PERFORMANCE ANALYSIS FOR GENETIC ALGORITHMS

by
Hermean Wong

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Mechanical Engineering

October 1995

APPROVAL PAGE

PERFORMANCE ANALYSIS FOR GENETIC ALGORITHMS

Hermean Wong

Dr. Ming C. Leu, Dissertation Advisor — Date
Professor of Mechanical Engineering, NJIT

Dr. Rong-Yaw Chen, Committee Member — Date
Professor of Mechanical Engineering, NJIT

Dr. Denis Blackmore, Committee Member — Date
Professor of Mathematics, NJIT

Dr. Nouri Levy, Committee Member — Date
Associate Professor of Mechanical Engineering, NJIT

Dr. Zhiming Ji, Committee Member — Date
Assistant Professor of Mechanical Engineering, NJIT

ABSTRACT

PERFORMANCE ANALYSIS FOR GENETIC ALGORITHMS

by
Hermean Wong

Genetic algorithms have been shown effective for solving complex optimization problems such as job scheduling, machine learning, pattern recognition, and assembly planning. Due to the random process involved in genetic algorithms, the analysis of performance characteristics of genetic algorithms is a challenging research topic. Studied in this dissertation are methods to analyze convergence of genetic algorithms and to investigate whether modifications made to genetic algorithms, such as varying the operator rates during the iterative process, improve their performance. Both statistical analysis, which is used for investigation of different modifications to the genetic algorithm, and probability analysis, which is used to derive the expectation of convergence, are used in the study. The Wilcoxon signed rank test is used to examine the effects of changing parameters in genetic algorithms during the iterations. A Markov chain is derived to show how the random selection process affects the genetic evolution, including the so called genetic drift and preferential selection. A link distance is introduced as a numerical index for the study of the convergence process of order-based genetic algorithms. Also studied are the effects of random selection, mutation operator, and the combination of both to the expected average link distance. The genetic drift is shown to enforce the convergence exponentially

with increase in the number of iterations. The mutation operator, on the other hand, suppresses the convergence. The combined results of these two parameters lead to a general formula for the estimation of the expected number of iterations needed to achieve convergence for the order-based genetic algorithm with selection and mutation and provide important insights about how order-based genetic algorithms converge.

BIOGRAPHICAL SKETCH

Author: Hermean Wong

Degree: Doctor of Philosophy

Date: October 1995

Undergraduate and Graduate Education:

- Doctor of Philosophy in Mechanical Engineering,
  New Jersey Institute of Technology,
  Newark, New Jersey, 1995

- Master of Science in Mechanical and Industrial Engineering,
  New Jersey Institute of Technology,
  Newark, New Jersey, 1991

- Bachelor of Science in Mechanical Engineering,
  National Taiwan University,
  Taiwan, Republic of China, 1986

Major: Mechanical Engineering

Publications:

Wong, Hermean, April 1991, "Genetic Algorithm for Solving Printed Circuit Board Assembly Planning Problems," *Proceedings of the First ASME Region II Graduate Student Technical Conference*, New Jersey Institute of Technology, Newark, New Jersey, pp. 32-33.

Wong, Hermean, Ming C. Leu, Sep. 1991, "PCB Assembly Optimization Software Part I - HSP01.C," *Optimization for SFP and HSP Placement Machines and Adhesive Dispensors*, Report of Project with Universal Instrument Corporation, Binghamton, New York.

Ji, Zhiming, Ming C. Leu, and Hermean Wong, 1991, "Application of Linear Assignment Model for Planning of Robotic Printed Circuit Board Assembly," *Manufacturing Processes and Materials Challenges in Microelectronic Packaging*, AMD-Vol. 131, pp. 35-41.

Wong. Hermean, 1991, *Genetic Algorithm for Solving Printed Circuit Board Assembly Planning Problems*, Master Thesis, New Jersey Institute of Technology, Newark, New Jersey.

Wong, Hermean, Ming C. Leu, April 1992, "PCB Assembly Optimization Software Part II - HSP02.C," *Optimization for SFP and HSP Placement Machines and Adhesive Dispensors*, Report of Project with Universal Instrument Corporation, Binghamton, New York.

Ji, Zhiming, M. C. Leu, and Hermean Wong, Dec. 1992, "Application of Linear Assignment Model for Planning of Robotic Printed Circuit Board Assembly," *Journal of Electronic Packaging*, Vol. 114, No. 4, pp. 455-460.

Leu, Ming C., Hermean Wong, Zhiming Ji, July 1992, "Genetic Algorithm for Solving Printed Circuit Board Assembly Planning Problems," *Proceedings of Japan-U.S.A. Symposium on Flexible Automation*, San Francisco, California, pp. 1579-1586.

Wong, Hermean, MengChu Zhou, Aug. 1992, "Automated Generation of Modified Reachability Tree for Petri Nets," *Proceedings of 1992 IEEE Regional Conference on Control Systems*, Polytechnic University, Brooklyn, New York, pp. 85 - 90.

Wong, Hermean, Ming C. Leu, 1993, "Adaptive Genetic Algorithm for Optimal Printed Circuit Board Assembly Planning," *Annals of the CIRP*, Vol. 42/1/1993, pp. 17-20.

Wong, Hermean, Apr. 1993, "Adaptive Parameter Search for an Order-Based Genetic Algorithm," *Proceedings of ASME Region II Graduate Student Technical Conference*, Polytechnic U., Brooklyn, New York, pp. 42-43.

Wong, Hermean, Leu, Ming C., 1993, "Adaptive Search of Operator Rates for Order-Based Genetic Algorithms," *Proceedings of 1993 IEEE Regional Conference on Control Systems*, New Jersey Institute of Technology, Newark, New Jersey, pp. 66-69.

Leu, Ming C., Wong, Hermean, Zhiming Ji, Dec. 1993, "Planning of Component Placement/Insertion Sequence and Feeder Setup in PCB Assembly Using Genetic Algorithm," *Journal of Electronic Packaging*, Vol. 115, No. 4, pp. 424-432.

Wong, Hermean, Leu, Ming C., July 1994, "Application of Genetic Algorithm for Optimization of Printed Circuit Board Assembly Systems," *Proceedings of the Third International Conference on Automation Technology*, Taipei, R.O.C., Vol. 1, pp. 327-334.

Presentations:

Wong, Hermean, April 1991, "Genetic Algorithm for Solving Printed Circuit Board Assembly Planning Problems," *The First ASME Region II Graduate Student Technical Conference*, New Jersey Institute of Technology, Newark, New Jersey.

Wong, Hermean, MengChu Zhou, Aug. 1992, "Automated Generation of Modified Reachability Tree for Petri Nets," *1992 IEEE Regional Conference on Control Systems*, Polytechnic University, Brooklyn, New York.

Wong, Hermean, Apr. 1993, "Adaptive Parameter Search for an Order-Based Genetic Algorithm," *ASME Region II Graduate Student Technical Conference*, Polytechnic U., Brooklyn, New York.

Wong, Hermean, Leu, Ming C., 1993, "Adaptive Search of Operator Rates for Order-Based Genetic Algorithms," *1993 IEEE Regional Conference on Control Systems*, New Jersey Institute of Technology, Newark, New Jersey.

Wong, Hermean, Leu, Ming C., July 1994, "Application of Genetic Algorithm for Optimization of Printed Circuit Board Assembly Systems," *The Third International Conference on Automation Technology*, Taipei, R.O.C.

This dissertation is dedicated to
my parents
my lovely wife, Poulie Ju
my son, Arick Wong
and my daughter, Nancy Wong

ACKNOWLEDGMENT

The author will always remain indebted to his advisor, Professor Ming C. Leu, for his ever present guidance and support, technical, financial and interpersonal, throughout this research.

Thanks are due to Professors Rong-Yaw Chen, Denis Blackmore, Nouri Levy, and Zhiming Ji for dedicating their time to serve as members of the committee.

Timely help and suggestions came from many good friends: Dr. Jack G. Zhou, Dr. Hsin-Te Liao, Dr. Tea-Quin Kim, and many others.

The author is grateful to the members of his family, especially his father, his mother who died during his doctorate research, his son Arick Wong, for their support, their love, faith, and patience.

Lastly, the author would like to express his greatest and deepest gratitude to his lovely wife, Poulie Ju, for the many sacrifices she has made for his education that he will never be able to compensate in any way.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

Table **Page**

LIST OF FIGURES

NOMENCLATURE

$a_{i,m}$	normalized percentage of minimum solution from adaptive operator production ratio method w.r.t. optimal solution of problem i after m iterations
$b_{i,m}$	normalized percentage of minimum solution from fixed operator production ratio method w.r.t. optimal solution of problem i after m iterations
C	a set of genes
$C(n, i)$	Combination of i objects from n objects
C_b	a set of binary coded genes
C_p	a set of permuted genes
c_i	the i-th gene
$\overline{D}$	average link distance from randomly generated mating pools
$\overline{D}_m$	average link distance after m iterations
D_c	average link distance after convergence
$\overline{d}$	average link distance
$\Delta\overline{d}$	average link distance change
$\Delta\overline{d}_i$	average link distance change of sub-case i
d'_{ij}	link distance after mutation operation between link i and link j
d_{ij}	link distance between link i and link j
$E(\,)$	expected value
$E_M(\Delta\overline{d})$	expected average link distance change due to mutation operator

NOMENCLATURE
(Continued)

$E_S(\overline{D}_m)$	expected average link distance due to selection after m iterations
e_{ki}	number of copies of link k in a mating pool with population size n
$e_{k,m}$	number of copies of link k in a mating pool after m iterations
f_i	cost of link i
G_j	j-th operator
H_0	null hypothesis
H_1	alternative hypothesis
h	number of operators
K_i	signed rank of the i-th sample
L_j	the lower bound of the operator production ratio of operator j
l	link length
M_n^m	mating pool with n links after m iterations
m	number of iterations
N	number of all possible states
n	number of links or samples
$n!$	n factorial
n_j	number of links after j-th operator is applied
O_m	mutation operator
$\mathbf{P}$	transition probability matrix

NOMENCLATURE
(Continued)

$\tilde{p}_m$	a probability vector contains all p_i's after m iterations
$p(X)$	probability of event X
$p(X_1 \mid X_2)$	conditional probability of event X_1 under event X_2
$p_{i,m}$	the probability of staying in state i after m iterations
p_i	the probability of staying in state i
p_{ij}	transition probability from state i to state j
q_j	production ratio of the j-th operator
R_n	the set of the ranks of U_n
r	operator rate
r_j	operator rate of the j-th operator
ra_i	rank of i-th link (or sample)
S_i	state i
$\tilde{s}$	a link
T	total number of iterations
U	the set of all possible links
U_n	a subset of U which contains n links
$X_{i,m}$	outcome of the random variable for Wilcoxon signed rank test of problem i after m iterations
x_i	random variables

NOMENCLATURE
(Continued)

y_m	total number of link pairs with link distance equal to zero after m iterations
$Z(\alpha)$	standard normal distribution w.r.t. a level of significance α
Φ	change of the portion of zero link distances
$\aleph$	a set of positive integer
ψ_j	optimal solution of problem i
α	level of significance
$\gamma_{i,m}$	minimum solution from fixed operator production ratio method w.r.t. optimal solution of problem i after m iterations
$\eta_{i,m}$	minimum solution from adaptive operator production ratio method w.r.t. optimal solution of problem i after m iterations
μ	mean of samples
μ_0	mean of the random variables
τ	a specified number of iterations
ω_i	selection probability of link i
$\omega_{i,m}$	selection probability of link i after m iterations
ζ	sum of the signed ranks
$\zeta_{a,n}$	critical value of S for n samples w.r.t. a level of significance α

CHAPTER 1

INTRODUCTION

1.1 Background of the Research

Genetic algorithms have been shown effective for solving complex optimization problems in job scheduling, machine learning, pattern recognition, assembly planning, and others (Davis, 1985; Englander, 1985; Leu, Wong, and Ji, 1992; Wong and Leu, 1993). The concept of the initial genetic algorithm was based on the improvement of bit string representations for real problems. An initial pool of solutions represented by bit strings, called the *mating pool,* is created usually from a random process. The solutions in the pool are then randomly selected to be applied with some operations for creating new solutions. The improvement of the solutions was made by the creation of new bit strings with better schemata than old ones. A fitness value for each bit string representation is evaluated according to the main concern in a real problem and is used as the solution improvement criterion since it represents goodness of strings.

Some researchers have expanded the bit string representation technique to other representation schemes. Grefenstette, et al., (1985) used a node string to solve the traveling salesperson problem. Koza (1990) used a LISP function and argument string as a non-linear genetic algorithm problem solving technique. Shahookar, et al., (1990) used a character string for solving the standard cell placement problem. Leu, et al., (1993) used integer strings to solve the planning problem of printed circuit board assembly.

The random process involved in genetic algorithms makes it difficult to analyze their performance characteristics including the effects of some parameters and their optimal values, the population size of the initial mating pool, and the number of iterations needed to stop a genetic algorithm. Random selection plays an important role in genetic algorithms. Due to the probabilistic nature of random selection, there is always the existence of selection drift or preferential selection accompanying a genetic algorithm. The *genetic drift* is the selection induced bias due to the fact that a sequence of selections with equal probabilities of different objects has a high probability of resulting in unequal numbers of different objects. *Preferential selection* is similar to genetic drift except that the probabilities of choosing objects are different.

To analyze the performance of a genetic algorithm, it is essential to analyze the effect of random selection. Goldberg and Segrest (1987) used *finite Markov chain* (Kemeny and Snell, 1960) to analyze the *genetic drift* and *preferential selection* for a simple genetic algorithm with binary coding (0 or 1). They showed that Markov chain analysis in general is useful to understanding the performance of finite genetic algorithms with binary coding, sizing populations appropriately, and selecting proper operation rates. Other researchers (Nix and Vose, 1992; Suzuki, 1993) extended the Markov model for the integer representation. The effect of random selection in general coding techniques, such as the permutation coding, remains an open research issue. The Markov chain analysis is not ideal because the possible constructions of the mating pool are tremendous and it is practically impossible to calculate their results considering the enormous computations required.

Some statistical analyses have been studied to find optimal parameters in the genetic search. Schaffer, et. al., (1989) used statistical results from a large number of experiments to show the effects of parameters on the performance of some genetic algorithms. Instead of finding optimal parameters globally, some researchers studied the adaptation of parameters during the genetic evolution process. Focusing on operator rates, Davis (1989) adapted the probabilities of operators during the genetic search when multiple operators are used. In (Davis, 1989), only one operator was chosen among the genetic operators to create the offspring according to their associated probabilities. The sum of the probabilities of all operators is equal to 100%. The adapting method raises the probabilities of the operators that generate more of the offspring, compared with other operators, which are better than the current best solution during the genetic iteration process. Because of the flexibility of genetic algorithms, it is often desirable to add more modifications to the algorithm to suit new applications. A general scheme for analyzing the performance of an order-based genetic algorithm, however, is not available from the literature.

1.2 Research Objectives and Tasks

The objectives of this dissertation are two-fold : (1) to investigate whether modifications added to genetic algorithms improve their performance or not and (2) to build a foundation for analyzing the convergence of order-based genetic algorithms.

Both statistical analysis and probability analysis are made in this study. A Wilcoxon signed rank test (Lawer, 1985; Mosteller, 1973) is used to compare an adaptive operator production ratio method with the corresponding fixed operator production ratio method. The genetic drift and preferential selection in genetic algorithms are analyzed without regard to specific coding methods. Instead of the use of genes which is coding method related, the selection of links is considered in the development of a new method for the convergence analysis. A link distance is defined for order-based genetic algorithms as the measure of difference between two links. The average link distance of the mating pool is then used as the reference of convergence in the analysis of order-based genetic algorithms. The changes of average link distance due to the individual effects of random selection and mutation operation are investigated. By combining the effects of both random selection and mutation operation, the expected average link distance is formulated and used to estimate the expected number of iterations needed for the convergence of a genetic algorithm.

1.3 Outlines of Dissertation

The remaining of the dissertation is organized as follows. Chapter 2 gives a description of the genetic algorithm and describes a statistical analysis for comparing variations in an order-based genetic algorithm with an increasing mating pool. In Chapter 3, the Markov chain analysis based on the states of the mating pool is discussed. In Chapter 4, an index called the link distance is created for studying the convergence of the order-based genetic algorithm, and the expected average link distance of the initial mating pool is discussed. Chapter 5 discusses the expected average link distance change due to random selection. Chapter 6 discusses the expected average link distance change due to mutation operation. Chapter 7 combines the effects of both random selection and mutation operation and formulates an equation for estimation of the expected number of iterations needed for order-based genetic algorithms. Chapter 8 concludes the study.

CHAPTER 2

STATISTICAL ANALYSIS

In this chapter, we will provide the description of a general genetic algorithm and a methodology for statistically investigating the effect of adapting the production ratios of genetic algorithms to an increasing-mating-pool order-based genetic algorithm (Wong, 1991). For a rigorous discussion of the genetic algorithm we give the following definitions:

Definition 1: Let $\aleph$ be a set of l continuous positive integers starting from 1, i.e. $\aleph = \{1, 2, \ldots, l\}$ and C be a set of l *genes*, i.e. $C = \{c_1, c_2, \ldots, c_l\}$, where each element c_i is a gene. $C_b = \{c_1, c_2, \ldots, c_l\}$ is called binary coded if $c_i \in \{0, 1\}$ $\forall$ $i \in \aleph$. $C_p = \{c_1, c_2, \ldots, c_l\}$ is said to be permutation coded if $c_i \in \aleph$ and $c_i \neq c_j$ if $i \neq j$ $\forall$ $i \in \aleph$.

