1999

Markov chain models of genetic algorithms

Yong Zhao

The University of Montana

Let us know how access to this document benefits you.
Follow this and additional works at: https://scholarworks.umt.edu/etd

Recommended Citation
Zhao, Yong, "Markov chain models of genetic algorithms" (1999). Graduate Student Theses, Dissertations, & Professional Papers. 8350.
https://scholarworks.umt.edu/etd/8350

This Thesis is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact scholarworks@mso.umt.edu.
The University of MONTANA

Permission is granted by the author to reproduce this material in its entirety, provided that this material is used for scholarly purposes and is properly cited in published works and reports.

** Please check "Yes" or "No" and provide signature **

Yes, I grant permission [ ]
No, I do not grant permission [ ]

Author's Signature [Signature]

Date [May 20, 1999]

Any copying for commercial purposes or financial gain may be undertaken only with the author's explicit consent.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Markov Chain Models of Genetic Algorithms

by

Yong Zhao

Presented in partial fulfillment of the requirements
for the degree of
Master of Science
in Computer Science

The University of Montana-Missoula

July 1999

Approved by:

Chairperson

Dean, Graduate School

5-20-99

Date
Markov chain models of genetic algorithms have been an active research area since early 1990's. In 1990, Vose [Vose, 1990] provided an infinite population model for a genetic algorithm called the Simple Genetic Algorithm (with proportional selection, mutation determined by a mutation rate, and one-point crossover). In 1992, Nix and Vose [Nix and Vose, 1992] derived a finite population model for the same algorithm. Thereafter, Vose and others refined their work and came up with a more generalized model, Random Heuristic Search. In this work, we start with an overview of the Random Heuristic Search model and its applications to the Simple Genetic Algorithm. We prove three new results regarding the mixing scheme of Random Heuristic Search. We will conduct an extended work on some selection strategies that Random Heuristic Search does not include. Then we will integrate our work with Random Heuristic Search to construct the Markov chain models of several practical genetic algorithms. Finally, we will present some experimental results based on our models.
# TABLE OF CONTENTS

ABSTRACT .................................................................................................................... ii

ACKNOWLEDGMENTS ................................................................................................. v

1 Introduction ................................................................................................................ 1

2 Notation and Terminology .......................................................................................... 2

2.1 Algebra .................................................................................................................... 2
2.2 Populations .............................................................................................................. 4
2.3 The multiple hypergeometric distribution ................................................... 6

3 Random Heuristic Search and The Simple Genetic Algorithm ....................... 8

3.1 Random Heuristic Search ................................................................................ 9
3.2 The Simple Genetic Algorithm ...................................................................... 12
  3.2.1 Selection .................................................................................................... 12
  3.2.2 Mutation ................................................................................................. 14
  3.2.3 Crossover ................................................................................................. 15
  3.2.4 Mixing ..................................................................................................... 18
  3.2.5 The Heuristic function of the Simple Genetic Algorithm ................ 22

4 An extended work on selection .............................................................................. 24

  4.1 Step 3 done by sampling with replacement ................................................... 25
  4.2 Step 3 done without replacement ................................................................ 26
    4.2.1 Step 3 done by truncation selection from X .................................. 28
    4.2.2 Step 3 done by truncation selection from X+Y ......................... 30

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ACKNOWLEDGMENTS

This work is dedicated to my mother, Menghui Liu. Her support and encouragement helped me achieve one goal after another in my life. I would like to thank my advisor Dr. Wright. I took my first computer science course from him two years ago. It was in his class where I first learned the theoretical aspect of computer science besides its programming part. I am glad to work on my thesis under his direction. His comprehensive knowledge and never-rest ideas helped me understand the subject and push my work from one level to another. I would like to thank Dr. Kayll, my math advisor and friend. I own him for all the grammar, spelling flaws as well as mathematical mistakes I made in the draft of this work. Also I would like to thank Dr. Opitz. I first learned about genetic algorithm from his lectures. Part of this work was actually originated from my course project in his class. I am grateful to Diane Oman for her assistance in the experimental phase of this work. I am in debt to Dr. Murray, Dr. Fleming and Dr. Hewitt for my whole life. Without their help, I would probably be working on my bureaucratic job in China today. I thank Mei Zhou, my colleague and my wife. I find it is much more helpful to check her for some questions rather than look it up in a scientific reference. Finally, I thank our parents for the support.
CHAPTER I
INTRODUCTION

First developed by Holland [Holland, 1975] and others in the 1970's, genetic algorithms have been widely used to search over large, irregular and poorly understood spaces. With so many varieties used in practice, it is an active research area to compare different genetic algorithms both empirically and theoretically. One common approach of the research is to obtain some empirical results first and then apply an analysis based on Holland's Schema Theorem to gain some insight. However, this insight is limited in the sense that it can loosely predict the generational changes of the building blocks (i.e. schemata) under certain conditions. In order to gain a more complete insight of the genetic algorithms, many researchers have been working on the Markov chain models of the genetic algorithms since the late 1980's. Most genetic algorithms can be modeled as Markov chains over populations. The most commonly used Markov chain model is an exact model whereas the Schema Theorem model is an approximate model. Also, once the Markov chain model is constructed, it is just a matter of computation to get the stationary distributions for ergodic chains and absorption probabilities for absorbing chains, which can provide a solid ground for algorithm comparisons and theoretical research. In this work, we first review a well-researched Markov chain model, namely the Random Heuristic Search model, and obtain three generalized results based on the model. Then we will construct Markov chain models for several practical genetic algorithms. Finally, we will derive average absorption time for these algorithms based on their Markov chain models.
CHAPTER II
NOTATION AND TERMINOLOGY


2.1 Algebra

Let $\mathbb{Z}$ be the set of integers, $\mathbb{N}$ be the set of non-negative integers (i.e. $\mathbb{N} = \{0, 1, 2, \ldots\}$), and $\mathbb{R}$ be the set of real numbers. Let $\Omega$ be the set of length-$l$ binary strings and $n$ be the cardinality of $\Omega$. Clearly, $n = 2^l$. Let $\mathbb{Z}_2$ be the set of integers modulo 2 (i.e. $\mathbb{Z}_2 = \{0, 1\}$). Algebraically, $\mathbb{Z}_2$ is a finite field with addition $\oplus$ and multiplication $\otimes$ defined by

\[
\begin{array}{c|cc}
\oplus & 0 & 1 \\
\hline
0 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\quad
\begin{array}{c|cc}
\otimes & 0 & 1 \\
\hline
0 & 0 & 0 \\
1 & 0 & 1 \\
\end{array}
\]

The set $\Omega$ can be represented by $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \ldots \times \mathbb{Z}_2$, or $(\mathbb{Z}_2)^l$ for simplicity. Under bitwise addition $\oplus$ and multiplication $\otimes$, $\Omega$ retains most algebraic properties of $\mathbb{Z}_2$ except the existence of multiplicative inverses. Each binary string of $\Omega$ can be represented by an integer from $[0, n - 1]$. For the rest of this work, we interchangeably use $\Omega$ to represent both $(\mathbb{Z}_2)^l$ and integers from $[0, n - 1]$. We assume that the elements of $\Omega$ are ordered according to the usual ordering of $[0, n - 1]$. For example, if $l = 2$, then $n = 2^2$ and

$$\Omega = \{00, 01, 10, 11\} \equiv \{0, 1, 2, 3\}.$$
For a binary string $i \in \Omega$, let $|i|$ denote the number of 1's in $i$. In the above example,

$$|00| = 0, |01| = 1, |10| = 1, \text{ and } |11| = 2.$$ 

If $expr$ is an expression that has a value of true or false, then

$$[expr] = \begin{cases} 
1 & \text{if } expr \text{ is true} \\
0 & \text{otherwise.} 
\end{cases}$$

