than steady-state [17]. This assumption guarantees that the migration probabilities remain the same throughout each generation.

Second, an island can emigrate to itself. The immigrating and emigrating islands are probabilistically chosen from the entire population. So there is a chance that the immigrating and emigrating islands are the same. This is similar to a chromosome crossing over with itself in a GA.

Third, migration only happens between SIVs with the same type. The environment of BBO/Complex is a group of archipelagos. Each archipelago has a unique population
structure, depending on the subsystem with which it is associated. So islands from different archipelagos might not contain the same SIV types. SIVs represent features, and each feature has a unique domain. That is the reason that migration is only valid between the same SIV types. For example, suppose some island consists of five SIVs, where SIV1 is the proportional gain of a PID controller, SIV2 is the integral gain of a PID controller, SIV3 is the derivative gain of a PID controller, SIV4 is the acceleration rate of an engine, and SIV5 is the acceleration time of an engine. Each SIV has a unique type and parameter domain. For example, the domain of SIV1 might be from 0.5 to 1, while the domain of SIV2 might be from 0.1 to 0.4. Migration between SIV1 and SIV2 would not make sense because SIV1 represents a proportional gain while SIV2 represents an integral gain, which is a completely different parameter with a completely different purpose.

Fourth, we use predetermined migration rates for each island rank rather than calculating the migration rates each generation. All the ranks are calculated each generation based on the non-dominated sorting method [18]. The emigration rate \( \mu \) and immigration rate \( \lambda \) of each island are calculated based on the rank of the island, which is similar to the original BBO algorithm, except here we use ranks based on multi-objectives and multi-constraints, instead of ranks based on scalar cost values.

**Within-Subsystem Migration**

BBO/Complex contains two types of migration: within-subsystem migration and cross-subsystem migration. Within-subsystem migration is similar to the original migration method in BBO, and it is used for migration between islands within the same subsystem. This migration process has two possible situations. First, since it is selected probabilistically, it might not be performed, which means the features in the potential immigrating island won’t be changed from one generation to the next. This situation is represented for the \( k \)-th individual in the \( i \)-th subsystem as follows:

\[
y_{ik}^{(s)}(t+1) = y_{ik}^{(s)}(t) = x_{iz(k)}^{(s)}(2.7)
\]

The second situation is that migration is performed to the potential immigrating island. The probability of obtaining a certain bit at a certain locus in a given island is proportional to two factors: the total number of occurrences of that bit in the entire subsystem population; and the emigration rates of the islands that contain those bits. This probability is calculated for the \( k \)-th individual in the \( i \)-th subsystem as follows:

\[
Pr(y_{ik}^{(s)}(t+1) = x_{il}^{(s)}(s) | \text{immigration}) = \frac{\sum_{j \in J_{il}(s)} v_{ij} \mu_{ij}}{\sum_{j=1}^{n_i} v_{ij} \mu_{ij}} (2.8)
\]

where \( J_{il}(s) \) is the set of islands in subsystem \( i \) that contain the same bit in position \( s \) as island \( x_{il} \):
\[ J_d(s) = \left\{ j : x_j^s(s) = x_d^s(s) \right\} \]  

(2.9)

Considering both situations described above and combining them into one equation, we obtain the probability of obtaining a given bit from within-subsystem migration:

\[
\begin{align*}
Pr_{\text{m-within-sub}}(y_{ik}^s(s)_{t+1} = x_d^s(s)) &= Pr(\text{no immigration})(y_{ik}^s(s)_{t+1} = x_d^s(s) | \text{no immigration}) \\
&+ Pr(\text{immigration})(y_{ik}^s(s)_{t+1} = x_d^s(s) | \text{immigration}) \\
&= (1 - \lambda_{iz(k)})I_0(x_{iz(k)}(s) - x_d^s(s)) + \lambda_{iz(k)} \frac{\sum_{j=1}^{n_i} v_{ij} \mu_{ij}}{\sum_{j=1}^{n_i} v_{ij} \mu_{ij}} 
\end{align*}
\]

(2.10)