Definition 2: A link is defined as $\tilde{s} = \{s_1, s_2, \ldots, s_l\}$, where s_k is the gene at the k-th position (called locus) of the link, and s is the mapping $s : \aleph \rightarrow C$. l is called the *link length.*

Definition 3: The set U is the collection of all possible links, i.e. $U = \{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_\phi\}$, where ϕ is the total number of all possible links. $\phi = 2^l$ for binary coded genes and $\phi = l!$ for permutation coded genes.

Definition 4: Let U_n denote a subset of U containing n links, i.e. $U_n = \{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n\}$, where n is the number of links in U_n.

Definition 5: The mating pool M_n is defined as $M_n = \{(\tilde{s}_1, \omega_1), (\tilde{s}_2, \omega_2), \ldots, (\tilde{s}_n, \omega_n)\}$ where $\tilde{s}_i \in U_n$ and ω_i is the probability for $\tilde{s}_i$ being chosen in the genetic algorithm optimization process. $\sum_{i=1} \omega_i = 1$.

Definition 6: The cost function is defined as $f : U \rightarrow \Re$, where $\Re$ is the set of real numbers. $f(\tilde{s}_i) = f_i$ is the cost of solution $\tilde{s}_i$.

Definition 7: An operator is a mapping $G : U_m \rightarrow U_n$, where m is the number of parents and n is the number of offspring generated by the operator.

A traditional genetic algorithm usually consists of the following steps:

1. Randomly generate a set of initial parents, U_n, that forms a mating pool M_n. Select the operators used for the genetic algorithm. Set an operator rate, r_j, $0 < r_j < 1$, for each operator. Find f_i for each $\tilde{s}_i$. Assign the selection probability, ω_i, for link $\tilde{s}_i$ according to its cost f_i.
2. Select n links (with replacement) from the mating pool M_n according to the selection probability of each link to form an intermediate mating pool M'_n.
3. For each operator, do step 3.1.

   3.1For each link $\tilde{s}_i'$ in M'_n, randomly draw a real number between 0 and 1. If the drawn number is less than r_j, apply the operator to the link to create a new link and replace $\tilde{s}_i'$ with this link.
4. Replace M_n with M'_n.
5. Repeat steps 2, 3, and 4.

There are many variations in practical applications of the genetic algorithm in terms of constitution of genes, selection, replacement, and the operators used. Selection of parameters such as population size and operator rate are also important study issues. In this Chapter, we will use a statistical method called Wilcoxon signed rank test to compare different genetic algorithms.

Based on the experience in using the genetic algorithm for printed circuit board assembly planning (Leu, Wong, and Ji, 1993), we will compare an adaptive operator production ratio method with a fixed operator production ratio method for an order-based genetic algorithm with an increasing mating pool. The adaptive operator production ratio method uses a set of adaptive operator production ratios instead of using fixed numbers of operator production ratios during the iterative process. The operator production ratios are adjusted according to the proportions of survived offspring generated by individual operators. Since the genetic algorithm is a heuristic search method, the results of different trials are usually different, even using the same values of parameters. Instead of comparing the means of the best solutions, what should be compared are the differences between the best solutions obtained from different methods for a broad range of similar problems. A Wilcoxon signed rank test is applied for this comparison. We will show the obtained experimental results from the Wilcoxon signed rank test after describing the fixed operator production ratio method and the adaptive operator production ratio method in the following sections.

2.1 The Fixed Operator Production Ratio Method

Let n_0 denote the number of initial parent links, h denote the number of operators, n_j denote the number of links in the mating pool M_{n_j} after applying operator G_j, $1 \leq j \leq h$. Instead of using the operator rate in the traditional genetic algorithm just described, we assign an operator production ratio to each operator G_j as $q_j = n_j / n_{j-1}$. q_j is used to control the number of offspring links created by each operator in the iteration. q_j must be greater than 1 to guarantee the production of offspring links by operator G_j.

For the two genetic algorithms tested in this chapter, we assign the selection probability of each link according to the rank of the link compared to the other links in the mating pool. The ranking of the mating pool before the operations is defined as follows.

Definition 8: The *ranking* of the set U_n is defined as $R_n = \{ra_1, ra_2, \ldots, ra_n\}$ where ra_i is a positive integer between 1 and n representing the rank of $\tilde{s}_i$ in U_n based on the cost of $\tilde{s}_i$, i.e. f_i. R_n is a permutation of integers from 1 to n such that $ra_i = 1$ if f_i is the minimum cost.

The ranking of the mating pool after the genetic operations is defined as follows.

Definition 9: The ranking of the set $U_{n+k} = \{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n, \tilde{s}_{n+1}, \ldots, \tilde{s}_{n+k}\}$ is defined as $R_{n+k} = \{ ra_1, ra_2, \ldots, ra_n, ra_{n+1}, ra_{n+2}, \ldots, ra_{n+k}\}$, where k is the total number of new offspring links generated by the operators. For every $\tilde{s}_{n+i}$, $1 \le i \le k$, $ra_{n+i} = n + i$.

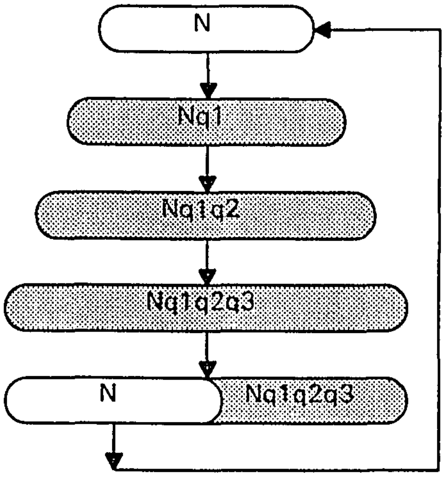

Figure 2.1 The Process of Genetic Algorithm in Each Iteration

We will assign the selection probability of link $\tilde{s}_i$ proportional to $(n - ra_i + 1)$. That is, the selection probabilities are assigned such that : (1) the lower the rank of a link, the higher the selection probability, and (2) the selection probabilities of all links have the ratio 1 : 2 : 3 : ... : n. This is better than assigning the selection probabilities of the links proportional to the fitness values of these links, because it avoids the possibility that selection probabilities of some links may be unreasonably high. The fixed operator production ratio method is as follows:

1. Let the total number of iterations for the genetic algorithm be T. Randomly generate a set of initial parents, U_{n0}, that forms the mating pool M_{n0}. Find the cost f_i, $i = 1, 2, \ldots, n_0$. Find R_{n0}, i.e. the ranking of U_{n0}. The probability ω_i is proportional to $(n_0 - ra_i + 1)$. Since $\sum_{i=1}^{0} \omega_i = 1$, we can find that

$$\omega_i = \frac{n_0 - ra_i + 1}{\sum_{k=1}^{n_0} (n_0 - ra_i + 1)} = \frac{2(n_0 - ra_i + 1)}{n_0(n_0 + 1)}$$

   Select the production ratio q_j of each operator G_j, $j = 1, 2, \ldots, h$, where h is the total number of operators.

2. Select parent link(s) from the mating pool $M_{n_{j\text{-}1}}$ according to the selection probability of each link. Sequentially apply the operators to create offspring links for each operator G_j with production ratio q_j, $j = 1, 2, \ldots, h$. The mating pool is sequentially enlarged to M_{nj}'s as shown in Figure 2.1. The ranks of the newly created links are assigned as described before. The selection probabilities of the links are updated as

$$\omega_i = \frac{n_j - ra_i + 1}{\sum_{k=1}^{n_j} (n_j - ra_i + 1)} = \frac{2(n_j - ra_i + 1)}{n_j(n_j + 1)}$$

3. After all the operators are applied, find f_i of M_{n_h}, $i = n_0+1, n_0+2, \ldots, n_h$. Rerank M_{n_h}.
4. Define the new mating pool M'_{n0} such that $\tilde{s}_i$ in M'_{n0} is the same as $\tilde{s}_i$ in M_{n_h} for $i = 1, 2, \ldots, n_0$. Replace M_{n0} with M'_{n0}.
5. If the number of iterations is equal to T, stop; else go to step 2.

We will call the fixed operator production ratio method as the fixed method later in the text. It should be noted that the population size of the mating pool is increased after each operation. After all the operations are applied, we then create a new mating pool from the collection of the best solutions among the original mating pool and the newly created offspring. The new mating pool has the same size as the initial mating pool. The newly created links are not evaluated until all the operators have been applied.

The probability of having the combined effect of different operators is controlled by both the number of initial parents and the total number of offspring generated by the various operators in each iteration. The selection probability of the links for the last operation is

$$\omega_i = \frac{n_{h-1} - ra_i + 1}{\sum_{k=1}^{n_{h-1}} (n_{h-1} - ra_i + 1)} = \frac{2(n_{h-1} - ra_i + 1)}{n_{h-1}(n_{h-1} + 1)}$$

The probability of the links in the initial mating pool U_{n0} being selected as parent(s) for the last operator G_h is

$$\sum_{i=1}^{n_0} \omega_i = \sum_{i=1}^{n_0} \left(\frac{n_{h-1} - ra_i + 1}{\sum_{k=1}^{n_{h-1}} (n_{h-1} - ra_i + 1)} \right) = \frac{\frac{(2n_{h-1} - n_0 + 1)n_0}{2}}{\frac{(n_{h-1} + 1)n_{h-1}}{2}} = \frac{(2n_{h-1} - n_0 + 1)n_0}{(n_{h-1} + 1)n_{h-1}} \qquad (2.1)$$

Let $n_{h\text{-}1} = kn_0$, $k > 1$, and $n_0 >> 1$, (2.1) becomes

$$\sum_{i=1}^{n_0} \omega_i = \frac{(2k-1)n_0}{k^2 n_0} = \frac{2k-1}{k^2} \qquad (2.2)$$

Equation (2.2) represents the probability that the parent link of the offspring generated by the last operator G_h is from U_{n0}. If the number of offspring links generated in each iteration is the same as the number of links in the initial mating pool, $k \approx 2$. Then the probability for the links in U_{n0} being chosen as the parent links for the last operator is around 75%. The larger the number of offspring links is generated in each generation, the more the emphasis is placed on the effect of combined operators. It should be noted that the first operator uses only the mating pool M_{n0}, without any unevaluated parents.

2.2 The Adaptive Operator Production Ratio Method

In this section, we will describe a heuristic approach to adjust operator production ratios during the iterative process of the order-based genetic algorithm. The main idea of the adaptive operator production ratio method is that during the iterations we raise the production ratios of the operators that generate more survived offspring links and reduce the production ratios of the operators that generate less survived offspring links. The total number of parent links and the total number of offspring links in each generation are fixed; that is, n_0 and n_h are fixed during the genetic evolution process. Although the larger n_h in each generation, the more improvement is expected, but the total computation time is also increased. So, the efficiency of the genetic algorithm is not necessarily improved by increasing n_h. We will denote the adaptive operator production ratio method as the adaptive method.

To fix the population size in each iteration, with a given set of operator production ratios $\{q_1, q_2, \ldots, q_h\}$, $\prod_{j=1}^{h} q_j$ should be kept constant. If an offspring link is created by a sequential combination of several operators, credits are given to all operators involved in generating this link.

Let the production ratios of operators be updated after every τ iterations. Also let N_j be the total number of offspring links created by operator G_j that are contained in the mating pool M_{n0} in the next iteration, for each of these τ iterations. To adjust the operator production ratios in the iterations, we let

$$q_j' = kq_j\left(1 + \frac{\dfrac{N_j}{\tau \cdot (n_j - n_{j-1})}}{\sum_{i=1}^{h}\left(\dfrac{N_i}{\tau \cdot (n_i - n_{i-1})}\right)}\right) \tag{2.3}$$

where q'_j is the operation production ratio of operator j in the following iterations, j = 1, 2, ..., h. The proportional factor k in (2.3) should be such that the number of offspring links generated in each iteration remains unchanged. For this condition to hold, it is required that

$$k = \left(\prod_{j=1}^{h}\left[1 + \frac{\dfrac{N_j}{\tau \cdot (n_j - n_{j-1})}}{\sum_{i=1}^{h}\left(\dfrac{N_i}{\tau \cdot (n_i - n_{i-1})}\right)}\right]\right)^{-\frac{1}{h}}$$

Therefore,

$$q_j' = q_j\left(1+\frac{\dfrac{N_j}{\tau\cdot\left(n_j-n_{j-1}\right)}}{\sum_{i=1}^{h}\left(\dfrac{N_i}{\tau\cdot\left(n_i-n_{i-1}\right)}\right)}\right)\left(\prod_{j'=1}^{h}\left[1+\frac{\dfrac{N_{j'}}{\tau\cdot\left(n_{j'}-n_{j'-1}\right)}}{\sum_{i=1}^{h}\left(\dfrac{N_i}{\tau\cdot\left(n_i-n_{i-1}\right)}\right)}\right]\right)^{-\frac{1}{h}} \tag{2.4}$$

Another consideration in the adaptive method is that the productions of some of the operators may keep decreasing to the degree that these operators no longer affect the iterative process. An operator can not create offspring if its production ratio is equal to or less than 1. So the production ratio of each operator should be given a lower bound which is larger than 1, i.e. $L_j > 1$, where L_j is the lower bound of the production ratio of operator G_j.

The process of the adaptive method is thus as follows:

1. Let the total number of iterations for the genetic algorithm be T. Randomly generate a set of initial parents, U_{n0}, that forms a mating pool M_{n0}. Find f_i, $i = 1, 2, \ldots, n_0$. Find R_{n0}. The probability ω_i is proportional to $(n_0 - ra_i + 1)$. Since $\sum_{i=1}^{0}\omega_i = 1$, we can find that

$$\omega_i = \frac{n_0 - ra_i + 1}{\sum_{k=1}^{n_0}\left(n_0 - ra_i + 1\right)} = \frac{2\left(n_0 - ra_i + 1\right)}{n_0\left(n_0 + 1\right)}$$

   Select the initial operator production ratio q_j, $j = 1, 2, \ldots, h$ where h is the total number of operators. Select a period representing the number of iterations, τ, to update the operator production ratios.

2. Let $k = 1$, where k indexes the number of iterations for adapting the operator rates. Let $N_j = 0$ for $j = 1, 2, \ldots, h$.

3. Select parent links from the mating pool $M_{n_{j-1}}$ according to the selection probabilities of the links. Sequentially apply the operators to create offspring links according to their production ratios. The mating pool is sequentially enlarged to M_{n_j}'s. The selection probabilities of the links are updated as

$$\omega_i = \frac{n_j - ra_i + 1}{\sum_{k=1}^{n_j} (n_j - ra_i + 1)} = \frac{2(n_j - ra_i + 1)}{n_j(n_j + 1)}$$

4. After all the operators are applied, find f_i for each link of M_{n_h}, $i = n_0+1, n_0+2, \ldots, n_h$. Rerank M_{n_h}.
5. Define a new mating pool M'_{n0} such that $\tilde{s}_i$ in M'_{n0} is the same as $\tilde{s}_i$ in M_{n_h} for $i = 1, 2, \ldots, n_0$. Replace M_{n0} with M'_{n0}.
6. For every link in the new M_{n0}, increment N_j, $j = 1, 2, \ldots, h$, by 1 if operator G_j is involved in generating the link. Increment k by 1.
7. Repeat steps 3 to 6 while $k \leq \tau$.
8. Compute the new operator production ratios q_j', $j = 1, 2, \ldots, h$, using equation (2.4),
9. If $q_j' < L_j$, let $q_j' = L_j$ for $j = 1, 2, \ldots, h$. Then compute the new operator production ratios again as $q_j'' = k'q_j'$ where

$$k' = \left(\frac{\prod_{j=1}^{h} q_j}{\prod_{j=1}^{h} q_j'} \right)^{\frac{1}{h}}$$

Repeat this step until the new operator rates satisfy the lower bound requirement, i.e. $q_j'' \geq L_j$ for $j = 1, 2, \ldots, h$.

10. If the number of iterations is more than T, stop; else go to step 2.

2.3 Wilcoxon Signed Rank Test

The most straightforward way of showing whether a variation in a genetic algorithm makes the algorithm better or not is using statistical analysis. Golden and Stewart (1985) used a statistical analysis called Wilcoxon signed rank test (Lawer, 1985; Mosteller, 1973) to compare different heuristics for solving the traveling salesperson problem. The Wilcoxon signed rank test is a non-parametric hypothesis test. It can be applied to all continuous distributions, especially for those with symmetric densities, for testing the null hypothesis $H_0 : \tilde{\mu} = \tilde{\mu}_0$, where $\tilde{\mu}_0$ is the mean of the random variables which is usually known, and $\tilde{\mu}$ is the mean of samples. Let H_1 denote the alternative hypothesis to be tested. H_1 can be either $H_1 : \tilde{\mu} \neq \tilde{\mu}_0$ or $H_1 : \tilde{\mu} > \tilde{\mu}_0$ or $H_1 : \tilde{\mu} < \tilde{\mu}_0$.

Suppose X_1, X_2, ..., X_n comprise a random sample corresponding to a symmetric continuous population distribution with mean $\tilde{\mu}$, where n is the number of sampling. If $X_i = \tilde{\mu}_0$, X_i is discarded and n is reduced by one. Let K_i be the rank of the values of $|X_i - \tilde{\mu}_0|$ in increasing order, $i = 1, 2, \ldots, n$. If m samples are tied for the k-th rank, each of them is assigned a rank $\frac{k + (k+1) + \ldots + (k+m-1)}{m}$. Define the signed rank, R_i, as

$$R_i = \begin{cases} K_i & \text{if } X_i - \tilde{\mu}_0 > 0 \\ -K_i & \text{if } X_i - \tilde{\mu}_0 < 0 \end{cases}$$

where $i = 1, 2, \ldots, n$. Let ζ denote the sum of the signed ranks, i.e. $\zeta = \Sigma R_i$.