We call $[expr]$ the logical bracket of $expr$. We use $\mathbf{1}$ to denote the vector of all 1's and $\mathbf{0}$ to denote the vector of all 0's. For $k \in \Omega$, $\sigma_k$ is the matrix such that

$$[\sigma_k]_{i,j} = [i \oplus j = k], \text{ for all } i, j \in \Omega.$$ 

For example, if $\Omega = \{00, 01, 10, 11\}$ and $k = 01$, then

$$\sigma_k = \begin{pmatrix}
00 \oplus 00 = 01 & 00 \oplus 01 = 01 & 00 \oplus 10 = 01 & 00 \oplus 11 = 01 \\
01 \oplus 00 = 01 & 01 \oplus 01 = 01 & 01 \oplus 10 = 01 & 01 \oplus 11 = 01 \\
10 \oplus 00 = 01 & 10 \oplus 01 = 01 & 10 \oplus 10 = 01 & 10 \oplus 11 = 01 \\
11 \oplus 00 = 01 & 11 \oplus 01 = 01 & 11 \oplus 10 = 01 & 11 \oplus 11 = 01
\end{pmatrix} = \begin{pmatrix}
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{pmatrix}.$$ 

Because the operator $\oplus$ is commutive, $\sigma_k$ is symmetric. One can verify that

$$\sigma_k [x_0, \ldots, x_{n-1}]^T = [x_{0 \oplus k}, \ldots, x_{(n-1) \oplus k}]^T.$$
Therefore, $\sigma_k$ is sometimes called a permutation matrix. One can also check that

$$\sigma_i \sigma_j = \sigma_{i \sigma j}.$$ 

2.2 Populations

A multiset is a set with repeated elements (e.g. \{1, 1, 2\} is a multiset with three elements). In this work, we do not try to distinguish between set and multiset. Our definitions apply to both of them. For a set $S$, we use $|S|$ to denote its cardinality. A population $P$ is a multiset with elements drawn from $\Omega$. In this work, we are concerned mainly with finite populations. In case of an infinite population, we will point it out explicitly. We use $P_r$ to denote a population of size $r$. When there is no ambiguity, we often omit the subscript $r$. With a dot-bar representation, which can be found in many discrete mathematics books such as [Gersting, 1993], it can be shown that the total number of size-$r$ populations drawn from $\Omega$ is given by

$$N = \binom{n + r - 1}{r}.$$ 

We denote a population by an incidence vector from $\mathcal{N}^n = \mathcal{N} \times \mathcal{N} \times \ldots \times \mathcal{N}$. If $X$ is a size-$r$ population represented by an incidence vector, then its $i$th entry $X_i$ represents the number of appearances of $i \in \Omega$. Clearly, $\sum_{i \in \Omega} X_i = r$. For two populations $X$ and $Y$, we say that $X \leq Y$ if $X_i \leq Y_i$ for all $i \in \Omega$. For example, if $X = [0, 1, 0, 2]$ and $Y = [1, 1, 0, 3]$, then $X \leq Y$. For two populations $X$ and $Y$, $Y - X$ denotes the population $Z$ with $Z_i = [Y_i - X_i \geq 0](Y_i - X_i)$, where the first factor is a logical bracket. For example, if $X = [2, 1, 0, 2]$ and $Y = [1, 1, 0, 3]$, then $Y - X = [0, 0, 0, 1]$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
For a population $X$ of size $r$, the vector $X/r$ gives a probability distribution over $\Omega$. The $i$th entry of the probability distribution $(X/r)_i$ represents the frequency of $i \in \Omega$ in population $X$. Vose and Liepins [Vose and Liepins, 1991] used these probability distributions to denote populations. In this case, a population is an element from

$$\Lambda = \left\{ [x_0, \ldots, x_{n-1}] \in \mathcal{R}^n : \sum x_i = 1 \text{ and } x_j \geq 0 \text{ for all } 0 \leq j \leq n-1 \right\}.$$

For $p \in \Lambda$, its $i$th entry $p_i$ can also be interpreted as the probability that $i \in \Omega$ will be selected when an element is chosen from $\Omega$. Clearly, this representation is population-size-independent, because each $p \in \Lambda$ represents an infinite number of finite populations. For example, populations $[0,1,0,1]$ and $[0,2,0,2]$ have the same probability distribution representation $[0,1/2,0,1/2]$. In this work, we use both the incidence vector and the probability distribution representation of populations. In the former case, we call it a population, while in the latter one, we call it a population probability distribution. We use capital letters to denote populations and lower-case letters to denote their probability distributions. For a population $X$ of size $r$, its population probability distribution is given by $X/r$. For example, for two populations $X = [0,2,0,2]$ and $Y = [1,1,0,3]$ over $\Omega = \{0,1,2,3\}$, the corresponding population probability distributions are given by $x = [0,0.5,0,0.5]$ and $y = [0.2,0.2,0,0.6]$.

For a set $S$, we use $\mathcal{P}_r(S)$ to denote the set of size-$r$ subsets of $S$. For a population $X$, $\mathcal{P}_r(X)$ represents the set of size-$r$ subpopulations of $X$. Similarly, $\mathcal{P}_r(\Omega)$ is the set of all size-$r$ populations over $\Omega$. In this case, we abbreviate $\mathcal{P}_r(\Omega)$ by $\mathcal{P}_r$ for simplicity. Recall that $|\mathcal{P}_r| = N = \binom{n+r-1}{r}$.
2.3 The multiple hypergeometric distribution

An interesting problem that appears in most probability books arises when, given 5 red balls, 4 blue balls and 3 green balls, one is asked to find the probability of choosing one ball of each color when 3 balls are selected uniformly at random without replacement. This probability is given by

\[
\frac{\binom{5}{1} \binom{4}{1} \binom{3}{1}}{\binom{12}{3}}.
\]

When we enumerate all the samples having 3 balls, such as 3 red, 0 blue and 0 green, or 2 red, 1 blue, and 0 green, etc., we get a probability distribution. This is a simple example of the *multiple hypergeometric distribution*. For a population \( X \) of size \( r \), the multiple hypergeometric distribution describes the probabilities of choosing subpopulations from \( X \). Let \( W \) of size \( k \) (\( k \leq r \)) be a subpopulation of \( X \). The probability of selecting \( W \) from \( X \) is given by

\[
\rho_X(W) = \frac{\prod_{j \in \Omega} \binom{x_j}{w_j}}{\binom{r}{k}}.
\]

Wright and Zhao [Wright and Zhao, 1999] first applied the multiple hypergeometric distribution to model selection without replacement in genetic algorithms. Revisit our example at the beginning of this section. The probability of obtaining 2 red balls under the same context is given by

\[
\frac{\binom{5}{2} \binom{4}{1} \binom{3}{0} \binom{12}{3} + \binom{5}{2} \binom{4}{0} \binom{3}{1} \binom{12}{3}}{\binom{12}{3}}.
\]
and the conditional probability of obtaining 2 red balls and 1 blue ball, given that 2 red balls are selected, is given by

\[
\frac{\binom{2}{2} \binom{1}{1} \binom{3}{1}}{\binom{5}{2}} + \frac{\binom{2}{2} \binom{1}{1} \binom{3}{1}}{\binom{5}{2}}.
\]

If \( B \) is a set of subpopulations of population \( X \), i.e. members of \( B \) are populations whose elements are selected without replacement from \( X \),

\[
\rho_X (B) = \sum_{W \in B} \rho_X (W).
\]

For \( W \in B \), the conditional probability of choosing \( W \) from \( X \) given that one element (i.e. a subpopulation) of \( B \) is selected is given by

\[
\frac{\rho_X (W)}{\rho_X (B)}.
\]
CHAPTER III
RANDOM HEURISTIC SEARCH AND THE SIMPLE GENETIC ALGORITHM

In a 1990 paper [Vose, 1990], Vose introduced an infinite population model for a genetic algorithm with proportional selection, mutation determined by a mutation rate, and one-point crossover. In the next year, Vose and Liepins [Vose and Liepins, 1991] refined and formalized the model mathematically and applied the model to the Simple Genetic Algorithm based on infinite populations. In 1992, Nix and Vose [Nix and Vose, 1992] described a Markov chain model of the Simple Genetic Algorithm for finite populations. The Simple Genetic Algorithm is described by the following steps.