Assume we have \( q \) bits in our unified island format, as given in Equation (2.6). The probability that the \( k \)-th individual in the \( i \)-th subsystem is equal to a given island \( x_d \) can be calculated based on Equation (2.10), which shows the probability of obtaining a single bit. We use \( P^{(1)}_{ikl}(v) \) to denote this probability, which is a function of the current population vector \( v \) at the \( t \)-th generation. (The term population vector will be defined later in Section 2.2, but for now we simply need to know that it represents the current population in the BBO algorithm.) This probability is given as follows:

\[
P^{(1)}_{ikl}(v) = Pr(y_{ik,t+1} = x_d) = \prod_{s=1}^{q} \left[ (1 - \lambda_{iz(k)})I_0(x_{iz(k)}(s) - x_d^s(s)) + \lambda_{iz(k)} \frac{\sum_{j=1}^{n_i} v_{ij} \mu_{ij}}{\sum_{j=1}^{n_i} v_{ij} \mu_{ij}} \right] 
\]

(2.11)

Cross-Subsystem Migration

The second type of migration is called cross-subsystem migration, which is the migration process between subsystems. Cross-subsystem migration is more complicated than within-subsystem migration for two reasons: (1) the island structure varies from subsystem to subsystem; (2) Ranking cannot be used to compare the performance cost of islands across subsystems, because rank information is useless when islands do not share identical cost and constraint functions. Issue 1 can be addressed with our unified island structure, which is shown in Equation (2.6). For issue 2, we need to introduce a new strategy for island selection that is specifically geared toward the optimization of multiple related subsystems, and this strategy will be introduced later in this section.

For cross-subsystem migration, when considering the possibility of immigration to a given individual, we have the same two possibilities as we do for within-subsystem
migration: (1) migration is not performed (recall that the migration decision is made probabilistically); (2) migration is performed. The first scenario is exactly like the corresponding scenario in within-subsystem migration. The only thing we need to reconsider here is the second possibility, and how to compute the probability of occurrence of each island, since ranks within a subsystem do not indicate their cost values relative to islands in other subsystems. BBO/Complex introduces the concept of distances between islands for the selection of islands in cross-subsystem migration [14] [19]. The motivation of this method is based on the concept of diversity: a larger diversity in a population provides more opportunities to find an optimal solution. The probability that we obtain a given bit $x_{il}(s)$ at a given position $s$ in the $k$-th individual $y_{ik}$ in the $i$-th subsystem is calculated as follows:

$$
Pr_{\text{im-cross-sub}}(y_{ik}(s)_{t+1} = x_{il}(s) \mid \text{immigration from subsystem } m) = Pr(\text{no immigration})(y_{ik}(s)_{t+1} = x_{il}(s) \mid \text{no immigration}) + Pr(\text{immigration})(y_{ik}(s)_{t+1} = x_{il}(s) \mid \text{immigration})
$$

$$
= (1 - \lambda_{iz(k)})I_0(x_{iz(k)}(s) - x_{il}(s)) + \lambda_{iz(k)} \sum_{j \in J} V_{mj} \sigma_{ilmj} \sum_{j=1}^{n_m} V_{mj} \sigma_{ilmj}
$$

(2.12)

$\sigma_{ilmj}$: distance between island $l$ in subsystem $i$ and island $j$ in subsystem $m$.

The probability that $y_{ik,t+1} = x_{il}$ after cross-subsystem migration can be calculated based on Equation (2.12), and is denoted as $P_{ikl}^{(2)}(v)$.

$$
P_{ikl}^{(2)}(v) = \text{Pr}(y_{ik,t+1} = x_{il}) = \prod_{s=1}^{q} \left[ (1 - \lambda_{iz(k)})I_0(x_{iz(k)}(s) - x_{il}(s)) + \lambda_{iz(k)} \sum_{j \in J} V_{mj} \sigma_{ilmj} \sum_{j=1}^{n_m} V_{mj} \sigma_{ilmj} \right]
$$

(2.13)

**Combined Within-Subsystem and Cross-Subsystem Migration**

Recall that Section 2.1 provides the detailed BBO/Complex procedure. According to this procedure, there are three steps in modifying a population, with the sequence given as follows: within-subsystem migration, cross-subsystem migration, and mutation. To find the total probability of obtaining a given island, we need to combine the probabilities of those three processes. Based on our derivations up to this point, we combine the probabilities of the two types of migration, and we use $P_{ikl}^{(3)}(v)$ to denote this probability.

$$
P_{ikl}^{(3)}(v) = \sum_{j=1}^{n} P_{ikj}^{(1)}(v)P_{ijl}^{(2)}(v)
$$

(2.14)

This is the probability that $y_{ik}$ is equal to $x_{il}$ after both within-subsystem and
cross-subsystem migration have been considered.