If the null hypothesis H_0 is true, one would expect a somewhat uniform mixing of both positive and negative values of $X_i - \tilde{\mu}_0$ in the sampled data. Since the sum of the first n integers is $\frac{n(n+1)}{2}$, under the null hypothesis H_0 we would expect one half of the signed ranks be positive and the sum of these positive signed ranks be $\frac{n(n+1)}{4}$. The other half of the signed ranks would be expected to be negative and the sum of the signed ranks be $-\frac{n(n+1)}{4}$. So, the total sum is expected to be around 0.

For the upper one-tailed alternative, $H_1 : \tilde{\mu} > \tilde{\mu}_0$, the sum would likely be near $\frac{n(n+1)}{2}$. For the lower one-tailed alternatives, $H_1 : \tilde{\mu} < \tilde{\mu}_0$, the sum would likely be near $-\frac{n(n+1)}{2}$. For the two-tailed alternative, $H_1 : \tilde{\mu} \neq \tilde{\mu}_0$, it would be expected that ζ would likely be near to $-\frac{n(n+1)}{2}$ or $\frac{n(n+1)}{2}$.

Because ζ is discrete, for a desired level of significance α, a critical value of ζ that yields approximately the desired α level needs to be found. For the upper one-tailed alternative, $H_1 : \tilde{\mu} > \tilde{\mu}_0$, at significance level α (and for sample size n), the critical value $\zeta_{\alpha,n}$ is defined by $P(\zeta \geq \zeta_{\alpha,n}; H_0) = \alpha$. H_0 is rejected if $\zeta \geq \zeta_{\alpha,n}$. Owing to the symmetry of the ranking scheme, H_0 is rejected in favor of the lower one-tailed alternative $H_1 : \tilde{\mu} < \tilde{\mu}_0$ if $\zeta \leq \zeta_{\alpha,n}$. For the two-tailed alternative $H_1 : \tilde{\mu} \neq \tilde{\mu}_0$, H_0 is rejected if $\zeta \geq \zeta_{\alpha/2,n}$ or $\zeta \leq -\zeta_{\alpha/2,n}$.

For $n \geq 10$, $\zeta_{\alpha,n}$ can be approximated by

$$\zeta_{\alpha,n} = Z(\alpha)\sqrt{\frac{n(n+1)(2n+1)}{6}}$$

where $Z(\alpha)$ is the standard normal distribution such that α is the proportion α of the area is to the left of $Z(\alpha)$; see Figure 2.2.

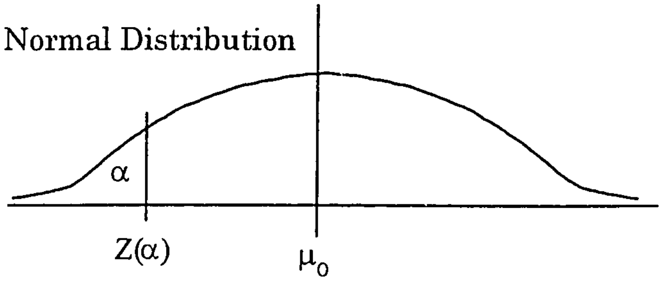

Figure 2.2 Standard normal distribution

For our application of the Wilcoxon signed rank test to investigate whether a variation added to the order-based genetic algorithm makes it better or worse, we use the null hypothesis, $H_0 : \tilde{\mu} = \tilde{\mu}_0$, which has the assumption that the two methods perform equally. The comparison is based on the best (minimal) solutions searched from both methods for the same number of iterations. Since the fixed method is the one to be challenged, the random variable is defined as the difference between the normalized best solution achieved by the adaptive method and that of the corresponding fixed method. According to the null hypothesis $H_0 : \tilde{\mu} = \tilde{\mu}_0$, the mean, $\tilde{\mu}_0$, is equal to 0. The alternative hypothesis, $H_1 : \tilde{\mu} < \tilde{\mu}_0$, is that the adaptive method outperforms the fixed method.

2.4 Experiments and Discussion

To show the comparison of the two methods, we use the classical traveling salesperson problem. Figure 2.3 shows an example of the traveling salesperson problem with only 6 nodes. The trajectory of the travel in Figure 2.2 is represented as 1–2–5–6–4–3. Twenty four tests are made for the comparison of the proposed fixed and adaptive methods. They are divided into 8 groups according to the number of nodes. The numbers of nodes are 50, 60, 70, 80, 90, 100, 110 and 120 for the eight groups. The coordinates of the nodes are randomly and uniformly generated in a square area. The link is represented by a permutation of the nodes. The operator production ratios are updated every twenty iterations for the adaptive method.

The initial population size n_0 is 40 for all the tests. Four operators are used in sequence: order crossover operator (Olive, 1987), inversion operator, rotation operator (Leu, Wong, and Ji, 1993), and mutation operator. The total number of offspring generated in each iteration is 80. The links in the initial population are all randomly generated. The initial operator production ratios for each test are also randomly chosen in a reasonable range. Both the fixed method and adaptive method are used to find the optimal solutions. The initial operator production ratios of the adaptive method are the same as the corresponding fixed method.

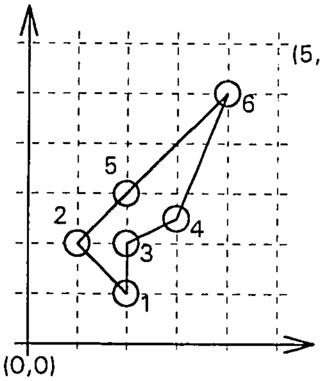

Figure 2.3 An example of traveling salesperson problem

When applying the Wilcoxon signed rank test, the random variable should be first identified. Let $\eta_{i,m}$ denote the minimum travel distance achieved by the adaptive method for test i after m iterations, $i = 1, 2, \ldots, 24$. Let $\gamma_{i,m}$ denote the minimum travel distance achieved by the fixed method for test i after m iterations, $i = 1, 2, \ldots, 24$. Let ψ_i denote the travel distance of the optimal solution of test i, $i = 1, 2, \ldots, 24$. Let $a_{i,m}$ denote the normalized difference between $\eta_{i,m}$ and ψ_i, $a_{i,m} = (\eta_{i,m} - \psi_i)/\psi_i$, $i = 1, 2, \ldots, 24$. Let $b_{i,m}$ denote the normalized difference between $\gamma_{i,m}$ and ψ_i, $b_{i,m} = (\gamma_{i,m} - \psi_i)/\psi_i$, $i = 1, 2, \ldots, 24$. The random variables are $X_{i,m} = a_{i,m} - b_{i,m}$, $i = 1, 2, \ldots, 24$.

The null hypothesis to challenge is that the optimal solution search abilities for the two methods during the whole process are equally powerful, which implies $\tilde{\mu}_0 = 0$. The alternative hypothesis is that the adaptive operator rate search method yields a better solution. Instead of the traditional way of testing the rejection of the hypothesis with a pre-specified level of significance, we find the largest level of significance to support the hypothesis that the adaptive method is better. The level of significance can be thought as the largest probability error to support the argument that the adaptive method is better.

Since the global minimum for each of the tests is unknown, the best solution found for each test from both methods is used as the global minimum. The best solutions found from the genetic algorithms all converge to very nice trajectories. Therefore, we assume that the differences between the global optimal solutions and the best solutions from genetic algorithms are reasonably small. This assumption is especially true when the focus is on the early stages of convergence for the genetic algorithms.

Table 2.1 illustrates the random variables, $X_{i,m}$, from the tests. The first column in Table 2.1 is the test number from 1 to 24. The bottom three rows of Table 2.1 are different from the other rows which list the singed ranks of the tested problems. The third row from the bottom contains the sum of signed rank, $\zeta_{\alpha,24}$, for various numbers of iterations. The second row from the bottom contains the corresponding probability in normal distribution, $Z(\alpha)$. Since the total number of problems is 24 for the Wilcoxon signed rank test, the sum of the signed ranks can be approximated from normal distribution as $\zeta_{\alpha,24} = Z(\alpha)\sqrt{\frac{24(24+1)(2\cdot 24+1)}{6}} = 70Z(\alpha)$. The last row of Table 2.1 contains the largest level of significance, α, to reject the null hypothesis for different number of iterations, m. The percentages represent the largest probability errors for various stages of the evolutionary processes to conclude that the adaptive method outperforms the fixed method. For any level of significance less than 50%, the adaptive method statistically performs better. For any level of significance around 50%, these two methods are of about the same performance statistically. The level of significance should be as small as possible to support the hypothesis.

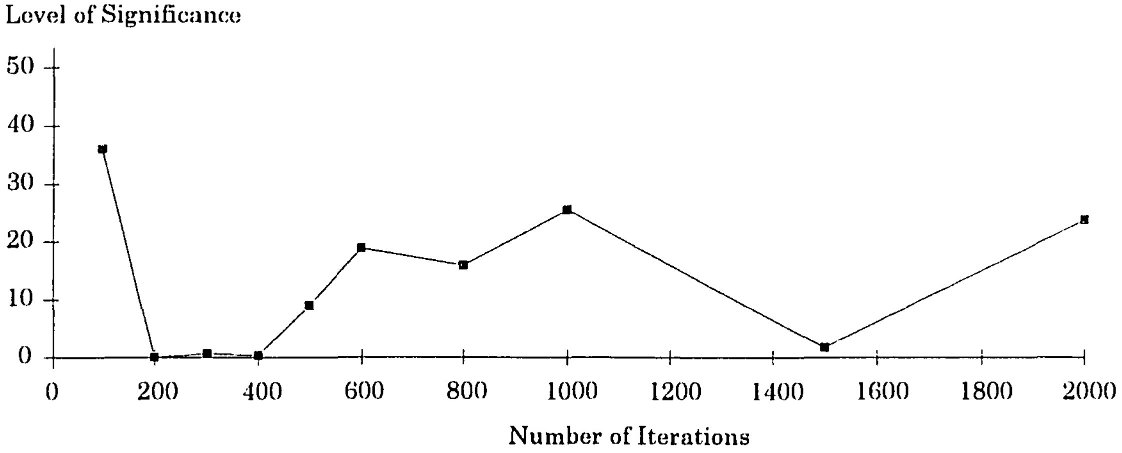

Figure 2.4 Level of significance versus number of iterations

Table 2.1 Random variables sampled from the tests

Test	$X_{i,100}$	$X_{i,200}$	$X_{i,300}$	$X_{i,400}$	$X_{i,500}$	$X_{i,600}$	$X_{i,800}$	$X_{i,1000}$	$X_{i,1500}$	$X_{i,2000}$
1	16	1	-3	1	5	24	14	24	-20	-3
2	-15	-13	-11	-8	-12	-3	-20	-5	-16	-20
3	-21	-14	-2	-2	-19	-23	-13	23	3	5
4	17	12	6	7	15	4	-18	-4	-23	-23
5	-8	-2	8	11	8	-5	-4	-12	-19	-7
6	-6	3	-1	6	4	7	6	3	13	4
7	-20	-11	-7	-5	-7	-6	-2	-13	-11	9
8	9	9	13	4	13	1	7	7	17	11
9	-23	-16	12	17	11	15	-15	-8	-22	-17
10	14	7	14	10	14	9	9	18	15	13
11	7	-5	10	-9	-6	-17	-11	-10	-10	-14
12	10	-15	-16	-13	23	11	16	19	5	8
13	-1	4	-5	-15	10	12	22	17	7	19
14	-22	-17	-18	-12	-22	-14	-24	-1	-18	-1
15	12	-8	-15	-16	-2	10	5	14	6	21
16	4	-10	-20	-3	-3	-21	-23	-6	2	18
17	-18	-23	-22	-14	-17	-22	-10	-11	-21	-10
18	-3	-20	-19	-18	-21	-19	-8	-20	-14	-6
19	24	-6	-4	-19	-9	8	17	2	8	15
20	-19	-19	-23	-20	-18	-2	-3	-9	-9	-12
21	13	-21	-9	-23	-20	18	19	-15	-12	-16
22	-2	-22	-17	-24	-24	-20	-21	-21	-24	-22
23	11	-18	-24	-21	-16	-16	-12	-16	-4	2
24	-5	-24	-21	-22	-1	-13	-1	-22	-1	-24
Σ	-26	-228	-174	-188	-94	-62	-70	-46	-148	-50
Z(α)	-0.37	-3.26	-2.49	-2.69	-1.34	-0.89	-1	-0.657	-2.114	-0.714
α	36%	0.06%	0.65%	0.36%	9%	18.8%	15.9%	25.5%	1.73%	23.7%

Figure 2.4 shows the level of significance, α, versus the number of iterations. It is clear that during the former iterations, the adaptive method

converges much faster than the fixed operator rate method except at the beginning of the genetic search. This is because the operator rates of the two methods are still very similar at the beginning. Among all the problems for all stages, the adaptive method statistically outperforms the fixed method. Figure 2.4 provides a numerical evidence that the adaptation added to the increasing-mating-pool order-based genetic algorithm actually improves the performance. Using the same method, we can test other variations such as different initial populations of the mating pools, different operator rates, etc. We can also verify the speed of convergence during the evolution processes instead of just looking at the final solutions.

One thing we need to point out is that the adaptive method takes a little bit longer computation time than the fixed method. However, the extra computation time added is very small (<<0.1%) compared to the computation time for the genetic evolution process. The computation time needed to vary the operator production ratios is also small and it is not calculated for every iteration.

CHAPTER 3

PROBABILITY ANALYSIS

The statistical analysis of genetic algorithm can only show the numerical results after a lot of trials. It is desirable to establish a more analytical model for performance prediction before trying the genetic evolution process. Since the genetic algorithm involves a lot of random processes, the establishment of a probability model to analyze the expected performance of a genetic algorithm is a main focus of this dissertation. Based on the ideas in (Goldberg and Segrest, 1987), we use a simple genetic algorithm having only selection to analyze genetic drift and preferential selection. The simple genetic algorithm contains the following steps:

1. Randomly generate a set of initial parents, U_n, that forms a mating pool M_n^t, where t represents the number of iterations and $t = 0$ at the beginning. Decide on the selection probability, ω_i, of each link $\tilde{s}_i$.
2. Select n links (with replacement) according to the selection probabilities of the links in the mating pool M_n^t to form another mating pool M_n^{t+1}.
3. Let $t = t + 1$.
4. Repeat steps 2, 3 until the process has reached a pre-specified maximum number of iterations.

It is obvious that the process is likely to come to a situation that all the links in the mating pool are identical. This situation is defined as follows:

Definition 11: Let M_n^0 be the initial mating pool and M_n^i be the mating pool after i selections. If there exists a number t such that $M_n^t = M_n^k$ for all $k > t$, the mating pool is said to have *converged*. For a converged mating pool if all the selection probabilities are identical, i.e. $\omega_i = \omega$, $1 \le i \le n$, the phenomenon of this convergence is called genetic drift. Otherwise, it is called preferential selection.

According to Definition 11, if any one of the links in the mating pool is different from the others, there is always a possibility that this link will not be selected in the next iteration. Once it is not selected the mating pool is changed, which conflicts with the definition that $M_n^t = M_n^k$ for all $k > t$. So, for a converged mating pool M_n, all links are identical.

3.1 Markov Chain Analysis

We use the finite Markov chain method to analyze the convergence property of the genetic algorithm. Suppose we have a sequence of random variables x_0, x_1, ..., and suppose the possible values of these random variables are drawn from the set $\aleph = \{1, 2, \ldots, l\}$. Let the random variable x_t denote the state number at time t. The system is in state S_i at time t if $x_t = i$. If at each time t there is a fixed probability p_{ij} that the system will be in state S_j at time t+1 when the system was in state S_i at time t, we say the sequence of random variables forms a Markov chain. The fixed quantities p_{ij} are said to be *transition probabilities*:

$$p_{ij} = p\{x_{t+1} = j \mid x_t = i\}$$

We provide the following definitions to describe the states in the evolution process.

Definition 12: Define a state $S_i = (e_{1i}, e_{2i}, \ldots, e_{ni})$, where i is the state number starting from 1 and $\sum_{k=1} e_{ki} = n$, $0 \le e_{ki} \le n$. e_{ki} is the number of duplicates of link k in a mating pool, which is an integer, and n is the population size.

Definition 13: Define a probability vector $\tilde{p} = (p_1, p_2, \ldots, p_N)$ where p_k is the probability of staying at state S_k and N is the number of all possible states.

For the genetic algorithm, the mating pool can be in any of the states defined by Definition 12. During the selection process, the mating pool changes from one state to another state. The probability of staying in any of the states is described by Definition 13. For our analysis, t represents the number of iterations and $t = 0$ stands for the initial state (i.e. before any selection). We also assume that the first state, S_1, is equal to (1, 1, ..., 1). According to Definition 12, the total number of states N is

$$N = C(2n-1,n) = \frac{(2n-1)!}{(n-1)!n!} \tag{3.1}$$

where $n!$ is n factorial. Equation (3.1) can be derived from the combination of $n-1$ separators and n objects.

Some states are not reachable from other states because during the selection process, it is possible that some of the links in the initial mating pool will no longer be contained in the later mating pools (since they are not selected in the selection process). They are impossible to be selected again since they are no longer in the mating pool for the later selections. We will describe this reachability problem as follows.

Definition 14: State S_j is said to be later than state S_i, denoted by $S_i \rightarrow S_j$, if for any $e_{ki} = 0$, $k = 1, 2, \ldots, n$, e_{kj} is also zero.