1. Given an initial population $X$ of size $r$;
2. Let $Y$ be empty;
3. Select two parents based on fitness;
4. Apply crossover to the parents to obtain a child;  
5. Apply mutation to the child;
6. Add child to population $Y$;
7. Repeat step 3 through 6 for $r$ times;
8. Replace $X$ with $Y$;

Thereafter, Vose and Wright ([Vose and Wright, 1994], [Vose, 1999]) abstracted this work into a much more general model called Random Heuristic Search. Random
Heuristic Search can be described by

1. Given an initial population $X$ of size $r$;
2. Let $p = \mathcal{G}(X/r)$ (see below for $\mathcal{G}$);
3. Select $r$ independent samples with the probability distribution $p$ forming population $Y$;
4. Replace $X$ with $Y$;
5. Goto step 2.

Here $\mathcal{G} : \Lambda \rightarrow \Lambda$ is a heuristic function defined such that, for $p \in \Lambda$, $\mathcal{G}(p)_i$ is the probability that $i \in \Omega$ will be selected into the next generation. In other words, for a given population, the function $\mathcal{G}$ gives a probability distribution over $\Omega$. We rely on the Random Heuristic Search model for part of our work. This chapter is a brief excursion through Vose’s work on Random Heuristic Search and the Simple Genetic Algorithm. Most material presented is summarized from [Nix and Vose, 1992], [Vose and Wright, 1994] and [Vose, 1999] with the exception of the definitions of the mutation scheme and the crossover scheme, two-point crossover heuristic function and theorems 5, 6 and 7.

### 3.1 Random Heuristic Search

The Random Heuristic Search model provides a rigorous method for modeling genetic algorithms as Markov chain processes. A Markov chain is a discrete-time stochastic process defined over a set of states by a transition matrix $Q$, where $Q_{i,j}$
corresponds to the \textit{transition probability} that the process will enter state $j$, given that the current state is $i$. One important feature of a Markov chain is the Markov (memoryless) property, i.e. the future behavior depends only on its current state, and not on how it arrived at that state. We will use Markov chain notation extensively in this work. However, we assume a general familiarity with this subject. See [Isaacson and Madsen, 1976] for more details.

The Random Heuristic Search model is a Markov chain process, where the states are the populations. From algorithm 3.2, we can see that each transition step consists of two substeps: calculating the probability distribution $\mathcal{G}(X/r)$ based on current population $X$ at step 2, then forming a new population by independent samplings with the probability distribution $\mathcal{G}(X/r)$ at step 3. We leave the discussion about $\mathcal{G}$ function to section 3.2 and focus on the modeling of step 3 of algorithm 3.2 in this section. Let $p = \mathcal{G}(X/r) \in \Lambda$ be a probability distribution over $\Omega$ and $Y$ be a size-$r$ population drawn from $\Omega$. We want to compute the probability that $Y$ is obtained given the probability distribution $p$. Recall that $Y_i$ is the number of appearances of $i \in \Omega$ in $Y$ and $p_i$ is the probability that the $i \in \Omega$ is selected into the next generation.

First, let's calculate the probability of selecting the first element (i.e. 0) of $\Omega$ $Y_0$ times. We can imagine this process as selecting $Y_0$ positions out of a total of $r$ positions then put 0's in them. There are

$$\binom{r}{Y_0}$$

ways of selecting these $Y_0$ positions. For each of these, the probability of putting 0's
in the selected positions is \( p_o^{Y_0} \). After \( Y_0 \) 0's have being selected, we proceed to choose \( Y_1 \) positions out of the remaining \( r - Y_0 \) positions and put 1's in them. Inductively, we find that the probability that \( Y \) is obtained after \( r \) independent samples with probability distribution \( p \) is given by

\[
\binom{r}{Y_0} \binom{r - Y_0}{Y_1} \cdots \binom{r - Y_0 - \cdots - Y_{n-2}}{Y_{n-1}} (p_o)^{Y_0} (p_1)^{Y_1} \cdots (p_{n-1})^{Y_{n-1}},
\]

which can be simplified to

\[
\frac{r! \prod_{i \in \Omega} p_i^{Y_i}}{(Y_i)!}.
\]

Substituting \( \mathcal{G}(X/r) \) for \( p \) in the formula above, we obtain

\[
\frac{r! \prod_{i \in \Omega} \mathcal{G}(X/r)^{Y_i}}{(Y_i)!},
\]

which is the transition probability from population(state) \( X \) to population(state) \( Y \) in algorithm 3.2.

The Random Heuristic Search model can be used to describe a wide range of search methods with various levels of detail. Modeling genetic algorithms is just one of its applications. A full investigation is beyond the scope of this work. See [Vose, 1998] for a detailed discussion. When the Random Heuristic Search model is used to describe the Simple Genetic Algorithm, the heuristic function \( \mathcal{G} \) is used to encapsulate the selection, mutation and crossover genetic operations during the recombination stage.
3.2 The Simple Genetic Algorithm

Compare algorithm 3.1 and algorithm 3.2. If steps 3 through 5 of algorithm 3.1 can be encapsulated into step 2 of algorithm 3.2, then algorithm 3.1 will become a special case of the Random Heuristic Search model. Our goal is to describe steps 3 through 5 of algorithm 3.1 as a heuristic function \( G \). Vose and Liepins gave this model in their joint paper [Vose and Liepins, 1991]. The same model was used and refined in [Nix and Vose, 1992] and [Vose, 1999]. The discussion we presented in this section mostly summarizes from [Vose, 1999]. The basis of Simple Genetic Algorithm model is the successful modeling of three genetic operations, namely selection, mutation and crossover.

3.2.1 Selection

In genetic algorithms, the selection operation consists of selecting a member from a population based on some criteria. Depending on the algorithms, the selection can be random, i.e. each member has equal chance of being selected, or biased, i.e. some members have higher chance of being selected. A selection scheme \( F: \Lambda \to \Lambda \) is a heuristic function such that, for \( p \in \Lambda \), \( F(p) \) is the probability that \( i \in \Omega \) will be selected for the next generation. For a population \( X \) of size \( r \), \( F(X/r) \) gives a probability distribution for the selection operation.

With well-defined selection schemes, one can model a variety of selections used in genetic algorithms. We show some example schemes. For random selection, \( F \) is the
identity function where an element will be selected with a probability in proportion to its frequency in the current population. In most genetic searches, the selection operation is biased toward individuals with higher fitness. In this case, the function $F$ is related to the underlying fitness function over $\Omega$. Two commonly used selection schemes are proportional selection and ranking selection. Let $f$ be a vector of dimension $n$ (i.e. the size of $\Omega$) such that $f_i$ is the fitness of the $i \in \Omega$. Let $X$ be a population of size $r$. Let $x_i$ be the $i$th entry of the population probability distribution $X/r$. The proportional selection scheme is given by

$$F(x)_i = \frac{f_i x_i}{\sum_{j \in \Omega} f_j x_j}.$$

Note that when $f = 1$, $F$ becomes the identity function. Proportional selection is used in genetic algorithms to select candidates for the next generation based on each element's proportion in the current population and its fitness value. However, one problem with this scheme is that, when the proportion of higher fitness individuals increases, there may not be enough selection pressure to favor the highest fitness individual(s). For example, given a population probability distribution $x = [0.4, 0.3, 0.2, 0.1]$ with fitness vector $f = [9, 10, 3, 2]$, we hope the second element can have a better chance of being selected because of its higher fitness value. However, with proportional selection, we find that $F(x) = [0.49, 0.41, 0.08, 0.02]$, which indicates that the selection is biased toward the first element instead of the second one. To overcome this deficiency, many genetic algorithms use another selection strategy, ranking selection,
whose selection scheme is described by

\[ \mathcal{F}(x)_i = \int \sum_{[f_j \leq f_i]x_j} \varphi(y) \, dy \]

where \( \varphi \) is any increasing function over \([0, 1]\). Taking the same example, with \( \varphi = y/2 \), we have

\[
\begin{array}{ccc}
   i & \sum_j [f_j < f_i] x_j & \mathcal{F}(x)_i \\
   0 & 0.3 & 0.40 \\
   1 & 0.7 & 0.51 \\
   2 & 0.1 & 0.08 \\
   3 & 0 & 0.01 \\
\end{array}
\]

wherein the selection is biased toward the highest fitness member as we desired. From another perspective, ranking selection is used to normalize the fitness values of each element in order to create selection pressure to favor the highest fitness individual(s).