### 2.2.2 Mutation

Mutation is another way to alter islands in BBO/Complex. In this section, we obtain the probability of transforming a given island to another given island due to mutation, and then we combine this probability with Equation (2.14) in order to obtain the transition matrix for the Markov model of BBO/Complex.

Assuming that the mutation rate is predefined and constant, we can easily create a mutation matrix for each subsystem. When we denote the mutation matrix as \( U_i \) for subsystem \( i \), the mutation probability that island \( x_{ir} \) mutates to island \( x_{il} \) is represented by \( U_{irl} \) which is the \( l \)-th element in the \( r \)-th row in the mutation matrix \( U_i \). So the size of \( U_i \) is \( n_i \times n_i \). We next combine \( U_i \) with Equation (2.14) to obtain the probability that \( y_{ik,t+1} = x_{il} \) after within-subsystem migration, cross-subsystem migration, and mutation have all been considered:

\[
P_{ik}^{(4)}(v) = \sum_{r=1}^{n_i} \sum_{j=1}^{n_i} P_{ijk}^{(1)}(v)P_{ijr}^{(2)}(v)U_{irl}
\]

(2.15)

Now we will extend the probability from the island level to the population level. Before we do that, there is a term that needs to be introduced – population vector. In BBO/Complex, the population distribution is represented by a population vector. This is best illustrated by example. Assume we have two subsystems with four possible islands in subsystem 1, and four possible islands in subsystem 2. Then the population vector contains eight elements as illustrated in Figure 2.

<table>
<thead>
<tr>
<th>Population vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Island-11</td>
</tr>
</tbody>
</table>

**Figure 2**: Population vector for a system that is comprised of two subsystems, where each subsystem has a search space cardinality of four. The population vector has eight elements. Island-ik represents the number of \( x_{ik} \) individuals in subsystem \( k \).

As an example based on Figure 2, a population vector \([0 0 2 2 3 1 0 0]\) indicates that subsystem 1 contains two island-3 individuals, and 2 island-4 individuals; and subsystem 2 has three island-1 individuals and one island-2 individual.

We follow the method from [9] [20] and use the generalized multinomial theorem to find the probability that population vector \( v \) transitions to population vector \( u \) in subsystem \( i \) after one generation, and we use \( \text{Pr}_i(u/v) \) to denote this probability:
Based on Equation (2.16), we obtain the transition matrix $P_i$ for subsystem $i$. Each element in $P_i$ represents the probability of transitioning from one possible population vector to another. $P_i$ is a $T_i \times T_i$ matrix, where $T_i$ is the total number of possible population vectors in subsystem $i$. $T_i$ can be calculated as follows [9]:

$$T_i = \binom{n_i + N_i - 1}{N_i}$$

(2.17)

Note that there are $T_i \times T_i$ combinations of $u$ and $v$ vectors in Equation (2.16). These $T_i \times T_i$ different probabilities comprise the entries of the $P_i$ transition matrix. After obtaining the transition matrices of each subsystem, we combine the matrices to form the transition matrix $P$ for the entire system. The size of $P$ is $T \times T$, where $T$ is the total number of possible population vectors for the entire system:

$$T = \prod_{i=1}^{M} T_i$$

(2.18)

where $M$ is the number of subsystems in the entire complex system. The $P$ matrix can be calculated in pseudo-code as follows:

```
FOR (z_1 = 0; z_1 < t) {
    SET Count = 0;
    FOR (z_2 = 0; z_2 < t) {
        FOR (z_3 = 0; z_3 < t) {
            : 
        }
        FOR (z_M+1 = 0; z_M+1 < t) {
            P(Count, z_1) = P_1(z_2, z_1)P_2(z_3, z_1)...P_y(z_M+1, z_1);
            Count++;
        }
    }
}
```

M: number of subsystems

$P_1: (i,j)$: element in $i$-th row and $j$-th column of the transition matrix of subsystem $i$
After calculating transition matrix $P$, the probability of each possible population, in the limit as the generation count approaches infinity, can be calculated based on Equation (2.1).

### 3. Simulation Results

In the previous section, we introduced a method to calculate the limiting probability of each possible population, which exactly predicts the steady state probability of each population vector during BBO/Complex. In this section we use a sample problem to confirm the newly derived Markov model.