According to Definition 14, if state j is not later than state i, there exists at least one $e_{ki} = 0$ that $e_{kj} \neq 0$. Since it is impossible to select a link if the link is not in the mating pool, it is impossible that $e_{kj} > 0$ when $e_{ki} = 0$. Thus state j is not reachable from state i if S_j is not later than S_i.

The transition probability can be found as follows. Let **P** be the transition probability matrix, $\mathbf{P}=[p_{ij}]$, $0 \leq i, j \leq N$, where

$$p_{ij} = p\left(S_i, S_j\right) = \begin{cases} \begin{pmatrix} n \\ e_{1j}, e_{2j}, \ldots, e_{nj} \end{pmatrix} \prod_{\substack{k=1 \\ e_{ki} \neq 0}}^{n} \left(\frac{e_{ki}\omega_k}{\sum_{l=1}^{n} e_{li}\omega_l} \right)^{e_{kj}} & if\ S_i \rightarrow S_j \\ 0 & elsewhere \end{cases} \tag{3.2}$$

and

$$\begin{pmatrix} n \\ e_{1j}, e_{2j}, \ldots, e_{nj} \end{pmatrix} = \frac{n!}{e_{1j}!e_{2j}!\ldots e_{nj}!}$$

Assume that initially the links in the mating pool are all different from each other. This is very possible when l is large. Thus $S_{x_0} = (1, 1, \cdots, 1)$, i.e. $e_{kx_0} = 1$, $0 \leq k \leq n$. The probability vector of the initial state is $\tilde{p}_0 = (1, 0, \cdots, 0)$. The probability vector after m iterations is equal to $\tilde{p}_0\mathbf{P}^m$.

3.2 Probability of Reaching an Absorbing State

Among all the states, some states can not reach other states. They are called *absorbing states* defined as follows.

Definition 15: An absorbing state is a state, denoted by S_a, at which the mating pool is converged. Let S_{all} denote the set of all absorbing states.

According to Definition 15, for any S_a, there is an $e_{\varphi a}$ such that $e_{\varphi a} = n$, $e_{ka} = 0$, for $k \neq \varphi$, $1 \leq k \leq n$, $\varphi = 1, 2, \ldots, n$. It is obvious that the total number of absorbing states is equal to n.

We are interested in the probability of converging into any of the absorbing states, in other words, the probability of a mating pool becoming converged. Since the events of getting into any of the absorbing states are mutually exclusive, the probability of getting into any of the absorbing states is equal to the sum of the probabilities of getting into each of the absorbing states. Namely,

$$P(x \in S_{all}) = \sum_{S_a \in S_{all}} P(x = S_a) = \sum_{k=1}^{n} P(x = S_{ak})$$

where S_{ak} is an absorbing state. We can find $P(x \in S_{all})$ after m iterations from Equation (3.2).

3.3 A Simple Example

Let us look at a simple example. Suppose there are only two links in the mating pool, i.e. $n = 2$. The number of possible states is $N = 3$. Let $S_1 = (1, 1)$, $S_2 = (2, 0)$, $S_3 = (0, 2)$. Since the total number of states is only 3, we can not assume that the initial state is in S_1. The initial probability vector is $\tilde{p}_0 =$ (0.5, 0.25, 0.25) if the selection probabilities are equal for both links, $\omega_1 = \omega_2 = 0.5$. From Equation (3.2) The transition probability matrix is equal to

$$\mathbf{P} = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The probability vector after m iterations is

$$\tilde{p}_m = \tilde{p}_0\mathbf{P}^m = \left(\tfrac{1}{2}\cdot\left(\tfrac{1}{2}\right)^m, \tfrac{1}{2}-\tfrac{1}{4}\cdot\left(\tfrac{1}{2}\right)^m,\ \tfrac{1}{2}-\tfrac{1}{4}\cdot\left(\tfrac{1}{2}\right)^m\right).$$

There are two absorbing states, S_2 and S_3. The probability of getting into the absorbing states is $p_{2,m} + p_{3,m} = 1 - 0.5^{m+1}$. From this formula, we can find that only 6 iterations is needed to exceed 99% probability of getting into the absorbing states. If the selection probabilities are different for the two links, $\tilde{p}_0 = (2\omega_1(1-\omega_1),\ \omega_1^2,\ (1-\omega_1)^2)$. The probability of getting into the absorbing states is $P(x \in S_{all}) = 1 - (2\omega_1(1-\omega_1))^{m+1}$. For $\omega_1 = 0.6$, $P(x \in S_{all}) = 1 - 0.48^{m+1}$. It is not difficult to prove that the probability of getting into the absorbing state in preferential selection is always higher than the probability in genetic drift. For a large ω_1, the probability is very close to 1 even for a small m. This can explain the phenomenon that when the mating pool contains a link with a high selection probability, there is a very good chance that the genetic algorithm will have a pre-mature convergence.

Shown above is a Markov chain analysis for the genetic algorithm with only selection. Although the genetic algorithm with only selection actually is not a genetic algorithm since it can not improve the solution, the study is very useful to providing a general idea about how the genetic drift and preferential selection affect the genetic algorithm. It also provides an evidence that the genetic drift is the lower bound for convergence of all kinds of preferential selections. In the next chapter, we will define a better reference for studying the convergence of order-based genetic algorithm and compare the results with those from the Markov chain analysis for the above example.

CHAPTER 4

AVERAGE LINK DISTANCE

Due to the random process involved in the genetic algorithm, analysis of the behavior of genetic algorithm is very difficult. It is desirable to establish a fundamental method for analyzing the convergence of the iterative process due to the random selection involved in a genetic algorithm. Two general methods researchers have used for the formal analysis of genetic algorithm are the schema theorem (Holland, 1975) and the Markov chain analysis (Goldberg and Segrest, 1987; Nix and Vose, 1992). Schemata represent subsets of binary strings which must have certain bit values in some bit positions while the other bit positions can have bit values of either 0 or 1. The schema theorem provides useful information about the genetic search space in terms of genes, but it can not show the status of convergence for the mating pool.

The early Markov chain analysis in (Goldberg and Segrest, 1987) for genetic algorithms is also based on genes. Some recent research works of the genetic algorithm using Markov chain, such as (Nix and Vose, 1992; and Suzuki, 1993), are based on links. They showed that the relationship between links plays an important role when analyzing the actual converge behavior of the genetic algorithm. The Markov chain analysis uses a transition probability matrix for analyzing the convergence of a genetic algorithm. Because of the tremendous matrix size and the huge amount of calculations required, it is computationally prohibitive to use the Markov chain analysis for the estimation of convergence in applying genetic algorithms to practically all real problems.

To analyze the convergence behavior of the genetic algorithm, we devise a numerical index, called *average link distance*, which describes the overall dissimilarity relationship among the links. In this chapter, we will define this numerical index and use it to analyze the convergence of order-based genetic algorithms. We will show how the expected average link distance can be obtained with the use of Markov chains. In the next chapter we will develop a new and better way for obtaining the expected average link distance without the use of Markov chains. The link distance is defined as follows:

Definition 16: Two links $\tilde{\phi} = \langle \phi_1, \phi_2, \ldots, \phi_n \rangle, \tilde{\varphi} = \langle \varphi_1, \varphi_2, \ldots, \varphi_n \rangle$ contain genes which are permutations of integers, where $\phi_i \in \aleph$ and $\varphi_i \in \aleph$, $i = 1, 2, \ldots, n$, and $\phi_i \neq \phi_j$, $\varphi_i \neq \varphi_j$ if $i \neq j$. Let m denote the number of (ϕ_i, φ_i) pairs such that $\phi_i \neq \varphi_i$, $i = 1, 2, \ldots, n$. It is obvious that $m \in \{0, 2, 3, \ldots, n\}$. Define the *link distance* between the two links as:

$$d(\tilde{\phi}, \tilde{\varphi}) = \begin{cases} m - 1 & if\ m > 1 \\ 0 & if\ m = 0 \end{cases}$$

The link distance is useful as a reference to represent how much the two links under consideration in the mating pool are different from each other. According to Definition 16, we can find the average link distance for a mating pool M_n. Let $\bar{d}$ denote the average of all link distances,

$$\bar{d} = \frac{\sum_{0 \le i < j \le n} d(\tilde{s}_i, \tilde{s}_j)}{C(n,2)} = \frac{2 \sum_{0 \le i < j \le n} d(\tilde{s}_i, \tilde{s}_j)}{n(n-1)}$$

For each state S_i, there is a corresponding average link distance $\bar{d}_i$. It is easy to show that if S_i is an absorbing state, $\bar{d}_i = 0$. We can think of $\bar{d}$ as an indicator of convergence. The smaller $\bar{d}$ is, the closer the mating pool is to convergence. We can also find the *expected average link distance* after m iterations of a mating pool. Let $E(\bar{D}_m)$ denote the expected average link distance, where $\bar{D}_m$ is the average link distance of the mating pool after m iterations,

$$E(\bar{D}_m) = \sum_{i=1}^{N} \bar{d}_i p_{i,m} \tag{4.1}$$

where $\bar{d}_i$ is the average link distance of state i, $p_{i,m}$ is the corresponding probability of staying in state i after m iterations, and N is the total number of states. It is easy to show that when a mating pool has converged, $\bar{D} = 0$. The average link distance of a mating pool provides a numerical value indicating whether the links in the pool are very different from each other.

4.1 Applying to the Two-Link Example

Let us look at the example in Section 3.3 again. Since there are only two links in the mating pool, the number of possible states is $N = 3$. S_1=(1, 1), S_2=(2, 0), S_3=(0, 2). $d_2 = d_3 = 0$. Let $d = d_1$. From equation (4.1), the average link distance is $\bar{D}_m = 0.5^{m+1}d$. If the selection probabilities are different for the two links, $\tilde{p}_0 = (2\omega_1(1-\omega_1), \omega_1^2, (1-\omega_1)^2)$. The probability of getting into the absorbing states is $P(x \in S_{all}) = 1 - (2\omega_1(1-\omega_1))^m$. The expected link distance is $E(\bar{D}_m) = (2\omega_1(1-\omega_1))^{m+1}d$. For $\omega_1 = 0.6$, $P(x \in S_{all}) = 1 - 0.48^m$ and $\bar{D}_m = 0.48^{m+1}d$.

4.2 Average Link Distance of the Initial Mating Pool

Besides the percentage change of the average link distance described above, we need to know the initial average link distance of a mating pool in order to find the average link distances during the iterations. We will start from the largest mating pool (without duplication of links) which is the union of all links formed by all possible permutations of genes. For any link of length l, the total number of gene permutations is $l!$. The average link distance is:

$$\overline{D} = \frac{\sum_{i=1}^{l!}\sum_{j=i}^{l!} d(\tilde{s}_i, \tilde{s}_j)}{C(l!,2)} \tag{4.2}$$

Table 4.1 shows the average link distances calculated for small l's. Since Equation (4.2) involves several factorials, it is technically impossible to calculate the average link distance for a large l. Fortunately, we can use random sampling to create mating pools and find the average link distance from the mating pools to estimate the average link distance of randomly created initial mating pools. Instead of trying to find the average link distance from equation (4.2), we can use a large number of random samples to create initial mating pools with various link lengths and population sizes.

Table 4.1 Average Link Distance from Equation (4.2)

Link Length (l)	Average Link Distance
2	1.00
3	1.40
4	2.10
5	3.03
6	4.01
7	5.00
8	6.00

Figure 4.1 shows the comparisons of the link distance distributions for the calculated distribution from equation (4.2), denoted as Cal. in Figure 4.1, and five sampled data with population size ranging from 40 to 80 links. The link lengths are 5 and 6 respectively for Figure 4.1a and Figure 4.1b. Table 4.2 and 4.3 show the average link distances for all the trials. The average link distances from sampled data are all very close to 3.0 for the case of link length equal to 5. The average link distances from sampled data for the case of link length equal to 6 are all very close to 4.0. The comparisons of sampled data and the calculated data are limited because of the difficulty of finding equation (4.2) for large link length. The average link distance of the largest link length we can find is 6.00 for link length equal to 8. Among the random sampling tests, the larger the link length is, the closer are the average link distances of the sampled data to the calculated results from equation (4.2). It is obvious that the sampled data can be used to estimate the average link distance of the initial mating pools.

According to the tested results, the average link distance is independent of the population size. It is only related to the link length. Figure 4.2 shows the expected average link distances from random sampling. The population sizes of all the sampled mating pools are 100 in Figure 4.2. Table 4.4 shows the numerical values for Figure 4.2. It is interesting that the expected average link distance approximately approaches $l - 2$ for l larger than 4 no matter what the population size and the link length are.

Table 4.2 Average Link Distance for Figure 4.1a

Population Size (*n*)	Average Link Distance
40	3.086
50	3.058
60	3.064
70	3.059
80	3.041

Table 4.3 Average Link Distance for Figure 4.1b

Population Size (*n*)	Average Link Distance
40	3.990
50	4.055
60	4.038
70	4.047
80	4.024

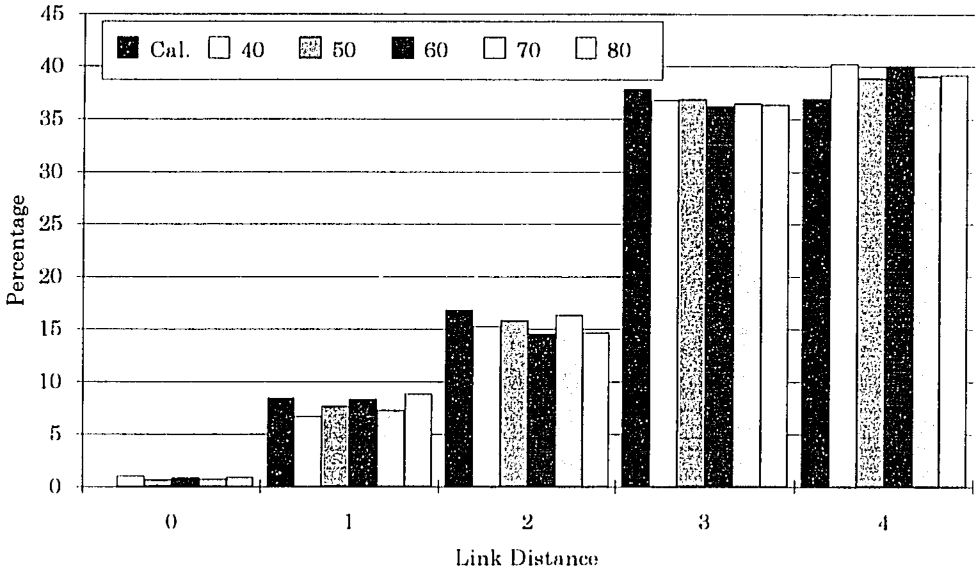

Figure 4.1a Comparison of Distribution of Link Distances, Link Length = 5

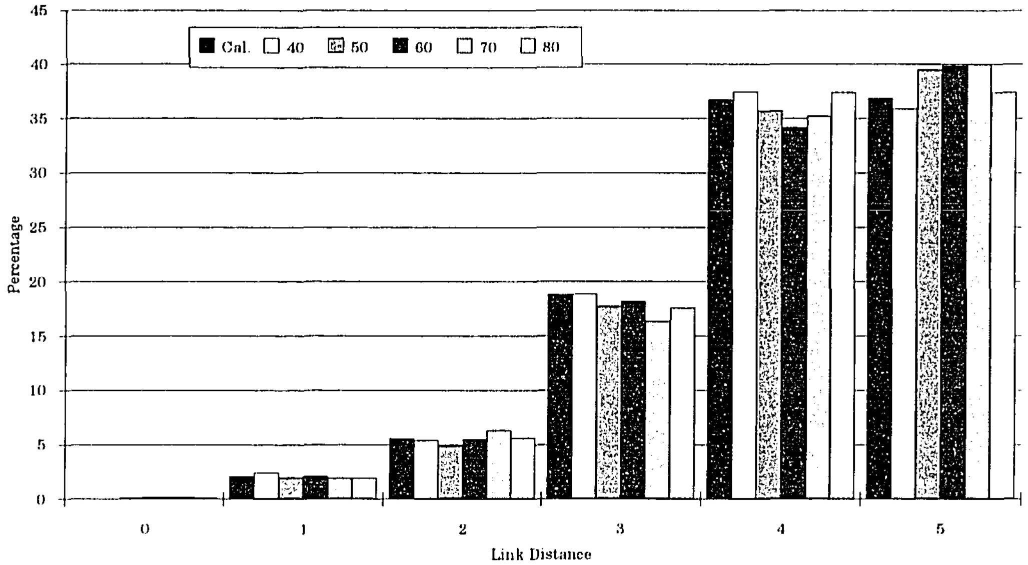

Figure 4.1b Comparison of Distribution of Link Distances, Link Length = 6

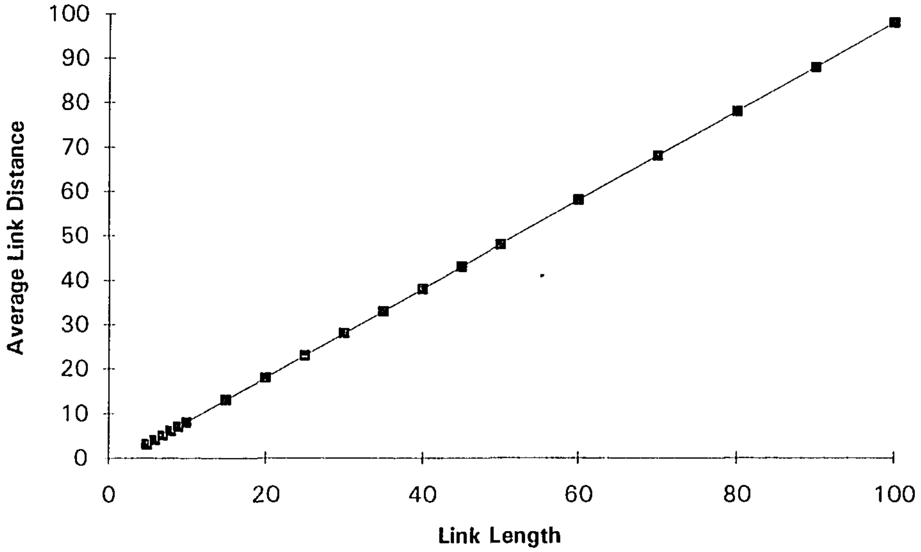

Figure 4.2 Expected Average Link Distance for Random Sampling

Table 4.4 Average Link Distance For Figure 4.2

Link Length (*l*)	Average Link Distance
5	3.021
6	4.000
7	5.019
8	6.027
9	7.006
10	8.024
15	13.005
20	18.007
25	23.001
30	27.970
35	32.995
40	38.004
45	43.015
50	47.975
60	58.000
70	67.998
80	77.996
90	87.972
100	98.007

CHAPTER 5

CHANGE OF AVERAGE LINK DISTANCE DUE TO SELECTION

We have discussed random selection for general genetic algorithms without considering the coding method in the previous chapter. We have also discussed the average link distance of a randomly generated mating pool for order-based genetic algorithm in Chapter 4. Based on the results of Chapter 3 and Chapter 4, we should be able to calculate the expected average link distance due to selection if the Markov chain can be found. In this chapter, we will focus on the effect of random selection to the average link distance of the mating pool. Instead of going through a big Markov probability matrix, which is computationally expensive, a better way of obtaining average link distance is desired. We have made a fundamental analysis leading to the estimation of the average link distance without the use of Markov probability matrix. This will be discussed below. We will use the simple genetic algorithm in Chapter 3 as an example in the discussion.