3.2.2 Mutation

For a binary string, the mutation operation randomly flips some bits of the string. This operator is usually used to introduce new genes into the current population. Theoretically, for two binary strings \( i, j \in \Omega \), the operation of mutating \( j \) to \( j \oplus i \) can be regarded as applying a mutation mask \( i \) to \( j \) by \( \oplus \) operator. Note that the \( k \)th bit of \( j \) is mutated if and only if the \( k \)th bit of the mutation mask \( i \) is 1. A mutation rate \( \mu \), i.e. the probability that each bit of a binary string being mutated, is usually associated with mutation operation. From another perspective, the mutation rate \( \mu \) can also be interpreted as a probability distribution over \( \Omega \) such that \( \mu_i \) (the \( i \)th entry
of the distribution) describes the probability of \( i \in \Omega \) being selected as the mutation mask. In this case, the probability distribution is given by

\[
\mu_i = (\mu)^{|i|} (1 - \mu)^{|i| - |i|},
\]

where \( \mu \) is the mutation rate and \(|i|\) denotes the number of 1's in mask \( i \). With the binomial theorem, it is easy to check that \( \sum_{i \in \Omega} \mu_i = 1 \). We interchangeably use \( \mu \) to denote the mutation rate and the probability distribution it defines. We say a mutation \( \mu \) (denoting a probability distribution) is independent if for all \( j, k \in \Omega \)

\[
\mu_j = \sum_{k \otimes i = 0} \mu_{i \otimes j} \sum_{k \otimes i = 0} \mu_{i \otimes j}.
\]

Vose proved following theorem in [Vose, 1999].

**Theorem 1** (Vose) If mutation probability distribution \( \mu \) is determined by a mutation rate, then \( \mu \) is independent.

A mutation scheme is a heuristic function \( U : \Lambda \rightarrow \Lambda \) such that, for a population probability distribution \( x \in \Lambda \),

\[
U(x)_i = \sum_{u \in \Omega} x_u \mu_{u \otimes i},
\]

where \( U(x)_i \) is the probability that \( i \in \Omega \) will be produced by the mutation operation.

### 3.2.3 Crossover

In genetic algorithms, the crossover operation is used to recombine the structures in the current population. For two binary strings, the crossover operation randomly
selects some bit positions and swaps the bit values at these positions between the two strings, and then keeps one of the two results. Given two strings \( x, y \in \Omega \), the crossover operation can be considered as applying a crossover mask \( i \) to two binary strings \( x, y \) to obtain one child randomly selected from two candidates \((x \otimes i) \oplus (\tilde{i} \otimes y)\) and \((y \otimes i) \oplus (\tilde{i} \otimes x)\). The positions where bit values are to be exchanged are determined by the 1's in the crossover mask. For mask \( 0 \in \Omega \), the corresponding crossover is called zero crossover; no bits are exchanged. There are three commonly used crossover operations, namely one-point, two-point and uniform crossover. We use the following table to illustrate the ideas behind each.

<table>
<thead>
<tr>
<th>Mask ( i )</th>
<th>String ( x )</th>
<th>String ( y )</th>
<th>((x \otimes i) \oplus (\tilde{i} \otimes y))</th>
<th>((y \otimes i) \oplus (\tilde{i} \otimes x))</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-point</td>
<td>001111</td>
<td>110101</td>
<td>011011</td>
<td>010101</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>111011</td>
</tr>
<tr>
<td>Two-point</td>
<td>011100</td>
<td>110101</td>
<td>011011</td>
<td>010111</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>111001</td>
</tr>
<tr>
<td>Uniform</td>
<td>010110</td>
<td>110101</td>
<td>011011</td>
<td>011101</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>110011</td>
</tr>
</tbody>
</table>

The crossover rate is the probability that crossover is used for two given strings. Similar to the mutation rate, the crossover rate \( \chi \) allows one to define a probability distribution \( \chi \) such that its \( i \)th entry \( \chi_i \) denotes the probability that \( i \) will be selected as the crossover mask. (From the context it should be clear whether \( \chi \) denotes a crossover rate or a probability distribution.) For a crossover rate \( \chi \), the probability distribution defined by \( \chi \) is given by

\[
\chi_i = \begin{cases} 
\chi c_i & \text{if } i > 0 \\
1 - \chi + \chi c_0 & \text{if } i = 0,
\end{cases}
\]
where \( c_i \) depends on the crossover type, i.e. one-point, two-point or uniform crossover.

For one-point crossover (assume \( l > 1 \)), only those masks whose 1's and 0's are partitioned into two contiguous blocks can be chosen. Since there are \( l - 1 \) such binary strings, the probability distribution is given by

\[
\chi_i = \begin{cases} 
\frac{\chi}{l-1} & \text{if } i = 2^k - 1, \text{ for some } k \in (0, l) \\
1 - \chi & \text{if } i = 0.
\end{cases}
\]

For two-point crossover (assume \( l > 1 \)), if we consider one-point crossovers be special cases of two-point crossovers, then there are \( \binom{l}{2} \) non-zero masks. An easy way to see this is to imagine \( l \) dots lined up with \( l - 1 \) interpolated spaces and an additional space at either end. We want to put two vertical bars in these \( l \) spaces. Clearly, there are \( \binom{l}{2} \) ways to achieve this. The probability distribution for two-point crossover is given by

\[
\chi_i = \begin{cases} 
\frac{\chi}{\binom{l}{2}} & \text{if } i = (2^k - 1) - (2^h - 1), \text{ for some } k, h \in [0, l) \text{ and } k > h \\
1 - \chi & \text{if } i = 0.
\end{cases}
\]

For uniform crossover, all binary strings of \( \Omega \) have the same chance of being selected, which implies \( c_i = 1/2^l \), or \( \chi 2^{-l} \) for all \( \chi_i \). So the probability distribution for uniform crossover is given by

\[
\chi_i = \begin{cases} 
\chi 2^{-l} & \text{if } i \neq 0 \\
1 - \chi + \chi 2^{-l} & \text{if } i = 0.
\end{cases}
\]

A crossover scheme is a heuristic function \( \mathcal{X} : \Lambda \rightarrow \Lambda \) such that

\[
\mathcal{X}(x) = \sum_{u,v,k \in \Omega} x_u x_v \frac{\chi_k}{2} \left( \left[ (u \otimes k) \oplus (v \otimes k) = i \right] + \left[ (u \otimes k) \oplus (v \otimes k) = i \right] \right)
\]
which can be simplified to

$$\mathcal{X}(x)_i = \sum_{u,v,k \in \Omega} x_u x_v \frac{x_k + x_{\overline{k}}}{2} [(u \otimes k) \oplus (v \otimes \overline{k}) = i].$$

(3.5)

### 3.2.4 Mixing

The combined usage of mutation and crossover is called *mixing*. Let $m_{u,v}(z)$ denote the probability of obtaining $z$ from parents $u$, $v$ by mixing. A mixing matrix $M(z)$ is a matrix such that

$$M_{u,v}(z) = m_{u,v}(z).$$

Following two theorems regarding $m_{x,y}(z)$ are fundamental for the model of the Simple Genetic Algorithm. Vose gave their proofs in [Vose, 1999].