The sample problem is a complex system which has two subsystems. Each subsystem contains two bits. Subsystem 1 contains type-1 and type-2 bits; subsystem 2 contains type-1 and type-3 bits. The subsystems share type-1 bits in common. The possible islands of subsystem 1 and subsystem 2 in a unified format are shown in Tables 1 and 2 (see Equation 2.6).

<table>
<thead>
<tr>
<th>Possible island 1</th>
<th>Type-1 bit</th>
<th>Type-2 bit</th>
<th>Type-3 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>Possible island 2</td>
<td>0</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>Possible island 3</td>
<td>1</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>Possible island 4</td>
<td>1</td>
<td>1</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Table 1: Possible islands of subsystem 1**

<table>
<thead>
<tr>
<th>Possible island 1</th>
<th>Type-1 bit</th>
<th>Type-2 bit</th>
<th>Type-3 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>N/A</td>
<td>0</td>
</tr>
<tr>
<td>Possible island 2</td>
<td>0</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>Possible island 3</td>
<td>1</td>
<td>N/A</td>
<td>0</td>
</tr>
<tr>
<td>Possible island 4</td>
<td>1</td>
<td>N/A</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 2: Possible islands of subsystem 2**

Each subsystem includes two cost functions. A smaller cost means better performance. The cost functions for subsystem 1 are given as follows:
\[ y_{11} = 2x_{11} + x_{12} + 1 \]
\[ y_{12} = \frac{y_{11}}{x_{11} + x_{12} + 1} + 1 \] (3.1)

- \( y_{11} \): first cost value of an island in subsystem 1
- \( y_{12} \): second cost value of an island in subsystem 1
- \( x_{11} \): first bit of an island in subsystem 1
- \( x_{12} \): second bit of an island in subsystem 1

The cost functions for subsystem 2 are given as follows:
\[ y_{21} = 2x_{21} + x_{23} + 1 \]
\[ y_{22} = \frac{x_{21} + x_{23} + 1}{y_{21} + 1} + 1 \] (3.2)

- \( y_{21} \): first cost value of an island in subsystem 2
- \( y_{22} \): second cost value of an island in subsystem 2
- \( x_{21} \): first bit of an island in subsystem 2
- \( x_{23} \): third bit of an island in subsystem 2

In order to verify the BBO/Complex Markov model derived in the previous section, we have two requirements for the simulation setup. First, we need to perform Monte Carlo simulations of BBO/Complex to obtain average performance. Second, the generation limit of each BBO/Complex run should be large enough that the simulation results converge to steady state values. The simulation setup in this paper is shown as follows:

- Monte Carlo simulations: 100
- BBO/Complex generations for each Monte Carlo simulation: 5000
- Number of subsystems: 2
- Number of islands per subsystem (population size): 4
- Number of bits per island: 3

We optimize this sample problem with three different mutation rates in BBO/Complex: 0.001, 0.01, and 0.1.
Based on the cost functions for each subsystem and the non-dominated ranking system, the optimal population vector is \([4 0 0 0 4 0 0 0]\). According to the results shown in Table 3, when the mutation rate is 0.001, the probability of obtaining the optimal population vector calculated by the Markov model is 0.9489, and the probability calculated by the simulation is 0.9590. This confirms that the optimal population vector dominates other populations, and we have a high probability of obtaining it. Also, the simulation results match the theoretical results well.

When the mutation rate is 0.01, the most probable population vector is still the optimal one, but the probability of the optimal population vector falls to around 60%. Although performance is degraded, the probabilities calculated by the Markov model and by the simulation are still close.

When the mutation rate is 0.1, the most probable population vector is \([3 1 0 0 4 0 0 0]\), which is not the optimal population vector. The optimal population vector \([4 0 0 0 4 0 0 0]\) is only the 7th most likely according to the Markov model, and the 5th most likely according to the simulation (not shown in Table 3). Since the probability values are relatively small for the top five population vectors, the differences between the Markov model results and the simulation results are larger compared to when the mutation rate is lower, but the differences between theory and