5.1 Selection Probability of Link Pairs with Link Distance Equal to Zero

Again, we assume that the initial mating pool has no identical links. The initial state is thus S_{x0} = (1, 1, ..., 1). Let n denote the population size of the mating pool. After the m-th selection, the probability of creating link pairs with link distance equal to zero is

$$\sum_{i=1}^{n}\left(\omega_{i,m}\cdot\omega_{i,m}\right) \tag{5.1}$$

where m denotes the number of iterations, and $\omega_{i,m}$ is the probability of link i being selected after m iterations (or selections). For the general case where the initial probabilities may be different for different links, we have

$$\omega_{i,m} = \frac{e_{i,m}\omega_i}{\sum_{l=1}^{n} e_{l,m}\omega_l}$$

Let $p(d_{m+1} = 0)$ denote the probability of creating link pairs with link distance equal to zero after m selections. Assume that the selection probabilities are equal for all the links in the initial mating, that is, $\omega_{i,0} = 1/n$ for all i's. Based on (5.1), we have

$$p(d_{m+1} = 0) = \sum_{i=1}^{n}(\omega_{i,m} \cdot \omega_{i,m}) = \sum_{i=1}^{n}\left(\frac{e_{i,m}}{n}\right)^2 = \frac{1}{n^2}\sum_{i=1}^{n} e_{i,m}^2 \tag{5.2}$$

The total number of link pairs with link distance equal to zero in the mating pool after m selections, denoted by y_m, is

$$y_m = \sum_{i=1}^{n}\frac{e_{i,m}(e_{i,m}-1)}{2} = \frac{1}{2}\sum_{i=1}^{n} e_{i,m}{}^2 - \frac{n}{2} \tag{5.3}$$

We can combine equation (5.2) and (5.3) to obtain

$$p(d_{m+1} = 0) = \frac{1}{n^2}\sum_{i=1}^{n} e_{i,m}{}^2 = \frac{2y_m}{n^2} + \frac{1}{n} \tag{5.4}$$

The expected value of y_m, denoted by $E(y_m)$, is equal to $p(d_m = 0)$ multiplied by the total number of link pairs in the mating pool, i.e.,

$$E(y_m) = p(d_m = 0) \cdot C(n,2) = p(d_m = 0)\frac{n(n-1)}{2} \tag{5.5}$$

The expected value of y_m can be also derived from equation (5.4) as

$$p(d_{m+1} = 0) = \frac{1}{n^2}\sum_{i=1}^{n} e_{i,m}{}^2 = \frac{2E(y_m)}{n^2} + \frac{1}{n} \tag{5.6}$$

Substitute (5.5) to (5.6), we have

$$p(d_{m+1} = 0) = \frac{2E(y_m)}{n^2} + \frac{1}{n} = \frac{n-1}{n}p(d_m = 0) + \frac{1}{n} \tag{5.7}$$

Equation (5.7) is a difference equation of $p(d_m = 0)$. By solving equation (5.7), we have

$$p(d_m = 0) = 1 - \left(\frac{n-1}{n}\right)^m \tag{5.8}$$

By substituting (5.8) in (5.5), the expected number of link pairs with zero link distance is

$$E(y_m) = \frac{n(n-1)}{2}\left[1 - \left(\frac{n-1}{n}\right)^m\right] \tag{5.9}$$

Assume that the expected average link distance of the link pairs whose link distance is non-zero is equal to the average link distance of the initial population. This assumption is reasonable because the average link distance is independent of the population size of a randomly created mating pool, as shown previously in Chapter 4, and the selection process is random. The expected average link distance is then:

$$E(\overline{D}_m) = p(d_m = 0)\cdot 0 + [1 - p(d_m = 0)]\overline{D} = \left(\frac{n-1}{n}\right)^m \overline{D} = \left(1 - \frac{1}{n}\right)^m \overline{D} \qquad (5.10)$$

where $\overline{D}$ denotes the initial average link distance of the mating pool. Since the preferential selection has a higher chance to converge faster than the genetic drift, as described in Chapter 3, equation (5.10) is a upper bound for all selection processes.

From (5.10) the expected reduction of average link distance for each iteration is $1/n$. Equation (5.10) applies to the results in Section 4.1 which were obtained from the same example as that in Chapter 3.3. Since $n = 2$ in this example, the rate of deduction of the average link distance, $1 - 1/n$, is equal to 0.5. Initially the link distance is either 0 or $\overline{d}$. From equation (5.10), the expected average link distance after m iterations is $E(\overline{d}) = 0.5^{m+1}\overline{d}$, which is the same as the expected average link distance with the exact solution from Chapter 4.

5.2 Comparison of Predicted Percentage Deduction of Expected Average Link Distances with Experimental Results

To verify equation (5.10), we designed some numerical experiments with random sampling and random selection. We tested various simple order-based genetic algorithms for different link lengths and population sizes. Figure 5.1 shows the average reduction ratio of the average link distance per iteration for the first ten iterations of some of the tests with the same population size, which is 20, but with the link length varied from 10, 20, 30, 40, 50, 60, 70, 80, 90, to 100. Since the expected average link distance change per iteration is $1/n = 0.05$, the ratio in Figure 5.1 is expected to be 5%. Figure 5.2 shows the average of the deduction ratios of the average link distances from the first ten iterations of the tests with the same link length, which is 10, but with the population size varied from 10, 20, 30, 40, 50, 60, 70, 80, 90 to 100. The dotted line in Figure 5.2 shows the expected ratio from equation (5.10). All the test results are very close to the expected average link distances calculated from equation (5.10). Figure 5.3 shows how the average link distance changes during the iterations from one of the above tests where the link length is 10 and the population size is 100. The dotted line in Figure 5.3 is the expected average link distance change from equation (5.10). Again there is a very good agreement.

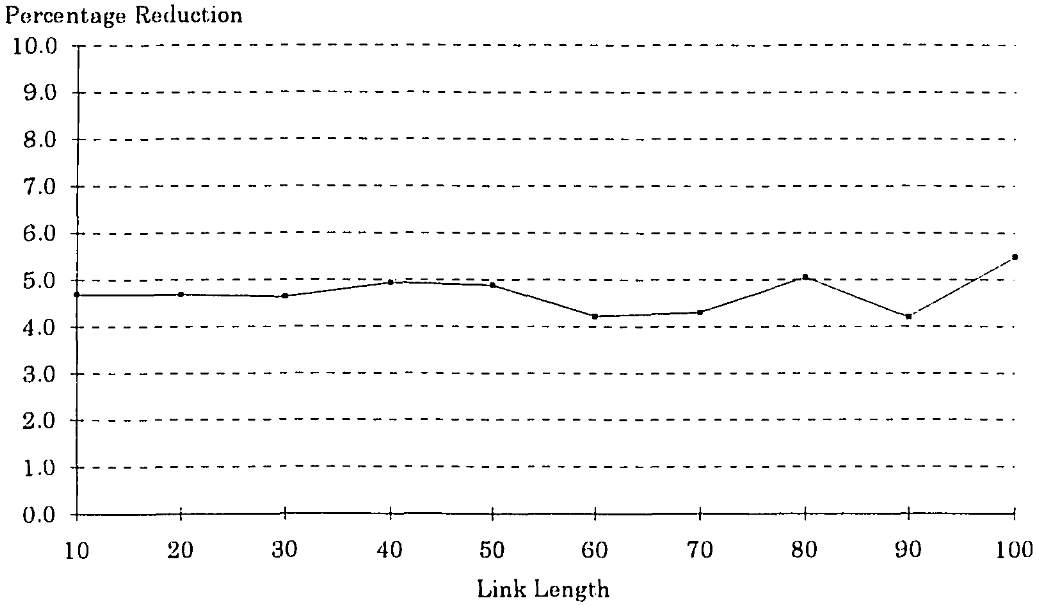

Figure 5.1 Percentage Deduction For Various Link Length, Population Size = 20

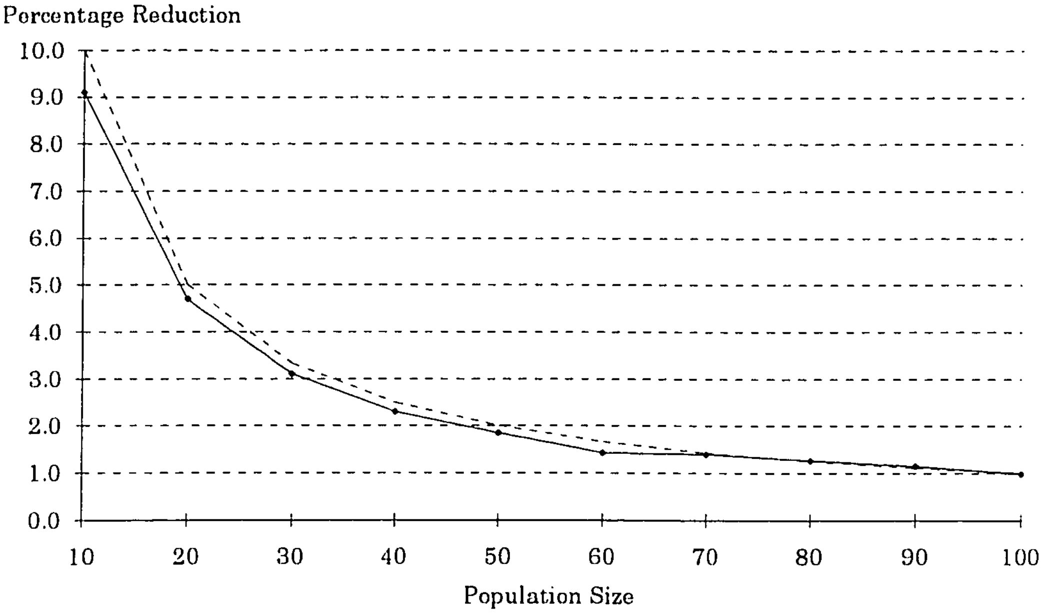

Figure 5.2 Percentage Deduction for Various Population Size,
Link Length = 10

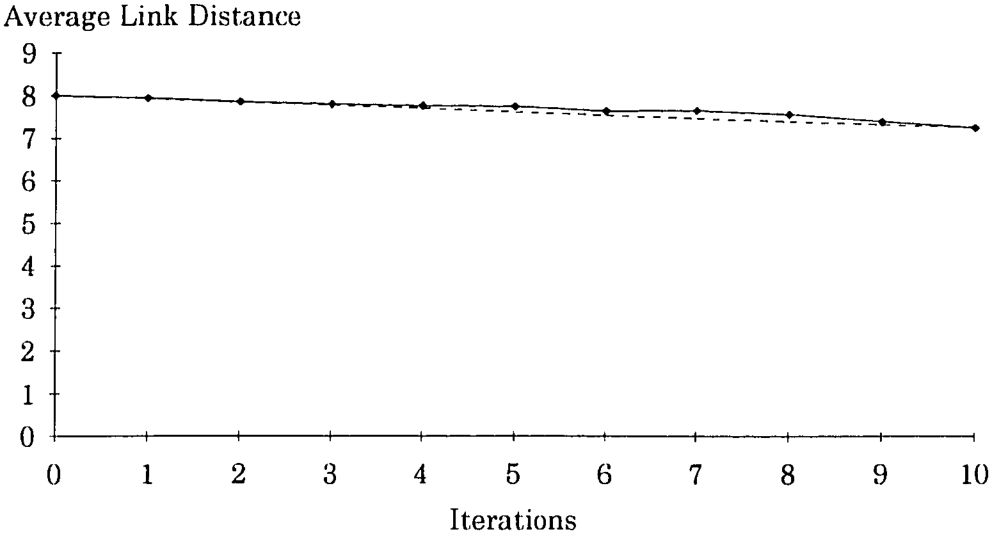

Figure 5.3 Average Link Distance During the Iterations,
Population Size = 100, Link Length = 10

CHAPTER 6

EFFECT OF MUTATION OPERATOR

We did not include any operator in the discussion of the simple genetic algorithm in Chapter 5. In this chapter, we will introduce the *mutation operator* and study its effect on the expected average link distance for the order-based genetic algorithm. The combined effect of the selection and the mutation operator will be discussed in chapter 7. We first provide the definition for the mutation operator.

Definition 17: A mutation operator is a mapping from a *parent* link to an *offspring* link as follows: Let O_m denote the mutation operator, $O_m(\tilde{s}) = \tilde{s}'$, where $\tilde{s} = \{s_1, s_2, \ldots, s_n\}$, $\tilde{s}' = \{s'_1, s'_2, \ldots, s'_n\}$. There exist i and j, $i \neq j$, $1 \leq i, j \leq n$, such that $s_k = s'_k$, $1 \leq k \leq n$, $k \neq i, j$, and $s_i = s'_j$, $s_j = s'_i$.

With the mutation operator, the genetic algorithm becomes:

1. Randomly generate a set of initial parents, U_n, that forms a mating pool M_n^t, where t represents the number of iterations and $t = 0$ at the beginning. Decide on the selection probability, ω_i, of the links. Select a number between 0 and 1 for the *mutation operator rate*, denoted as r.
2. Select n links (with replacement) according to the selection probabilities of the links from the mating pool M_n^t to form another mating pool M_n^{t+1}. Let the selection probability ω'_i of link i in the mating pool M_n^{t+1} be equal to

$$\omega_{i,m} = \frac{e_{it}\omega_i}{\sum_{l=1}^{n} e_{lt}\omega_l}.$$

3. For each link in M_n^{t+1}, attach a random drawing from a number between 0 and 1. If the drawing is less than r, apply the mutation operator to the link to create a new link and replace the original link in the mating pool M_n^{t+1} with it.
4. Let $t = t + 1$.
5. Repeat steps 2, 3, and 4 until the process has reached a pre-specified number of iterations.

For each link distance, there are two links associated with it. After the mutation operator is applied to the mating pool, there are three possible cases: (1) none of the links has been changed, (2) one of the links has been changed, and (3) both of the links have been changed. The effect of mutation operator to the link pairs with zero link distance is different from that to the other link pairs. We will discuss the effects of the mutation operator to the link pairs with non-zero link distances in Section 6.1 and the link pairs with zero link distances in Section 6.2.

6.1 Link Pairs with Non-Zero Link Distance

Let the link distance $d_{ij} = d(\tilde{s}_i, \tilde{s}_j)$, where $\tilde{s}_i$, $\tilde{s}_j$ are two different links arbitrarily selected from a mating pool. Let d'_{ij} denote the link distance after the operator is applied. For those link pairs with link distance greater than zero but smaller than 4, the sub-cases (to be discussed below) that involve the reduction of link distance greater than the link distance itself are simply impossible and therefore the probability is 0 for them.

The effect of any operator on the link distance d_{ij} can be summarized in the following:

Case 1: None of the links is affected by the operator, Case 2: Only one of the links is affected by the operator,

Case 3: Both of the links are affected by the operator.

Let r denote the operator rate. The individual probabilities for the three cases are:

Case 1: $p_1 = (1-r)^2$

Case 2: $p_2 = 2r(1-r)$

Case 3: $p_3 = r^2$

The effect of mutation on the link distance for each case is given as follows:

Case 1: no effect. $d'_{ij} = d_{ij}$.

Case 2:

Let the genes of the loci selected for exchange be C and D for the changed link and they correspond to A and B for the unchanged link. It is obvious that A ≠ B and C ≠ D since they are permutation links. Figure 6.1 shows the relationship between the links.