**Theorem 2** (Vose) If mutation is performed before crossover, then

$$m_{u,v}(z) = \sum_{i,j,k \in \Omega} \mu_i \mu_j \frac{x_k + x_{\overline{k}}}{2} [((u \oplus i) \otimes k) \oplus ((v \oplus j) \otimes \overline{k}) = z].$$

(3.6)

and if mutation is performed after crossover, then

$$m_{u,v}(z) = \sum_{j,k \in \Omega} \mu_j \frac{x_k + x_{\overline{k}}}{2} [(u \otimes k) \oplus (v \otimes \overline{k}) = z \oplus j].$$

(3.7)

**Theorem 3** (Vose) Whether or not mutation is performed before or after crossover,

$$m_{u,v}(z) = m_{v,u}(z) = m_{u \oplus z, v \oplus z}(0).$$

Theorem 3 told us two facts: (1) mixing matrices are symmetric; (2) any mixing matrix $M(i)$ can be expressed in terms of $M(0)$ and permutation matrix $\sigma_i$, i.e. $M(i) = \sigma_i M(0) \sigma_i$. Therefore, we can define $M(0)$ as the mixing matrix $M$. We need
to point out that the two formulas of theorem 2 are not in general equal to each other. In other words, the mixing result can be different, depending on the order of application of crossover and mutation. However, under some special circumstances, formula 3.7 and formula 3.6 equal to each other. For example, for a crossover-only mixing and a mutation-only mixing, either formula is applicable and will lead to the same result. For a crossover-only mixing, its mutation rate $\mu$ is 0, which implies that the mutation probability distribution is given by

$$
\mu_j = \begin{cases} 
0 & \text{if } j \neq 0 \\
1 & \text{if } j = 0.
\end{cases}
$$

In this case,

$$
m_{u,v}(z) = \sum_{k \in \Omega} \frac{X_k + X_{k}}{2} \left[ (u \otimes k) \oplus (v \otimes k) = z \right]. \quad (3.8)
$$

For a mutation-only mixing, its crossover rate is 0, which implies the crossover probability distribution is given by

$$
\chi_j = \begin{cases} 
0 & \text{if } j \neq 0 \\
1 & \text{if } j = 0.
\end{cases}
$$

In this case, from either formula in theorem 2, we can derive

$$
m_{u,v}(z) = \frac{1}{2} \left( \mu_{u \otimes z} + \mu_{v \otimes z} \right), \quad (3.9)
$$

which coincides with our intuition, i.e. $z$ is obtained by randomly selecting a parent from $u$ and $v$ then applying the mutation mask. In theorem 4, Vose points out another special situation where the two formulas of theorem 2 are equal to each other. Vose gave the proof of in [Vose, 1999].
Theorem 4 (Vose) If mutation is independent, then the two probabilities defined in theorem 2 are equivalent.

A mixing scheme $M : \Lambda \to \Lambda$ is a heuristic function such that, for a population probability distribution $x \in \Lambda$, $M(x)_i$ is the probability that $i \in \Omega$ will be produced as the result of mixing. The mixing scheme $M$ is given by

$$M(x)_i = \sum_{u,v \in \Omega} x_u x_v m_{u,v}(i). \quad (3.10)$$

For $x \in \Lambda$, the mixing scheme $M$ can also be written as

$$M(x) = \sigma_x M \sigma_x.$$  

For a crossover-only mixing, its mixing scheme $M$ is defined by

$$M(x)_i = \sum_{u,v,k \in \Omega} x_u x_v \frac{\chi_k + \chi_k}{2} \left[(u \otimes k) \oplus (v \otimes k) = i\right],$$

which coincides with crossover scheme 3.5 as we expected. For a mutation-only scheme, its mixing scheme $M$ is given by

$$M(x)_i = \sum_{u,v \in \Omega} x_u x_v \frac{1}{2} \left(\mu_{u \oplus i} + \mu_{v \oplus i}\right)$$

$$= \frac{1}{2} \sum_{u \in \Omega} x_u \mu_{u \oplus i} + \frac{1}{2} \sum_{v \in \Omega} x_v \mu_{v \oplus i}$$

$$= \sum_{u \in \Omega} x_u \mu_{u \oplus i},$$

which also coincides with mutation scheme 3.4 as we expected. The mixing scheme 3.10 can be written in matrix format as

$$M(x)_i = x^T M(i) x.$$
In terms of the mixing matrix, above formula can be expressed by

\[ M(x)_i = x^T \sigma_i M \sigma_i x, \]

where \( M = M(0) \).

Theorem 4 also leads to following theorem.

**Theorem 5** Let \( \mathcal{U} \) be a mutation scheme determined by an independent mutation
and \( \mathcal{X} \) be a crossover scheme, then \( \mathcal{U} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{U} \).

**Proof.** Let \( x \in \Lambda \),

\[
\mathcal{U} \circ \mathcal{X}(x)_i = \mathcal{U}(\mathcal{X}(x))_i \\
= \sum_i \mu_{i \oplus i} \mathcal{X}(x)_i \\
= \sum_i \mu_{i \oplus i} \sum_{u,v} x_u x_v \frac{\chi_k + \chi_{\overline{k}}}{2} [(u \otimes k) \oplus (v \otimes \overline{k}) = i] \\
= \sum_{u,v} x_u x_v \sum_{i,k} \mu_{i \oplus i} \frac{\chi_k + \chi_{\overline{k}}}{2} [(u \otimes k) \oplus (v \otimes \overline{k}) = i] \\
= \sum_{u,v} x_u x_v \sum_{m,n,k} \mu_{u \oplus m} \mu_{v \oplus n} \frac{\chi_k + \chi_{\overline{k}}}{2} [(m \otimes k) \oplus (n \otimes \overline{k}) = i] \\
= \sum_{m,n,k} \left( \sum_u x_u \mu_{u \oplus m} \right) \left( \sum_v x_v \mu_{v \oplus n} \right) \frac{\chi_k + \chi_{\overline{k}}}{2} [(m \otimes k) \oplus (n \otimes \overline{k}) = i] \\
= \sum_{m,n,k} \mathcal{U}(x)_m \mathcal{U}(x)_n \frac{\chi_k + \chi_{\overline{k}}}{2} [(m \otimes k) \oplus (n \otimes \overline{k}) = i] \\
= \mathcal{X}(\mathcal{U}(x))_i \\
= \mathcal{X} \circ \mathcal{U}(x)_i. \quad \blacksquare
\]

In the proof of 5, we actually obtained another result.
Theorem 6 For a mixing scheme $M$, if its mutation is independent, then

$$M = U \circ X = X \circ U,$$

where $X$ and $U$ are obtained by setting mutation and crossover to zero respectively.

Theorem 5 and theorem 6 imply that any mixing scheme $M$ with an independent mutation is actually a composition of a mutation scheme and a crossover scheme.

Following theorem is a direct result based on theorem 5 and theorem 6.

Theorem 7 Let $M_1$ and $M_2$ be two mixing schemes. If both are determined by an independent mutation, then $M_1 \circ M_2 = M_2 \circ M_1$.

3.2.5 The Heuristic function of the Simple Genetic Algorithm

Combining the results on mixing and selection operations, we can see the heuristic function $G$ of the Simple Genetic Algorithm is simply the composition of a mixing scheme $M$ and a selection scheme $F$, i.e.

$$G = M \circ F,$$

explicitly, for a population of size $r$, the heuristic function $G$ is given by

$$G(X/r) = M(F(X/r)) = M \circ F(X/r).$$

Recalling formula 3.3, we can see that the Markov chain transition probability from $X$ to $Y$ of the Simple Genetic Algorithm 3.1 is given by

$$P(X, Y) = r! \prod_{i \in \Omega} \frac{(G(X/r))_{Y_i}}{(Y_i)!}.$$

(3.11)
For simplicity, we use $R(G, X, Y)$ to denote above transition probability, i.e.

$$R(G, X, Y) = r! \prod_{i \in \Omega} \frac{(G (X/r))_{Y_i}^{Y_i}}{(Y_i)!}.$$ 

Thus the Markov chain transition matrix $Q$ of the Simple Genetic Algorithm is given by

$$Q_{X,Y} = R(G, X, Y).$$
CHAPTER IV
AN EXTENDED WORK ON SELECTION

The Random Heuristic Search model describes a variety of genetic algorithms. One important assumption of the model is that the survival selection is conducted with replacement. For example, in algorithm 3.2, it is possible for the next generation $Y$ to consist identically of elements from $X$. Many practical genetic algorithms do not enjoy this property. Actually, many genetic algorithms contain one or more selection steps that include selection without replacement. For example, some algorithms require a small portion of the parent generation, perhaps the best $k$ elements, to survive to the next generation. Clearly, the Random Heuristic Search model does not describe the selection without replacement. One goal of this chapter is to show how to model selection without replacement via genetic search. The other goal is to show how to integrate it with the Random Heuristic Search model in order to model some practical genetic algorithms.