<table>
<thead>
<tr>
<th>Mutation Rate</th>
<th>Population Vector</th>
<th>Probability</th>
<th>Markov</th>
<th>Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.001</td>
<td>4 0 0 0 4 0 0 0</td>
<td>0.9489</td>
<td>0.9590</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 1 0 0 4 0 0 0</td>
<td>0.0287</td>
<td>0.0194</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 0 0 0 3 1 0 0</td>
<td>0.0105</td>
<td>0.0074</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 0 0 0 3 0 1 0</td>
<td>0.0060</td>
<td>0.0070</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 0 1 0 4 0 0 0</td>
<td>0.0044</td>
<td>0.0058</td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>4 0 0 0 4 0 0 0</td>
<td>0.6051</td>
<td>0.5901</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 1 0 0 4 0 0 0</td>
<td>0.1655</td>
<td>0.1770</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 0 0 0 3 1 0 0</td>
<td>0.0647</td>
<td>0.0631</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 0 0 0 3 0 1 0</td>
<td>0.0385</td>
<td>0.0425</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 0 1 0 4 0 0 0</td>
<td>0.0284</td>
<td>0.0294</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>3 1 0 0 4 0 0 0</td>
<td>0.0425</td>
<td>0.0348</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 1 0 0 3 1 0 0</td>
<td>0.0371</td>
<td>0.0268</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 2 0 0 4 0 0 0</td>
<td>0.0329</td>
<td>0.0274</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 2 0 0 3 1 0 0</td>
<td>0.0287</td>
<td>0.0218</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 1 0 0 3 0 1 0</td>
<td>0.0278</td>
<td>0.0258</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: The five most likely populations for three mutation rates.
simulation are still small.

Based on Table 3, the Markov model is verified by the simulation results. Finally, note that the calculation time for the Markov model probabilities was 492 seconds, but the average calculation time of each simulation was 1166 seconds. In this case, the Markov model not only obtained more accurate steady-state results than the simulation, but does so in less computational time.

4. Conclusion

Heuristic algorithms can be helpful tools for optimization, especially when the optimization problem is complex in structure and interaction. But unlike traditional optimization methods, the performance of heuristic algorithms is nondeterministic, stochastic, and unpredictable before implementation. This is one of the reasons industry has tended to resist the use of heuristic algorithms, despite their empirical success in many areas. But with a well-developed mathematical model, we can exactly calculate the steady state performance of certain heuristic algorithms. Markov models can be useful tools to predict and evaluate the performance of heuristic algorithms, even before implementation.

BBO/Complex is a heuristic algorithm designed for complex systems. Such systems contain multiple subsystems, and each of the subsystems may contain multi-objectives and multi-constraints. Compared to the original BBO algorithm, BBO/Complex has a more complex structure that is better suited for real world problems.

In this paper, a Markov model was derived for BBO/Complex, and it was confirmed by a bi-subsystem sample problem. When the mutation was low – 0.001 or 0.01 – the optimal vector dominated the population with a probability of around 95% and 60% respectively. But with a high mutation rate of 0.1, the probability of obtaining the optimal population vector was only around 2.7%. Although the population probabilities are different with different mutation rates, the theoretical results calculated by the Markov model and the simulations are well matched, thus confirming the Markov model.

According to our results, the computational requirements of the Markov model can be much less than that of the simulation for small problems. Markov models are useful for predicting the performance of heuristic algorithms, and quantifying the performance of different components in a heuristic algorithm without relying on long time simulation times. Markov models can thus be helpful for algorithm design and parameter tuning.
But Markov models also have a disadvantage. The computational effort can be very high for large problems. For a complex system with \( M \) subsystems, the total number of possible populations is

\[
T = \prod_{i=1}^{M} \left( \frac{n_i + N_i - 1}{N_i} \right) \tag{4.1}
\]

\( n_i \): cardinality of search space in subsystem \( i \)

\( N_i \): population size of subsystem \( i \)

Based on this equation, the total number of possible populations in our small sample system is 1,225. When we have a larger population size or a non-binary problem, this number will increase to an extremely large number that will result in a large transition matrix that cannot be easily handled.

We envision two directions for future research. First, we want to design a method with which to save computational effort during Markov model development. Because of the heavy computational burden mentioned above, Markov models are limited to problems with small population sizes and binary island structures, which do not capture the structure of real world problems. This limitation might be able to partially addressed by combining similar Markov model states into a single state [21].

Second, as mentioned at the beginning of this paper, complex systems typically contain multiple subsystems, multiple objectives, and multiple constraints. In this paper, a Markov model is developed for complex systems with multiple subsystems and multiple objectives. In future research, a Markov model can be developed that for complex systems that also include multiple constraints.

Finally, the MATLAB software that we used to generate the Markov model and simulation results in this paper can be downloaded from [22].

**Acknowledgments**

This material is based on work supported by the National Science Foundation under Grant no. 0826124 and by a Dissertation Research Award from Cleveland State University.
References