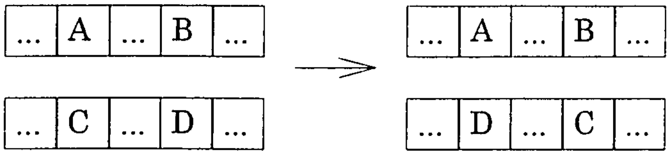

Figure 6.1 Relationship between the links of a link pair

The possible outcomes for Case 2 are:

Sub–case 1: A = C and B = D. $d'_{ij} = d_{ij} + 2$.

Sub–case 2: (A = C and B ≠ D) or (A ≠ C and B = D). $d'_{ij} = d_{ij} + 1$.

Sub–case 3: (A ≠ C and B ≠ D) and (A = D or B = C but not both), $d'_{ij} = d_{ij} - 1$.

Sub–case 4: A = D and B = C. $d'_{ij} = d_{ij} - 2$.

Sub–case 5: A ≠ C ≠ B ≠ D. $d'_{ij} = d_{ij}$.

For sub-case 1 above, the probability of the sub-case being true is equal to the probability of selecting two objects from l objects such that these two selected objects are from the group of identical gene pairs. Since the link distance befor3 the operation is d_{ij}, there are d_{ij} +1 different gene pairs and l–d_{ij}–1 identical gene pairs. So, the probability is C(d_{ij}+1, 0)•C(l–d_{ij}–1, 2) / C(l, 2). For all the other sub-cases, we will not describe the derivation but only show the results.

The individual probabilities for the sub–cases of Case 2 are:

Sub–case 1: C(d_{ij}+1, 0)•C(l–d_{ij}–1, 2) / C(l, 2)

Sub–case 2: C(d_{ij}+1, 1)•C(l–d_{ij}–1, 1) / C(l, 2)

Sub–case 3: [C(d_{ij}+1, 2)•C(l–d_{ij}–1, 0) / C(l, 2)] • 2(1/(l–2))[1–(1/(l–2))]

Sub–case 4: [C(d_{ij}+1, 2)•C(l–d_{ij}–1, 0) / C(l, 2)] • (1/(l–2))(1/(l–2))

Sub–case 5: [C(d_{ij}+1, 2)•C(l–d_{ij}–1, 0) / C(l, 2)] • [1–(1/(l–2))][1–(1/(l–2))]

Case 3:

Let the genes of the loci selected for exchange be A and B for the first link and C and D for the second link. The loci selected for link one and the loci selected for link two have three possible relations:

Sub–case 1: Both of the loci are at the same place (Figure 6.2). The probability is 1/C(l, 2).

Sub–case 2: Only one of the loci is at the same place (Figure 6.3). The probability is 2lC(l–1, 2)/[C(l, 2)•C(l, 2)]

Sub–case 3: Both of the loci are at different places (Figure 6.4). The probability is C(l–2, 2)/C(l, 2)

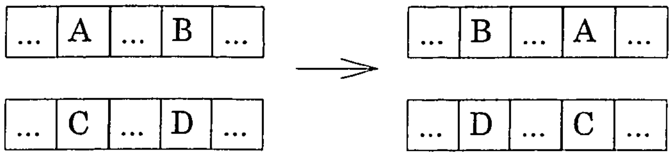

Figure 6.2 Relationship of the links of a link pair before and after the mutation operation.

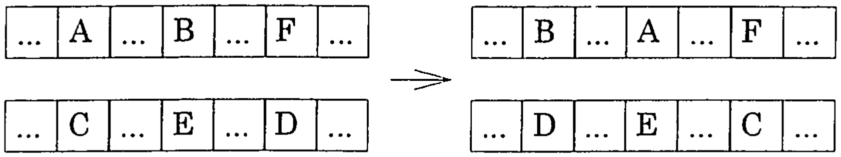

Figure 6.3 Relationship of the links of a link pair before and after the mutation operation.

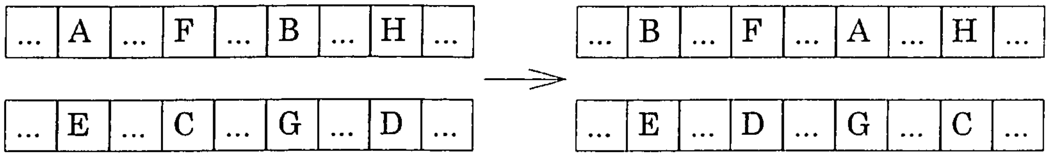
... A ... F ... B ... H ...
... E ... C ... G ... D ...
... B ... F ... A ... H ...
... E ... D ... G ... C ...

Figure 6.4 Relationship of the links of a link pair before and after the mutation operation.

The possible outcomes for the three sub-cases of Case 3 are:

Sub-case 1: There is no change for the link distance. $d'_{ij} = d_{ij}$.

Sub-case 2:

Let E denote the gene at the corresponding locus of B in the second link and F denote the gene at the corresponding locus of D in the first link. The relationships of the two links before and after the mutation operations are illustrated in Figure 6.3.

The possible situations and outcomes on the link distance are:

Sub-case 2.1: A = C, B = E, D = F, B ≠ D, A ≠ E, and C ≠ F. $d'_{ij} = d_{ij}+3$.

Sub-case 2.2: A = C, B = E, D ≠ F, B ≠ D, A ≠ E, and C ≠ F. $d'_{ij} = d_{ij}+2$.

Sub-case 2.3: A = C, B ≠ E, D = F, B ≠ D, A ≠ E, and C ≠ F. $d'_{ij} = d_{ij}+2$.

Sub-case 2.4: A = C, B ≠ E, D ≠ F, B = D, A ≠ E, and C ≠ F. $d'_{ij} = d_{ij}$.

Sub-case 2.5: A = C, B ≠ E, D ≠ F, B ≠ D, A ≠ E, and C ≠ F. $d'_{ij} = d_{ij}-1$.

Sub-case 2.6: A ≠ C, B = E, D = F, B ≠ D, A ≠ E, and C ≠ F. $d'_{ij} = d_{ij}+2$.

Sub-case 2.7: A ≠ C, B = E, D ≠ F, B ≠ D, A ≠ E, and C = F. $d'_{ij} = d_{ij}$.

Sub-case 2.8: A ≠ C, B = E, D ≠ F, B ≠ D, A ≠ E, and C ≠ F. $d'_{ij} = d_{ij}+1$.

Sub-case 2.9: A ≠ C, B ≠ E, D = F, B ≠ D, A = E, and C ≠ F. $d'_{ij} = d_{ij}$.

Sub-case 2.10: A ≠ C, B ≠ E, D = F, B ≠ D, A ≠ E, and C ≠ F. $d'_{ij} = d_{ij}+1$.

Sub-case 2.11: A ≠ C, B ≠ E, D ≠ F, B = D, A = E, and C = F. $d'_{ij} = d_{ij}-3$.

Sub-case 2.12: A ≠ C, B ≠ E, D ≠ F, B = D, A = E, and C ≠ F. $d'_{ij} = d_{ij}-2$.

Sub-case 2.13: A ≠ C, B ≠ E, D ≠ F, B = D, A ≠ E, and C = F. $d'_{ij} = d_{ij}-2$.

Sub-case 2.14: A ≠ C, B ≠ E, D ≠ F, B = D, A ≠ E, and C ≠ F. $d'_{ij} = d_{ij}-1$.

Sub-case 2.15: A ≠ C, B ≠ E, D ≠ F, B ≠ D, A = E, and C = F. $d'_{ij} = d_{ij}-2$.

Sub-case 2.16: A ≠ C, B ≠ E, D ≠ F, B ≠ D, A = E, and C ≠ F. $d'_{ij} = d_{ij}-1$.

Sub-case 2.17: A ≠ C, B ≠ E, D ≠ F, B ≠ D, A ≠ E, and C = F. $d'_{ij} = d_{ij}-1$.

Sub-case 2.18: A ≠ C, B ≠ E, D ≠ F, B ≠ D, A ≠ E, and C ≠ F. $d'_{ij} = d_{ij}$.

For sub-case 2.1 above, the probability of the sub-case being true is equal to the probability of having three identical gene pairs {A, C}, {B, E}, and {D, F}. The condition of the sub-case is that {A, C} pair is from the same loci. The probability of the sub-case is equal to the probability of selecting one gene pair from the identical gene pairs, and then selecting two other gene pairs also from the identical gene pairs. The probability of selecting a gene, A, among l genes such that the gene pair {A, C} is identical is $C(l-d_{ij}-1, 1) / C(l, 1)$. The probability of selecting the other two genes, B and D, among $l - 1$ genes such that the gene pairs {B, E} and {D, F} are identical is $C(l-d_{ij}-2, 2) / C(l-1, 2)$. According to the rule of conditional probability, Prob{A = C and B = E and D = F} = Prob{A = C} · Prob{ B = E and D = F | A = C}. The probability of sub-case 2.1 is $C(l-d_{ij}-1, 1) \cdot C(l-d_{ij}-2, 2) / [C(l, 1) \cdot C(l-1, 2)]$. For all the other sub-cases, again we will not describe the derivation but only show the results.

The probabilities of the situations of sub-case 2 are:

Sub-case 2.1: $C(l-d_{ij}-1, 1) \cdot C(l-d_{ij}-2, 2) / [C(l, 1) \cdot C(l-1, 2)]$

Sub-case 2.2: $C(l-d_{ij}-1, 1) \cdot [C(l-d_{ij}-2, 1) \cdot C(d_{ij}+1, 1)/2] / [C(l, 1) \cdot C(l-1, 2)]$

Sub-case 2.3: $C(l-d_{ij}-1, 1) \cdot [C(l-d_{ij}-2, 1) \cdot C(d_{ij}+1, 1)/2] / [C(l, 1) \cdot C(l-1, 2)]$

Sub-case 2.4: $[C(l-d_{ij}-1, 1) \cdot C(d_{ij}+1, 2) / [C(l, 1) \cdot C(l-1, 2)]] \cdot (1/(l-2))$

Sub-case 2.5: $[C(l-d_{ij}-1, 1) \cdot C(d_{ij}+1, 2) / [C(l, 1) \cdot C(l-1, 2)]] \cdot [1-(1/(l-2))]$

Sub-case 2.6: $C(l-d_{ij}-1, 2) \cdot C(d_{ij}+1, 1) / [C(l, 1) \cdot C(l-1, 2)]$

Sub-case 2.7: $[C(d_{ij}+1, 1) \cdot [C(l-d_{ij}-1, 1) \cdot C(d_{ij}, 1)/2] / [C(l, 1) \cdot C(l-1, 2)]] \cdot (1/(l-2))$

Sub-case 2.8: $[C(d_{ij}+1, 1) \cdot [C(l-d_{ij}-1, 1) \cdot C(d_{ij}, 1)/2] / [C(l, 1) \cdot C(l-1, 2)]] \cdot [1-(1/(l-2))]$

Sub-case 2.9: $[C(d_{ij}+1, 1) \cdot [C(l-d_{ij}-1, 1) \cdot C(d_{ij}, 1)/2] / [C(l, 1) \cdot C(l-1, 2)]] \cdot (1/(l-2))$

Sub-case 2.10: $[C(d_{ij}+1, 1) \cdot [C(l-d_{ij}-1, 1) \cdot C(d_{ij}, 1)/2] / [C(l, 1) \cdot C(l-1, 2)]] * [1-(1/(l-2))]$

Sub-case 2.11: $[C(d_{ij}+1,\ 1)\cdot C(d_{ij},\ 2)\ /\ [C(l,\ 1)\cdot C(l-1,\ 2)]]\cdot (1/(l-1))\cdot (1/(l-2))\cdot (1/(l-2))$

Sub-case 2.12: $[C(d_{ij}+1,\ 1)\cdot C(d_{ij},\ 2)\ /\ [C(l,\ 1)\cdot C(l-1,\ 2)]]\cdot (1/(l-1))\cdot (1/(l-2))\cdot [1-(1/(l-2))]$

Sub-case 2.13: $[C(d_{ij}+1,\ 1)\cdot C(d_{ij},\ 2)\ /\ [C(l,\ 1)\cdot C(l-1,\ 2)]]\cdot (1/(l-1))\cdot [1-(1/(l-2))]\cdot [(1/(l-2))]$

Sub-case 2.14: $[C(d_{ij}+1,\ 1)\cdot C(d_{ij},\ 2)\ /\ [C(l,\ 1)\cdot C(l-1,\ 2)]]\cdot (1/(l-1))\cdot [1-(1/(l-2))]\cdot [1-(1/(l-2))]$

Sub-case 2.15: $[C(d_{ij}+1,\ 1)\cdot C(d_{ij},\ 2)\ /\ [C(l,\ 1)\cdot C(l-1,\ 2)]]\cdot [1-(1/(l-1))]\cdot (1/(l-2))\cdot (1/(l-2))$

Sub-case 2.16: $[C(d_{ij}+1,\ 1)\cdot C(d_{ij},\ 2)\ /\ [C(l,\ 1)\cdot C(l-1,\ 2)]]\cdot [1-(1/(l-1))]\cdot (1/(l-2))\cdot [1-(1/(l-2))]$

Sub-case 2.17: $[C(d_{ij}+1,\ 1)\cdot C(d_{ij},\ 2)\ /\ [C(l,\ 1)\cdot C(l-1,\ 2)]]\cdot [1-(1/(l-1))]\cdot [1-(1/(l-2))]\cdot (1/(l-2))$

Sub-case 2.18: $[C(d_{ij}+1,\ 1)\cdot C(d_{ij},\ 2)\ /\ [C(l,\ 1)\cdot C(l-1,\ 2)]]\cdot [1-(1/(l-1))]\cdot [1-(1/(l-2))]\cdot [1-(1/(l-2))]$

Sub-case 3:

Let E and G denote the genes in the second link which are at the corresponding loci of A and B, F and H denote the genes in the first link which are at the corresponding loci of C and D in the second link. The relationships for the two links before and after the mutation operations are illustrated in Figure 6.4.

The effect of the mutation operator on link distance can be studied according to the relationships between the four gene pairs before the operation, i.e. (A, E), (B, G), (C, F), and (D, H), and the four gene pairs after the operation, i.e. (A, G), (B, E), (C, H), and (D, F). The possible situations are:

Sub-case 3.1: All the four original gene pairs are the same. In this case all the four modified gene pairs after the operation are different. $d'_{ij} = d_{ij}+4$.

Sub-case 3.2: Three of the four original gene pairs are the same and the other one is different. This different pair can be one of {(A, E), (B, G)} pair or one of {(C, F), (D, H)}. All the four modified gene pairs are different. $d'_{ij} = d_{ij}+3$.

Sub-case 3.3: (A = E and B = G) and (C ≠ F and D ≠ H) and (C = H and D = F). $d'_{ij} = d_{ij}$.

Sub-case 3.4: (A = E and B = G) and (C ≠ F and D ≠ H) and (either (C = H, D ≠ F) or (C ≠ H, D = F)). $d'_{ij} = d_{ij}+1$.

Sub-case 3.5: (A = E and B = G) and (C ≠ F and D ≠ H) and (C ≠ H and D ≠ F). $d'_{ij} = d_{ij}+2$.

Sub-case 3.6: (C = F and D = H) and (A ≠ E and B ≠ G) and (A = G and B = E). $d'_{ij} = d_{ij}$.

Sub-case 3.7: (C = F and D = H) and (A ≠ E and B ≠ G) and (either (A = G, B ≠ E) or (A ≠ G, B = E)). $d'_{ij} = d_{ij}+1$.

Sub-case 3.8: (C = F and D = H) and (A ≠ E and B ≠ G) and (A ≠ G and B ≠ E). $d'_{ij} = d_{ij}+2$.

Sub-case 3.9: (either (A = E, B ≠ G) or (A ≠ E, B = G)) and (either (C = F, D ≠ H) or (C ≠ F, D = H)). $d'_{ij} = d_{ij}+2$.

Sub-case 3.10: Only one of the four original gene pairs is the same, and all the others are different. This same pair can be one of {(A, E), (B, G)} pair or one of {(C, F), (D, H)}. In the modified gene pairs, the two gene pairs which do not contain either of the two genes from the same gene pair before the modification are the same. For example, A ≠ E, B ≠ G, C ≠ F, D = H, A = G, B = E, C ≠ H, D ≠ F. $d'_{ij} = d_{ij}-1$.

Sub–case 3.11: Only one of the four original gene pairs is the same, and all the others are different. This same pair can be one of {(A, E), (B, G)} pair or one of {(C, F), (D, H)}. In the modified gene pairs, one of the two gene pairs which do not contain either of the two genes from the same gene pair before the modification is the same and the other is different. For example, A ≠ E, B ≠ G, C ≠ F, D = H, A = G, B ≠ E, C ≠ H, D ≠ F. $d'_{ij} = d_{ij}$.

Sub–case 3.12: Only one of the four original gene pairs is the same, all the others are different. This same pair can be one of either {(A, E), (B, G)} pair or one of {(C, F), (D, H)}. In the modified gene pairs, both of the two gene pairs which do not contain either of the two genes from the same gene pair before the modification are different. For example, A ≠ E, B ≠ G, C ≠ F, D = H, A ≠ G, B ≠ E, C ≠ H, D ≠ F. $d'_{ij} = d_{ij}+1$.

Sub–case 3.13: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A = G, B = E, C = H, D = F. $d'_{ij} = d_{ij}-4$.

Sub–case 3.14: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A = G, B = E, C = H, D ≠ F. $d'_{ij} = d_{ij}-3$.

Sub–case 3.15: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A = G, B = E, C ≠ H, D = F. $d'_{ij} = d_{ij}-3$.

Sub–case 3.16: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A = G, B = E, C ≠ H, D ≠ F. $d'_{ij} = d_{ij}-2$.