We first consider the following paradigm:

1. Given an initial population $X$ of size $r$;
2. Form a population $Y$ of size $k$ by doing independent selections from $X$ with the probability distribution $G(X/r)$;
   \[ (4.1) \]
3. Form a population $Z$ by selecting $r$ individuals from $X$ and $Y$;
4. Replace $X$ with $Z$;
5. Goto step 2.
Step 3 of paradigm 4.1 can be conducted in a number of ways. One approach is to implement the Random Heuristic Search model. Then step 3 can be phrased as "Form population $Z$ by $r$ independent selections with probability distribution $F((X + Y)/(r + k))$", where $F$ is some selection scheme. In this case, step 3 is conducted with replacement. Another approach is to implement the selection without replacement at step 3. There are several variants on how this selection is to be done. For example, one way is to select the best $r - k$ elements of $X$ and combine them with $Y$ to form the next generation $Z$. In the next two sections, we will show how to model both approaches.

4.1 Step 3 done by sampling with replacement

Assume three populations $X$, $Y$ and $Z$ of size $r$, $k$ and $r$ respectively as described in paradigm 4.1. The Random Heuristic Search model describes the transition from $X$ to $Y$ in step 2. The transition probability is given by $R(G, X, Y)$. At step 3, assume that a selection scheme $F : A \to A$ is used to obtain a probability distribution and $Z$ is formed by $r$ independent selections from this distribution. In this case, the transition probability from $X + Y$ to $Z$ with selection scheme $F$ is given by $R(F, X + Y, Z)$, which

$$R(F, X + Y, Z) = P(X + Y, Z) = r! \prod_{i \in \Omega} \frac{(F((X + Y)/(r + k)))^Z_i}{(Z_i)!}.$$  

To find out the transition probability from $X$ to $Z$, we need to range $Y$ through all possible size-$k$ populations from $\Omega$. Thus, the Markov chain transition probability
from $X$ to $Z$ is

$$P(X, Z) = \sum_{Y \in \mathcal{P}_k} R(G, X, Y) R(F, X + Y, Z).$$

4.2 Step 3 done without replacement

It is hard to model selection based on fitness without replacement with the Random Heuristic Search model. An solution is to recalculate the sampling probability distribution after each selection step at step 3 such that the selected elements have less chance of being chosen in the next selection step. Let's consider selecting a subpopulation $Y$ of size $k$ from a population $X$ of size $r$ without replacement. If we imagine $Y$ is formed by selecting one element after another from $X$, then the transition probability from $X$ to $Y$ is the sum of the probabilities of all the ways of forming $Y$. For each way, it consists of a sequence of selection steps. To find out its probability, we recalculate the probability distribution for the next selection step after deleting the selected elements from $X$. We repeat this process $k$ times. For example, suppose $\Omega = \{0, 1\}$ and let $X = [2, 1], Y = [1, 1]$ and $F$ be some selection scheme. There are two ways of obtaining $Y$, i.e. selecting 0 then 1 or selecting 1 then 0. Thus, the probability of obtaining $Y$ from $X$ is given by

$$P(X, Y) = R(F, X, [1, 0]) R(F, X - [1, 0], [0, 1]) + R(F, X, [0, 1]) R(F, X - [0, 1], [1, 0]).$$

Actually, the second term of the formula above is just $R(F, X, [0, 1])$, because

$$R(F, X - [0, 1], [1, 0]) = R(F, [2, 0], [1, 0]) = 1.$$
Thus we have

\[ P(X, Y) = R(F, X, [1, 0]) R(F, X - [1, 0], [0, 1]) + R(F, X, [0, 1]). \]  \hspace{1cm} (4.2)  

To verify this approach, we check if the corresponding row sum of the Markov chain transition matrix equals 1. There are two 2-element subpopulations of \( X = [2, 1] \): \( Y = [1, 1] \) and \( Y' = [2, 0] \). Reasoning as we did to obtain 4.2, we find

\[ P(X, Y') = R(F, X, [1, 0]) R(F, X - [1, 0], [1, 0]). \]

Thus,

\[
\begin{align*}
P(X, Y) + P(X, Y') & = R(F, X, [1, 0]) R(F, X - [1, 0], [0, 1]) + R(F, X, [0, 1]) \\
& + R(F, X, [1, 0]) R(F, X - [1, 0], [1, 0]) \\
& = R(F, X, [1, 0]) [R(F, X - [1, 0], [0, 1]) + R(F, X - [1, 0], [1, 0])] \\
& + R(F, X, [0, 1]) \\
& = R(F, X, [1, 0]) + R(F, X, [0, 1]) \\
& = 1.
\end{align*}
\]

Although we may be able to come up with a general formula for this approach, the formula can be quite complicated as we can tell from the example above. For this reason, we do not favor this method of modeling selections without replacement. Two commonly used selection-without-replacement strategies are truncation selection, where the best \( k \) elements of a population are selected, and random selection, where \( k \)
elements of a population are selected randomly without replacement. Instead of using the Random Heuristic Search model, we use the multiple hypergeometric distribution to model these selection strategies and coordinate their models with Random Heuristic Search to calculate the Markov chain transition probabilities for several variants of paradigm 4.1.

4.2.1 Step 3 done by truncation selection from X

In this case, step 3 of paradigm 4.1 is done by selecting the best \( r - k \) (based on fitness values) elements from \( X \) and then combining them with \( Y \) obtained at step 2 to form \( Z \). From another perspective, it is equivalent to say replacing the worst \( k \) elements of \( X \) by \( Y \) to form \( Z \). If we use \( W \) to denote a subpopulation consisting of the best \( r - k \) elements of \( X \), then \( Y = Z - W \). The selection of \( W \) is not unique, because we do not require an injective fitness function over \( \Omega \). For example, suppose \( \Omega = \{0, 1, 2, 3\} \) with fitness vector \( f = [3, 3, 2, 1] \) and population \( X = [1, 2, 0, 1] \), there are two subpopulations consisting of the best one element of \( X \), which are \([1, 0, 0, 0]\) and \([0, 1, 0, 0]\). Let \( F(W) \) denote the sum of the fitness of all the elements of \( W \).

For a population \( X \), let \( B_{r-k}(X) = \{W \in P_{r-k} : W \leq X \text{ and } F(W) \text{ is maximal}\} \). With greedy search, one can verify that elements of \( B_{r-k}(X) \) are the subpopulations consisting of the best \( r - k \) elements of \( X \). For \( W \in B_{r-k}(X) \), the probability of obtaining \( W \) when the best \( r - k \) elements of \( X \) are selected is given by

\[
\frac{\rho_X(W)}{\rho_X(B_k(X))}
\]
where $\rho_X(W)$ is the probability of selecting $W$ from $X$ and

$$\rho_X(B_k(X)) = \sum_{W \in B_k(X)} \rho_X(W).$$

If $W \in B_{r-k}(X)$ and $W \leq Z$, then the transition probability from $X$ to $Z$ is given by

$$\frac{\rho_X(W)}{\rho_X(B_k(X))} R(G, X, Z - W),$$

where $R(G, X, Z - W)$ is the transition probability of step 2 of paradigm 4.1. Because of the multiple choices of $W$, we need to consider all the possibilities. The Markov chain transition probability of the algorithm is given by

$$P(X, Z) = \sum_{W \in B_{r-k}(X)} [W \leq Z] R(G, X, Z - W) \frac{\rho_X(W)}{\rho_X(B_{r-k}(X))}. \quad (4.3)$$

To verify the formula above, we need to check the following.

**Proposition 8** For all $X \in \mathcal{P}_r$, $\sum_{Z \in \mathcal{P}_r} P(X, Z) = 1$, where $P(X, Z)$ is defined by 4.3.

**Proof.**

$$\sum_{Z \in \mathcal{P}_r} P(X, Z) = \sum_{Z \in \mathcal{P}_r} \sum_{W \in B_{r-k}(X)} [W \leq Z] R(G, X, Z - W) \frac{\rho_X(W)}{\rho_X(B_{r-k}(X))}$$

$$= \frac{1}{\rho_X(B_{r-k}(X))} \sum_{W \in B_{r-k}(X)} \rho_X(W) \sum_{Z \in \mathcal{P}_r} [W \leq Z] R(G, X, Z - W)$$

$$= \frac{1}{\rho_X(B_{r-k}(X))} \sum_{W \in B_{r-k}(X)} \rho_X(W) \sum_{Y \in \mathcal{P}_k} R(G, X, Y)$$

$$= \frac{1}{\rho_X(B_{r-k}(X))} \sum_{W \in B_{r-k}(X)} \rho_X(W)$$

$$= 1. \blacksquare$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
4.2.2 Step 3 done by truncation selection from $X+Y$

In this case, step 3 of paradigm 4.1 is done by selecting the best $r$ elements from $X+Y$ to form the next generation $Z$. Equivalently, $Z$ is formed by deleting the worst $k$ elements from $X+Y$. The transition probability from $X$ to $Y$ at step 2 is given by $R(G, X, Y)$. Let $B_r(X+Y)$ be the set of subpopulations consisting of the best $r$ elements of $X+Y$. If $Y \in \mathcal{P}_k$ and $Z \in B_r(X+Y)$, then the transition probability from $X$ to $Z$ is given by

$$R(G, X, Y) \frac{\rho_{X+Y}(Z)}{\rho_{X+Y}(B_r(X+Y))}$$

Considering all the possible choices of $Y$, we find that the Markov chain transition probability is given by

$$P(X, Z) = \sum_{Y \in \mathcal{P}_k} [Z \in B_r(X+Y)] R(G, X, Y) \frac{\rho_{X+Y}(Z)}{\rho_{X+Y}(B_r(X+Y))}. \quad (4.4)$$

We verify the formula above by checking if the row sum of Markov chain transition matrix is 1.

**Proposition 9** For all $X \in \mathcal{P}_r$, $\sum_{Z \in \mathcal{P}_r} P(X, Z) = 1$, where $P(X, Z)$ is defined by 4.4.

**Proof.**

$$\sum_{Z \in \mathcal{P}_r} P(X, Z) = \sum_{Z \in \mathcal{P}_r} \left( \sum_{Y \in \mathcal{P}_k} [Z \in B_r(X+Y)] R(G, X, Y) \frac{\rho_{X+Y}(Z)}{\rho_{X+Y}(B_r(X+Y))} \right)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[
\sum_{Y \in \mathcal{P}_k} R(G, X, Y) \frac{1}{\rho_X + \rho_Y (B_r (X + Y))} \sum_{Z \in \mathcal{P}_r} [Z \in B_r (X + Y)] \rho_{X+Y} (Z)
\]

= \sum_{Y \in \mathcal{P}_k} R(G, X, Y)

= 1. \blacksquare

4.2.3 Step 3 done by Random selection from X

In this case, step 3 of paradigm 4.1 is done by randomly selecting \( r - k \) elements from \( X \) and then combine them with \( Y \) obtained at step 2 to form population \( Z \). Some genetic algorithms use random selection to maintain the gene diversity to avoid premature convergence. Let \( W \) be a subpopulation of size \( r - k \) of population \( X \). From our discussion on multiple hypergeometric distribution, we know the probability of selecting \( W \) from \( X \) is given by \( \rho_X (W) \). If \( W \in \mathcal{P}_{r-k} (X) \) and \( W \leq Z \), then the transition probability from \( X \) to \( Z \) is given by \( \rho_X (W) R(G, X, Z - W) \). Considering all possible choices of \( W \), we find the Markov chain transition probability is given by

\[
P(X, Z) = \sum_{W \in \mathcal{P}_{r-k} (X)} [W \leq Z] R(G, X, Z - W) \rho_X (W). \quad (4.5)
\]

Check that the row sum of the Markov chain transition matrix is 1.

**Proposition 10** For all \( X \in \mathcal{P}_r \), \( \sum_{Z \in \mathcal{P}_r} P(X, Z) = 1 \), where \( P(X, Z) \) is defined by 4.5.

**Proof.**

\[
\sum_{Z \in \mathcal{P}_r} P(X, Z)
\]
\[
\begin{align*}
&= \sum_{Z \in \mathcal{P}_r} \sum_{W \in \mathcal{P}_{r-k}(X)} [W \leq Z] R(G, X, Z - W) \rho_X(W) \\
&= \sum_{W \in \mathcal{P}_{r-k}(X)} \rho_X(W) \sum_{Z \in \mathcal{P}_r} [W \leq Z] R(G, X, Z - W) \\
&= \sum_{W \in \mathcal{P}_{r-k}(X)} \rho_X(W) \sum_{Y \in \mathcal{P}_k} R(G, X, Y) \\
&= \sum_{W \in \mathcal{P}_{r-k}(X)} \rho_X(W) \\
&= 1. \blacksquare
\end{align*}
\]
Our extended work on selection without replacement enables us to calculate the Markov chain transition probabilities for several practical genetic algorithms, namely Whitley’s Genitor Algorithm [Whitely, 1989], Syswerda’s Steady-State Genetic Algorithm [Syswerda, 1989] and Eshelman’s CHC Algorithm [Eshelman, 1991]. Actually, paradigm 4.1 and our work of the proceeding chapter are partially inspired by these algorithms. Whitley’s Genitor Algorithm and Syswerda’s Steady-State Genetic Algorithm both are known as steady-state genetic algorithms. The main difference between steady-state genetic algorithms and traditional genetic algorithm is that, at each evolution step, only a few members in the current generation are replaced. The change between the parent generation and child generation is very small. On the other hand, Eshelman’s CHC Algorithm is a generational genetic algorithm, in the sense that the change between the parent and child generations is significant. We will give the Markov chain models of Whitley’s Genitor Algorithm and Syswerda’s Steady-State Genetic Algorithm. However, we can only model Eshelman’s CHC Algorithm partially.
5.1 Whitley’s Genitor Algorithm

Whitley’s Genitor algorithms [Whitely, 1989] is described by

1. Given an initial population $X$ of size $r$;
2. Select two parents from $X$ by ranking selection, then apply mixing to them to produce one child;
3. Replace the worst element of $X$ with the child produced at step 2 to form population $Z$;
4. Replace $X$ by $Z$;
5. Goto step 2.

From our previous work, we can see that algorithm 5.1 resembles paradigm 4.1 if, in paradigm 4.1, $k$ is set to be 1 and truncation selection from $X$ is used at step 3. The Markov chain transition probability for algorithm 5.1 is described by formula (4.3). We do not make an effort to model the ranking selection at step 2 of algorithm 5.1, because its model is encapsulated in the Random Heuristic Search model which is embedded in formula (4.3).

5.2 Syswerda’s Steady-State Genetic Algorithm

Syswerda’s Steady-State Genetic Algorithm [Syswerda, 1989] is similar to Whitley’s Genitor Algorithm except that two parents are selected by proportional selection.
at step 2 and the step 3 can be conducted by truncation deletion or random deletion from $X$. Depending on how step 3 is done, the Markov chain transition probability are described either by formula (4.3) for truncation deletion or by formula (4.5) for random deletion. Still, we do not need to be concerned about the selection difference at the step 2 of the algorithm, because its detail is described in the Random Heuristic Search model which is embedded in both formulas (4.3) and (4.5).