Sub–case 3.17: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A = G, B ≠ E, C = H, D = F. $d'_{ij} = d_{ij}-1$.

Sub–case 3.18: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A = G, B ≠ E, C = H, D ≠ F.

$d'_{ij} = d_{ij}-2$.

Sub–case 3.19: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A = G, B ≠ E, C ≠ H, D = F.

$d'_{ij} = d_{ij}-2$.

Sub–case 3.20: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A = G, B ≠ E, C ≠ H, D ≠ F.

$d'_{ij} = d_{ij}-1$.

Sub–case 3.21: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A ≠ G, B = E, C = H, D = F.

$d'_{ij} = d_{ij}-3$.

Sub–case 3.22: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A ≠ G, B = E, C = H, D ≠ F.

$d'_{ij} = d_{ij}-2$.

Sub–case 3.23: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A ≠ G, B = E, C ≠ H, D = F.

$d'_{ij} = d_{ij}-2$.

Sub–case 3.24: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A ≠ G, B = E, C ≠ H, D ≠ F.

$d'_{ij} = d_{ij}-1$.

Sub–case 3.25: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A ≠ G, B ≠ E, C = H, D = F.

$d'_{ij} = d_{ij}-2$.

Sub–case 3.26: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A ≠ G, B ≠ E, C = H, D ≠ F.

$d'_{ij} = d_{ij}-1$.

Sub–case 3.27: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A ≠ G, B ≠ E, C ≠ H, D = F.

$d'_{ij} = d_{ij}-1$.

Sub–case 3.28: A ≠ E, B ≠ G, C ≠ F, D ≠ H, A ≠ G, B ≠ E, C ≠ H, D ≠ F.

$d'_{ij} = d_{ij}$.

We will show the derivation the sub-case 3.10 in Appendix B since the other sub-cases are either easier to be obtained or easy to follow after sub-case 3.10.

The probabilities of the above situations are:

Sub-case 3.1: $C(d_{ij}+1,\ 0)\cdot C(l-d_{ij}-1,\ 2)\cdot C(d_{ij}+1,\ 0)\cdot C(l-d_{ij}-3,\ 2)\ /\ [C(l,\ 2)\cdot C(l-2,\ 2)]$.

Sub-case 3.2: $2\cdot C(d_{ij}+1,\ 0)\cdot C(l-d_{ij}-1,\ 2)\cdot C(d_{ij}+1,\ 1)\cdot C(l-d_{ij}-3,\ 1)\ /\ [C(l,\ 2)\cdot C(l-2,\ 2)]$.

Sub-case 3.3: $[C(d_{ij}+1,\ 0)\cdot C(l-d_{ij}-1,\ 2)\cdot C(d_{ij}+1,\ 2)\cdot C(l-d_{ij}-3,\ 0)\ /\ [C(l,\ 2)\cdot C(l-2,\ 2)]]\cdot(1/(l-3))\cdot(1/(l-3))$.

Sub-case 3.4: $[C(d_{ij}+1,\ 0)\cdot C(l-d_{ij}-1,\ 2)\cdot C(d_{ij}+1,\ 2)\cdot C(l--d_{ij}-3,\ 0)\ /\ [C(l,\ 2)\cdot C(l-2,\ 2)]]\cdot 2\cdot(1/(l-3))\cdot[1-(1/(l-3))]$.

Sub-case 3.5: $[C(d_{ij}+1,\ 0)\cdot C(l-d_{ij}-1,\ 2)\cdot C(d_{ij}+1,\ 2)\cdot C(l-d_{ij}-3,\ 0)\ /\ [C(l,\ 2)\cdot C(l-2,\ 2)]]\cdot[1-(1/(l-3))]\cdot[1-(1/(l-3))]$.

Sub-case 3.6: $[C(d_{ij}+1,\ 2)\cdot C(l-d_{ij}-1,\ 0)\cdot C(d_{ij}-1,\ 0)\cdot C(l-d_{ij}-1,\ 2)\ /\ [C(l,\ 2)\cdot C(l-2,\ 2)]]\cdot(1/(l-3))\cdot(1/(l-3))$.

Sub-case 3.7: $[C(d_{ij}+1,\ 2)\cdot C(l-d_{ij}-1,\ 0)\cdot C(d_{ij}-1,\ 0)\cdot C(l-d_{ij}-1,\ 2)\ /\ [C(l,\ 2)\cdot C(l-2,\ 2)]]\cdot 2\cdot(1/(l-3))\cdot[1-(1/(l-3))]$.

Sub-case 3.8: $[C(d_{ij}+1,\ 2)\cdot C(l-d_{ij}-1,\ 0)\cdot C(d_{ij}-1,\ 0)\cdot C(l-d_{ij}-1,\ 2)\ /\ [C(l,\ 2)\cdot C(l-2,\ 2)]]\cdot[1-(1/(l-3))]\cdot[1-(1/(l-3))]$.

Sub-case 3.9: $C(d_{ij}+1,\ 1)\cdot C(l-d_{ij}-1,\ 1)\cdot C(d_{ij},\ 1)\cdot C(l-d_{ij}-2,\ 1)\ /\ [C(l,\ 2)\cdot C(l-2,\ 2)]$.

Sub-case 3.10: $2\cdot[C(d_{ij}+1,\ 2)\cdot C(l-d_{ij}-1,\ 0)\cdot C(d_{ij}-1,\ 1)\cdot C(l-d_{ij}-1,\ 1)\ /\ [C(l,\ 2)\cdot C(l-2,\ 2)]\cdot(l-4)\ /\ (l-2)(l-3)^2$.

Sub-case 3.11: $2\cdot[C(d_{ij}+1,\ 2)\cdot C(l-d_{ij}-1,\ 0)\cdot C(d_{ij}-1,\ 1)\cdot C(l-d_{ij}-1,\ 1)\ /\ [C(l,\ 2)\cdot C(l-2,\ 2)]\cdot[[1-1/(l-2)]\cdot(l-4)/(l-3)^2 + 1/(l-2)\cdot[1-(l-4)/(l-3)^2]]$.

Sub-case 3.12: $2\cdot[C(d_{ij}+1,\ 2)\cdot C(l-d_{ij}-1,\ 0)\cdot C(d_{ij}-1,\ 1)\cdot C(l-d_{ij}-1,\ 1)\ /\ [C(l,\ 2)\cdot C(l-2,\ 2)]\cdot[1-1/(l-2)]\cdot[1-(l-4)/(l-3)^2]$.

Sub-case 3.13: $[C(d_{ij}+1,\ 2)\cdot C(l-d_{ij}-1,\ 0)\cdot C(d_{ij}-1,\ 2)\cdot C(l-d_{ij}-1,\ 0)\ /\ [C(l,\ 2)\cdot C(l-2,\ 2)]\cdot[1/(l-1)^2]\cdot[1/(l-3)^2]$.

Sub-case 3.14: $[C(d_{ij}+1, 2)\cdot C(l-d_{ij}-1, 0)\cdot C(d_{ij}-1, 2)\cdot C(l-d_{ij}-1, 0) / [C(l, 2)\cdot C(l-2, 2)]\cdot[1/(l-1)^2]\cdot[1/(l-3)]\cdot[1-1/(l-3)]$.

Sub-case 3.15: $[C(d_{ij}+1, 2)\cdot C(l-d_{ij}-1, 0)\cdot C(d_{ij}-1, 2)\cdot C(l-d_{ij}-1, 0) / [C(l, 2)\cdot C(l-2, 2)]\cdot[1/(l-1)^2]\cdot[1-1/(l-3)]\cdot[1/(l-3)]$

Sub-case 3.16: $[C(d_{ij}+1, 2)\cdot C(l-d_{ij}-1, 0)\cdot C(d_{ij}-1, 2)\cdot C(l-d_{ij}-1, 0) / [C(l, 2)\cdot C(l-2, 2)]\cdot[1/(l-1)^2]\cdot[1-1/(l-3)]^2$.

Sub-case 3.17: $[C(d_{ij}+1, 2)\cdot C(l-d_{ij}-1, 0)\cdot C(d_{ij}-1, 2)\cdot C(l-d_{ij}-1, 0) / [C(l, 2)\cdot C(l-2, 2)]\cdot[1/(l-1)]\cdot[1-1/(l-1)]\cdot[1/(l-3)^2]$.

Sub-case 3.18: $[C(d_{ij}+1, 2)\cdot C(l-d_{ij}-1, 0)\cdot C(d_{ij}-1, 2)\cdot C(l-d_{ij}-1, 0) / [C(l, 2)\cdot C(l-2, 2)]\cdot[1/(l-1)]\cdot[1-1/(l-1)]\cdot[1/(l-3)]\cdot[1-1/(l-3)]$.

Sub-case 3.19: $[C(d_{ij}+1, 2)\cdot C(l-d_{ij}-1, 0)\cdot C(d_{ij}-1, 2)\cdot C(l-d_{ij}-1, 0) / [C(l, 2)\cdot C(l-2, 2)]\cdot[1/(l-1)]\cdot[1-1/(l-1)]\cdot[1-1/(l-3)]\cdot[1/(l-3)]$.

Sub-case 3.20: $[C(d_{ij}+1, 2)\cdot C(l-d_{ij}-1, 0)\cdot C(d_{ij}-1, 2)\cdot C(l-d_{ij}-1, 0) / [C(l, 2)\cdot C(l-2, 2)]\cdot[1/(l-1)]\cdot[1-1/(l-1)]\cdot[1-1/(l-3)]^2$.

Sub-case 3.21: $[C(d_{ij}+1, 2)\cdot C(l-d_{ij}-1, 0)\cdot C(d_{ij}-1, 2)\cdot C(l-d_{ij}-1, 0) / [C(l, 2)\cdot C(l-2, 2)]\cdot[1-1/(l-1)]\cdot[1/(l-1)]\cdot[1/(l-3)^2]$.

Sub-case 3.22: $[C(d_{ij}+1, 2)\cdot C(l-d_{ij}-1, 0)\cdot C(d_{ij}-1, 2)\cdot C(l-d_{ij}-1, 0) / [C(l, 2)\cdot C(l-2, 2)]\cdot[1-1/(l-1)]\cdot[1/(l-1)]\cdot[1/(l-3)]\cdot[1-1/(l-3)]$.

Sub-case 3.23: $[C(d_{ij}+1, 2)\cdot C(l-d_{ij}-1, 0)\cdot C(d_{ij}-1, 2)\cdot C(l-d_{ij}-1, 0) / [C(l, 2)\cdot C(l-2, 2)]\cdot[1-1/(l-1)]\cdot[1/(l-1)]\cdot[1-1/(l-3)]\cdot[1/(l-3)]$.

Sub-case 3.24: $[C(d_{ij}+1, 2)\cdot C(l-d_{ij}-1, 0)\cdot C(d_{ij}-1, 2)\cdot C(l-d_{ij}-1, 0) / [C(l, 2)\cdot C(l-2, 2)]\cdot[1-1/(l-1)]\cdot[1/(l-1)]\cdot[1-1/(l-3)]^2$.

Sub-case 3.25: $[C(d_{ij}+1, 2)\cdot C(l-d_{ij}-1, 0)\cdot C(d_{ij}-1, 2)\cdot C(l-d_{ij}-1, 0) / [C(l, 2)\cdot C(l-2, 2)]\cdot[1-1/(l-1)]^2\cdot[1/(l-3)^2]$.

Sub-case 3.26: $[C(d_{ij}+1, 2)\cdot C(l-d_{ij}-1, 0)\cdot C(d_{ij}-1, 2)\cdot C(l-d_{ij}-1, 0) / [C(l, 2)\cdot C(l-2, 2)]\cdot[1-1/(l-1)]^2\cdot[1/(l-3)]\cdot[1-1/(l-3)]$.

Sub-case 3.27: [C(d_{ij}+1, 2)•C(l−d_{ij}−1, 0)•C(d_{ij}−1, 2)•C(l−d_{ij}−1, 0) / [C(l, 2)•C(l−2, 2)]•[1−1/(l−1)]^2•[1−1/(l−3)]•[1/(l−3)].

Sub-case 3.28: [C(d_{ij}+1, 2)•C(l−d_{ij}−1, 0)•C(d_{ij}−1, 2)•C(l−d_{ij}−1, 0) / [C(l, 2)•C(l−2, 2)]•[1−1/(l−1)]^2•[1−1/(l−3)]^2.

The link distance change is $\Delta d_{ij} = d'_{ij} - d_{ij}$. The expected change of average link distance, denoted as $E_{nonzero}(\Delta\bar{d})$, is equal to the sum of the expected change of the link distance for each of the various cases multiplied by the corresponding probability for each case, i.e.,

$$E_{nonzero}(\Delta\bar{d}) = \sum_{k=1}^{3} p_k E(\Delta\bar{d}_{\text{case } k}) \qquad (6.1)$$

where $E(\Delta\bar{d}_{\text{case } k})$ is the expected average link distance change for case k. $E(\Delta\bar{d}_{\text{case } k})$ is equal to the summation of the link distance change of each sub-case times the expected value of the corresponding probability.

For any of the above combinations, the calculation of a probability involving C(a, b) is impossible if a < b. This may occur when d_{ij} is either too large or too small. In such a case, the probability is simply 0. The largest b among all the combinations is 2. Since C(a, 2) = a(a−1)/2 which has an order of 2 for a, Case 2 above has an order of 2 for d_{ij}. Similarly, Case 3 has an order of 4 for d_{ij}. Because the mutation operator rate is usually very small, the effect of mutation operator in Case 2 (whose probability is $2r(1-r)$) is an order of magnitude more significant than the effect of mutation operator in Case 3 (whose probability is r^2). The overall calculation of the probabilities thus is a quadratic function of d_{ij}. Since d_{ij} varies for different link pairs, we can assume $d_{ij} = \bar{d}$ to approximate the probabilities involved in the calculation of $E(\Delta\bar{d}_{\text{case } k})$. The error of the approximated probabilities comes mainly from substituting $E(d_{ij}^2)$ by $\bar{d}^2$, which is propotional to the variance of the link distances.

6.2 Link Pairs with Zero Link Distance

For link pairs with zero link distance, we again need to look at the link pairs of the two links. Here, $d_{ij} = d(\tilde{s}_i, \tilde{s}_j) = 0$, which implies $\tilde{s}_i = \tilde{s}_j$. Let d_{ij}' denote the link distance after the operator is applied and l denote the link length. The effect of any operator on the link distance can be summarized again in three cases:

Case 1: Non of the links is affected by the operator.

Case 2: One of the links is affected by the operator.

Case 3: Both of the links are affected by the operator.

The probabilities of the three cases are:

Case 1: $(1-r)^2$

Case 2: $2r(1-r)$

Case 3: r^2

The effect of mutation operator on the link distance for each of the three cases is:

Case 1: No effect. $d_{ij}' = d_{ij} = 0$.

Case 2: $d_{ij}' = d_{ij} + 1 = 1$.

Case 3: There are three sub-cases.

Sub-case 1: both of the loci are the same. $d'_{ij} = d_{ij} = 0$. The probability is $1/\mathrm{C}(l, 2)$.

Sub-case 2: only one of the loci is the same. $d'_{ij} = d_{ij} + 2 = 2$. The probability is $2l\mathrm{C}(l-1, 2)/[\mathrm{C}(l, 2)\cdot\mathrm{C}(l, 2)]$

Sub–case 3: both of the loci are different. $d'_{ij} = d_{ij} + 3 = 3$. The probability is C(l–2, 2)/C(l, 2)

The expected change of average link distance for the link pairs with zero link distance can be found in a way similar to equation (6.1). Let $E_{zero}(\Delta \bar{d})$ denote the expected average link distance change for zero link pairs. Then

$$E_{zero}(\Delta\bar{d}) = (1-r^2)\cdot 0 + 2r(1-r)\cdot 1$$
$$+r^2[0\cdot 1/C(l,2) + 2\cdot 2lC(l-1,2)/[C(l,2)C(l,2)] + 3C(l-2,2)/C(l,2)]$$

or

$$E_{zero}(\Delta\bar{d}) = 2r(1-r) + r^2\left[\frac{(3l-1)(l-2)}{l(l-1)}\right] \tag{6.2}$$

The percentage of the link pairs with zero link distance will be reduced since the link distance in Case 2 and in Sub-cases 2 and 3 of Case 3 above is increased from zero to non-zero. The probability of the change, denoted as $p(d'_{ij}{\neq}0 \mid d_{ij}{=}0)$, is

$$p(d'_{ij}{\neq}0 \mid d_{ij}{=}0) = 2r(1-r) + r^2[2lC(l-1, 2)/[C(l, 2)\bullet C(l, 2)] + C(l-2, 2)/C(l, 2)]$$

or

$$p(d'_{ij}{\neq}0 \mid d_{ij}{=}0) = 2r(1-r) + r^2(l+1)(l-2) / [l(l-1)]$$

The probability for link distance being zero, $p(d_m = 0)$, during the m-th iteration will be reduced by a factor of $p(d'_{ij}{\neq}0 \mid d_{ij}{=}0)$ after the mutation operation. Let Φ denote the change of the probability, we have

$$\Phi = p(d_m = 0)[2r(1-r) + r^2 \frac{(l+1)(l-2)}{l(l-1)}] \tag{6.3}$$

From the same concept of equation (5.9), we can estimate the total expected average link distance change, $E_M(\Delta \bar{d}_m)$, due to the mutation operator at the m-th iteration as follows:

$$E_M(\Delta \bar{d}_m) = p(d_m = 0)E_{zero}(\Delta \bar{d}) + p(d_m \neq 0)E_{nonzero}(\Delta \bar{d}) \tag{6.4}$$

where $E_{zero}(\Delta \bar{d})$ is the expected average link distance change due to those link pairs with zero link distance, and the subscript M in $E_M(\Delta \bar{d}_m)$ denotes the effect of mutation.