### 5.3 Eshelman's CHC Algorithm


1. Given an initial population $X$ of size $r$;
2. Apply crossover within $X$ to obtain a population $Y$ of size $k$;
3. Form $Z$ by selecting the best $r$ elements from $X + Y$;
4. Replace $X$ by $Z$;
5. Goto step 2.

Note that the mixing at step 2 is crossover-only. The Markov chain model of algorithm 5.2 is described by formula 4.4. We need to point out that algorithm 5.2 does not describe all the features of the CHC Algorithm. Some extra characteristics of the CHC Algorithm include: (1) instead of using common crossover operations at step 2, CHC uses a special kind of uniform crossover called HUX where exactly half of the different
bits are swapped; (2) CHC restricts the crossover to the strings that are at a certain distance apart; (3) Once a termination condition is detected, all population members except the best member go through a mutation with a fixed mutation rate. For the three characteristics that formula (4.4) has not addressed, they can be reflected by further work on the Random Heuristic Model. For example, for (3), it is actually required to use a mutation-only heuristic function for some states (i.e. populations) of the Markov chain instead of the crossover-only function used at step 2, while a truncation selection is enforced before the mutation operation takes place.
CHAPTER VI
APPLICATI0NS OF MARKOV CHAIN MODELS

One reason that we are interested in the Markov chain models of genetic algorithms is that the models can give us a deeper understanding about the transitions of genetic algorithms. These models also provide a powerful tool to investigate theoretical results, such as the rate of convergence. In this chapter, we derive average absorption time based on our Markov chain models. We focus on a special class of Markov chains, namely absorbing chain over a finite set of states.

6.1 Absorbing Markov Chains

Let \( P = (P_{i,j}) \) be a Markov chain transition matrix. A state \( i \) of \( P \) is absorbing if \( p_{ii} = 1 \). A state \( i \) is transient if there exists a state \( j \) which can be reached from \( i \), but it is impossible to reach \( i \) from \( j \). A Markov chain is an absorbing chain if (1) it contains at least one absorbing state, and (2) from every state it is possible to go to an absorbing state. Let \( P \) be a \( s \times s \) transition matrix for an absorbing chain. Suppose that there are \( m \) absorbing states, and thus \( s - m \) transient states. Then, \( P \) can be written as

\[
P = \begin{bmatrix}
I & 0 \\
R & Q
\end{bmatrix},
\]

where \( Q \) is a \((s - m) \times (s - m)\) matrix that gives transition probabilities between transient states; \( R \) is a \((s - m) \times m\) matrix that describes the transition probabilities between transient states and absorbing states; and \( I \) is an \( m \times m \) identity matrix representing the absorbing states. In a finite absorbing Markov chain, the probability
that the process will be absorbed into an absorbing state is 1 [Kemeny et al., 1976]. The matrix \((I - Q)^{-1}\) is called the Markov chain’s Fundamental Matrix. We summarize some quantities related to absorbing chains (see [Kemeny et al., 1976] and [Isaacson and Madsen, 1976]): (1) If a process starts in transient state \(t_i\), the expected number of periods that the process will spend in transient state \(t_j\) is given by the \(ij\)th entry of the matrix \((I - Q)^{-1}\). (2) If a process starts from transient state \(t_i\), the probability that the process will be absorbed in absorbing state \(a_j\) is given by the \(ij\)th entry of the matrix \((I - Q)^{-1}R\). If the process starts from a transient state, the number of time periods that the process will spend in transient states before absorption is given by the corresponding entry in \((I - Q)^{-1}1\). This time can also be interpreted as the expected absorption time.

### 6.2 Expected absorption time based on Markov chain models

Most genetic algorithms with a specific termination condition can be regarded as absorbing Markov chains. In this section, we investigate several examples based our Markov chain models.

We evaluate our models against three fitness functions, namely, a random function, the counting-zero function, and a deceptive function. For random function, we assign a random number from \([1, 100]\) to be the fitness value for each \(i \in \Omega\). For counting-zero function, the fitness function is defined by

\[
f_i = |\bar{z}|,
\]
where \( |i| \) denotes the number of 0's contained in \( i \in \Omega \). Our deceptive function is defined by

\[
f_i = \begin{cases} 
10 \times i & \text{if } i \neq 0 \\
10 \times 2^i & \text{if } i = 0 
\end{cases}
\]

Let the absorbing states be the populations containing at least one occurrence of the maximum fitness individual. Assume that the Markov chain transition matrix is

\[
P = \begin{bmatrix} I & 0 \\
R & Q \end{bmatrix}
\]

We first find out \((I - Q)^{-1}\). Then, we evaluate \((I - Q)^{-1} 1\). Let \( X \) be a size-\( r \) population drawn from \( \Omega \) of size \( n \). The probability of obtaining \( X \) with \( r \) independent selections from \( \Omega \) is given by

\[
p_X = \binom{r}{X_0} \binom{r - X_0}{X_1} \cdots \binom{r - X_0 - \cdots - X_{n-2}}{X_{n-1}} \left( \frac{1}{n} \right)^r
\]

\[
= \frac{r!}{X_0!X_1!\cdots X_{n-1}!} \left( \frac{1}{n} \right)^r.
\]

Enumerating all the size-\( r \) populations, we get a multinomial distribution \( p \) over \( \mathcal{P}_r(\Omega) \). If \( p' \) is the entries of \( p \) corresponding to transient states(populations), the average absorption time for any starting transient state is given by \( p'^T (I - Q)^{-1} 1 \).

For our experiment, we set \( l = 3, r = 2, \) and \( k = 1 \). We assume a one-point crossover with crossover rate \( \chi = 0.75 \) and a mutation with mutation rate \( \mu = 0.0625 \). The average absorption time for models 4.3, 4.4 and 4.5 based on three fitness
functions is given in the following table

<table>
<thead>
<tr>
<th></th>
<th>Truncation selection from ( X + Y )</th>
<th>Truncation selection from ( X )</th>
<th>Random selection from ( X )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random fitness</td>
<td>19.08</td>
<td>19.20</td>
<td>27.12</td>
</tr>
<tr>
<td>Count-zero fitness</td>
<td>11.46</td>
<td>11.95</td>
<td>17.20</td>
</tr>
<tr>
<td>Deceptive fitness</td>
<td>2417.27</td>
<td>1164.04</td>
<td>65.27.</td>
</tr>
</tbody>
</table>

It is not our goal to justify the efficiency of various genetic algorithms. However, based on these experimental results, it seems that implementing random selection at the survival stage is more efficient than truncation selection for genetic algorithm with an absorbing Markov chain model.
CHAPTER VII
CONCLUSION

In this work, we reviewed the Random Heuristic Search model and its application in the Simple Genetic Algorithm model. Two new schemes, a mutation scheme and a crossover scheme, were defined. Three theorems regarding the mixing scheme with independent mutation are presented. We extended the Random Heuristic Search model to several practical genetic algorithms and described the Markov chain models of them. In the end, we investigated the average absorption time based on their Markov chain models and three fitness functions.

The Markov chain model is a powerful tool in theoretical analysis of genetic algorithms. However, its practical usage is limited by the computational power of the current hardware. For example, our experiment results of section 6.2 were obtained via Maple V running on a Pentium II 450 personal computer. It took one to two hours to obtain each results. The size of the Markov chain matrix grows exponentially with the length of the binary strings, which suggests that it is hard to obtain similar results for large practical problems. One solution to this obstacle is to conduct the computation work with a compiled language, such as C++, rather than a symbolic language, such as Maple. Another solution is to define equivalence classes over all states and run the experiment over the equivalence classes instead of states.

The work on Markov chain models of genetic algorithms is far from complete. Many interesting problems are still untouched, for example the relationship between the Markov chain model and Schema Theorem. We believe that it is possible to find
an exact formula describing the schema change over generations. One may also be able to derive Schema Theorem from the formula.
REFERENCES