6.3 The Mutation Operator Rate for Maximum Change of Link Distance

From equation (6.4), we are able to derive the highest mutation operator rate for the largest expected link distance change during the iterations. Equation (6.4) can be expanded as

$$E_M(\Delta \bar{d}_m) = p(d_m = 0)\sum_{k=1}^{3} p_k E_{zero}(\Delta \bar{d}_{\text{case } k}) + [1 - p(d_m = 0)]\sum_{k=1}^{3} p_k E_{nonzero}(\Delta \bar{d}_{\text{case } k}) \tag{6.5}$$

Reorganizing (6.5), we have

$$E_M(\Delta \bar{d}_m) = p(d_m = 0)\left[(1-r)^2 \cdot 0 + 2r(1-r)E_{zero}\left(\Delta \bar{d}_{\text{case } 2}\right) + r^2 E_{zero}\left(\Delta \bar{d}_{\text{case } 3}\right)\right]$$

$$+\left[1 - p(d_m = 0)\right]\left[(1-r)^2 \cdot 0 + 2r(1-r)E_{nonzero}\left(\Delta \bar{d}_{\text{case } 2}\right) + r^2 E_{nonzero}\left(\Delta \bar{d}_{\text{case } 3}\right)\right] \tag{6.6}$$

so

$$E_M(\Delta\bar{d}_m) = 2r(1-r)E_1 + r^2E_2 \tag{6.7}$$

where

$$E_1 = p(d_m = 0)E_{zero}\left(\Delta\bar{d}_{\text{case 2}}\right) + \left[1 - p(d_m = 0)\right]E_{nonzero}\left(\Delta\bar{d}_{\text{case 2}}\right) \tag{6.8}$$

$$E_2 = p(d_m = 0)E_{zero}\left(\Delta\bar{d}_{\text{case 3}}\right) + \left[1 - p(d_m = 0)\right]E_{nonzero}\left(\Delta\bar{d}_{\text{case 3}}\right) \tag{6.9}$$

To have the maximum expected link distance change, we need only to find the partial derivative of equation (6.7) with respected to r and set it to zero. Namely, we have

$$\frac{\partial E_M(\Delta\bar{d}_m)}{\partial r} = (2-4r)E_1 + 2rE_2 = 0 \tag{6.10}$$

From equation (6.10), the mutation operator rate to achieve the maximum expected average link distance change is

$$r = \frac{E_1}{2E_1 - E_2} \tag{6.11}$$

The operator rate is limited to the range (0, 1). Equation (6.11) provides a possible maximum r value within the range. It needs to be pointed out that E_1 and E_2 from equation (6.8) and (6.9) change their values with the number of iterations. This suggests that the maximum mutation operator rate varies during the iteration, instead of having a fixed value for the whole process. The above technique of analysis for the mutation operator should be applicable to many of other popular operators such as the inversion operator used in genetic algorithms.

CHAPTER 7

CHANGE OF EXPECTED AVERAGE LINK DISTANCE DUE TO BOTH SELECTION AND MUTATION

In Chapter 5 and 6, the change of the average link distance due to selection and mutation are discussed separately. In this chapter, we will combine their effects on the average link distance to compute the expected change of average link distance.

Equation (5.7) can be rewritten as

$$1-p(d_{m+1}=0)=\left(\frac{n-1}{n}\right)[1-p(d_m=0)] \tag{7.1}$$

Equation (7.1) indicates that the percentage of non-zero link pairs are reduced by a factor of $1/n$ in each iteration due to random selection. Let $p(d_m \neq 0)$ denote the probability of link distance greater than zero. Equation (7.1) can be changed to

$$p(d_{m+1}\neq 0)=\left(\frac{n-1}{n}\right)p(d_m\neq 0) \tag{7.2}$$

Since those link pairs with zero link distance do not contribute to the average link distance, we can rewrite the relationship between the expected average link distance before and after selection as:

$$E_S(\overline{D}_{m+1})=p(d_m=0)\cdot 0+p(d_{m+1}\neq 0)\cdot\overline{D}=\left(\frac{n-1}{n}\right)p(d_m\neq 0)\cdot\overline{D}$$

$$=\left(\frac{n-1}{n}\right)E_S(\overline{D}_m) \tag{7.3}$$

where $E_S(\overline{D}_{m+1})$ is the expected average link distance due to random selection after m iterations and $\overline{D}$ is the average link distance of the initial mating pool. Combining equations (6.2) and (7.3), we have

$$E(\overline{D}_{m+1})=\left(\frac{n-1}{n}\right)E_S(\overline{D}_m)+E_M(\Delta\overline{d}_m) \tag{7.4}$$

where $E(\overline{D}_{m+1})$ is the expected average link distance of the mating pool after m iterations and $E_S(\overline{D}_0)=\overline{D}$, the average link distance of the initial mating pool. It should be noticed that the probability of the link distance being zero can no longer be simply calculated by equation (5.8). Instead, it should be adjusted as indicated by equation (6.3) in each iteration.

Equation (7.4) can be further rewritten as

$$E(\overline{D}_m)=\left[\left(1-\frac{1}{n}\right)\right]^m\overline{D}+\sum_{j=1}^{m}\left\{\left[\left(1-\frac{1}{n}\right)\right]^{m-j}E_M(\Delta\overline{d}_j)\right\} \tag{7.5}$$

From equation (5.9), $\overline{D}$ is a function of l. We again rewrite (7.5) as

$$E\big(\overline{D}(l,m,n)\big)=\left[\left(1-\frac{1}{n}\right)\right]^m\overline{D}(l)+\sum_{j=1}^{m}\left\{\left[\left(1-\frac{1}{n}\right)\right]^{m-j}E_M(\Delta\overline{d}_j)\right\} \tag{7.6}$$

From equation (7.6) we can find not only how the mating pool converges with respect to the number of iterations but also how many iterations are needed to reach an expected convergence stage measured by the average link distance. Based on equation (7.5) Figure 7.1 plots the expected average link distance versus iteration number for several mutation operator rates. The link length in Figure 7.1 is 20, the population size is 20, and the mutation operator rates are 0, 0.01, 0.02, 0.03, 0.04 and 0.05. A larger mutation operator rate bears a slower reduction in the expected average link distance. The dashed line in Figure 7.1 is the case that the mutation operator rate equals to zero, which is exactly the same as the simple genetic algorithm with only selection discussed before. Figure 7.1 provides an evidence that the mutation operator increases the average link distances during the iteration and thus slows down the converge process.

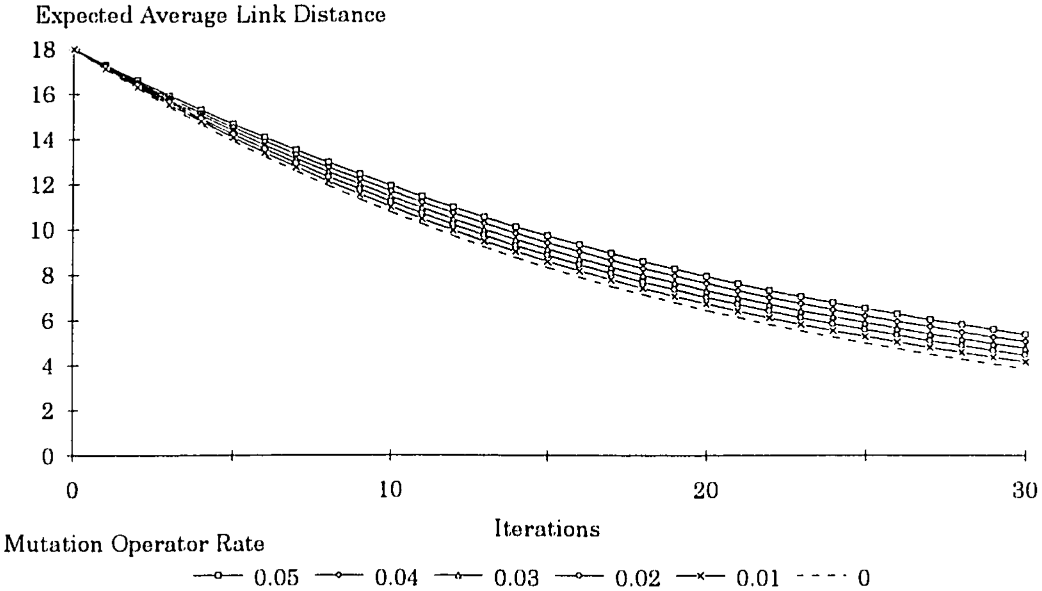

Figure 7.1 Expected Average Link Distance for Different Mutation Operator Rates. Link Length = 20, Population Size = 20.

Equation (7.5) contains the joint effects of selection and mutation. Selection is expected to always drag the average link distance down by a ratio of $1/n$ in each iteration. So, the joint effects from both selection and mutation after a long run will be that the selection is reducing the expected average link distance while the mutation operator is increasing it. If we fix the mutation operator rate during the iterations, once the joint effect of selection and mutation operator reaches a balance, the expected average link distance will converge to a positive number according to equation (7.6).

Let D_c denote the converged average link distance. We have D_c = $E(\overline{D}_{m+1}) = E(\overline{D}_m)$. Thus D_c can be easily found to be

$$D_c = nE_M(\Delta \bar{d}_m) = n[2r(1-r)E_1 + r^2 E_2] \tag{7.7}$$

Equation (7.7) shows the relationship between D_c and r. Since the mutation rate r is usually very small (say, a few percent at most), D_c increases if r is larger.

From our analysis, it is expected that the average link distance is continuously decreasing but will never reach zero. In the previous Markov chain analysis, we know that the least average link distance is zero in an absorbing state where the mating pool contains only identical links. The second least average link distance is $2/n$ which can be found from a state where one link is different from all the others and the link distance between this link and any other link is 1. There is no number between 0 and $2/n$ for the average link distance. If we select a mutation operator rate so small that D_c is smaller than $2/n$, the iteration should stop when the average link distance has reached $2/n$. We can therefore setup a criterion to estimate the number of iterations needed to reach an expected state of convergence.

If the mutation operator rate is selected such that

$$D_c > \frac{2}{n}$$

the smallest number of iterations, m, such that

$$E(\overline{D}(l,m,n)) = \left[\left(1-\frac{1}{n}\right)\right]^m \overline{D}(l) + \sum_{j=1}^{m}\left\{\left[\left(1-\frac{1}{n}\right)\right]^{m-j} E_M(\Delta\overline{d}_j)\right\} = D_c \qquad (7.8)$$

will bring the mating pool to a convergence. Otherwise, the smallest number of iterations, m, such that

$$E\left(\overline{D}\left(l,m,n\right)\right) = \left[\left(1-\frac{1}{n}\right)\right]^m \overline{D}(l) + \sum_{j=1}^{m}\left\{\left[\left(1-\frac{1}{n}\right)\right]^{m-j} E_M(\Delta\overline{d}_j)\right\} < \frac{2}{n} \qquad (7.9)$$

will bring the mating pool to a convergence.

The smaller number of m that satisfies equation (7.8) and m that satisfies equation (7.9) is the number of iterations needed for the expected convergence of the order-based genetic algorithm with equal selection probabilities for different links. Since the genetic drift suggests the slowest convergence, the m obtained from equation (7.8) and equation (7.9) is the upper bound for the number of iterations needed for the convergence of order-based genetic algorithms with any selection probabilities.

Table 7.1 shows some of the results calculated from equation (7.6). The meanings of the symbols are: l is the link length, n is the population size, r is the mutation operator rate, D_c is the converged average link distance, m is number of iterations needed for the average link distance to achieve D_c, $2/n$ is the other criteria to stop the iterative process, m^* is the number of iterations needed for the average link distance to arrive at $2/n$. From Table

7.1, the expected average link distance will reach $2/n$ when the population size or the mutation operator rate is small. The largest number of iterations needed to achieve a convergence is less than 1700 for the cases shown. The larger the link length or the population size, the slower the convergence.

Table 7.1 Some Numerical Results from Equation (7.6)

l	n	r (%)	D_c	m	$2/n$	m^*
10	10	1	0.20058	132	0.2	–
10	10	0.1	0.02001	130	0.2	36
10	10	0.01	0.002	136	0.2	36
50	50	1	1.45632	783	0.04	–
50	50	0.1	0.10005	768	0.04	–
50	50	0.01	0.01	801	0.04	366
100	100	1	3.22346	1542	0.02	–
100	100	0.1	0.2001	1619	0.02	–
100	100	0.01	0.02	1693	0.02	1693
10	100	0.1	0.20006	1388	0.02	–
100	10	0.1	0.10005	816	0.2	342
10	50	0.1	0.10003	698	0.04	–
50	10	0.1	0.02001	152	0.2	54
50	100	0.1	0.20009	1660	0.02	–
100	50	0.1	0.10005	816	0.04	–

Discussed in this chapter is the convergence of the order-based genetic algorithm with selection and mutation. In each iteration of the genetic algorithm, random selection and mutation operation are sequentially applied. For real genetic algorithm applications more operators, especially the crossover operator, are usually included. The effects of these operators are also sequential. Analysis of these applications can be done by adding the effect of each operator after equation (7.4) for the joint effect.

CHAPTER 8

CONCLUSIONS

Studied in this dissertation are methods and results of performance analysis for genetic algorithms. Both statistical analysis for comparing variations in genetic algorithms and probability analysis to investigate the expected convergence behavior of a genetic algorithm are performed.

A Wilcoxon signed rank test is used to study the effect of adapting the operator production ratios in the genetic algorithm. The adaptation of the operator production ratio during the iterative process is shown to be effective for achieving a faster convergence for the tested traveling salesperson problems. It provides a way of examining whether the modification is good or not when initiating a new genetic algorithm.

We analyze the genetic drift and preferential selection of the genetic algorithm using Markov chains. The probabilities of both phenomena are derived. It is shown that the genetic drift has a slower convergence than any preferential selections. The probability of pre-mature convergence due to the use of high selection probabilities for dominant links is shown to be high.

A new method of analysis is introduced which uses the "link distance" as a reference for studying the convergence of order-based genetic algorithms. The average link distance of a randomly generated mating pool is derived and shown to be a function of only link length. The value of this distance is shown based on numerical analysis to be the link length minus 2.

The expected average link distance changes for random selection, mutation operator, and the combination of both are derived. A mutation operator rate for the maximum expected average link distance change is also derived. The derived mathematical model for the expected average link distance during the iterations shows that this distance converges to a positive number which is a function of the population size and mutation operator rate. The expected number of iterations needed to converge has been obtained for some typical values of link length, population size, and mutaiton operator rate. We plan to study the effects of other operators in the future.

APPENDIX A

CONDITIONAL PROBABILITY OF AN OUTCOME FROM MUTATION OPERATION

From the rule of conditional probability, we can find that

$\text{Prob}\{B = E \text{ and } A = G \mid D = H, A \neq E, B \neq G, C \neq F\} =$

$\text{Prob}\{A = G \mid B = E, D = H, A \neq E, B \neq G, C \neq F\} \cdot$

$\text{Prob}\{B = E \mid D = H, A \neq E, B \neq G, C \neq F\}$.

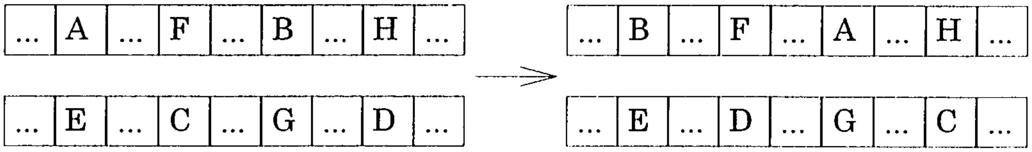
... A ... F ... B ... H ...
... E ... C ... G ... D ...
... B ... F ... A ... H ...
... E ... D ... G ... C ...

Figure A1 Relationship of the links of a link pair before and after the mutation operation.

$\text{Prob}\{D = H, A \neq E, B \neq G, C \neq F\}$ can be found from selecting two objects (A and B) such that both are from the group of different gene pairs and then selecting other two objects (C and D) such that one is from the group of different gene pairs and the other is from the group of identical gene pairs.

To find $\text{Prob}\{B = E \mid D = H, A \neq E, B \neq G, C \neq F\}$ we need to find all possible outcomes. Since E can not be A, H can not be C, and C can not be H or F, the possible outcomes are:

1. $C = B$: The probability of $C = B$ is $1/(l-2)$. Since $B = C$, B can not be E. The probability in this case is 0 for $B = E$.
2. $C = A$: The probability of $C = A$ is $1/(l-2)$. The probability for $B = E$ given $C = A$ is $1/(l-3)$ since B can not be either C, D, or G.

3. C equals to any others : Since C can not be any of A, B, F, or H, C must be equal to another gene, let's denote it as K. The probability is $1 - 2/(l-2)$ for C = K. The probability for B = E given C = K is $1/(l-3)$ since B can not be either C, D, or G.

So, Prob{B = E | D = H, A ≠ E, B ≠ G, C ≠ F} $= 0 \cdot [1/(l-2)] + [1/(l-3)] \cdot [1/(l-2)] + [(l-4)/(l-2)] \cdot [1/(l-3)] = [1/(l-3)] \cdot [1/(l-2)+(l-4)/(l-2)]=1/(l-2)$

To find Prob{A = G | B = E, D = H, A ≠ E, B ≠ G, C ≠ F} we also need to find all possible outcomes. Since A can not be E, D can not be C and C can not be B, F or H, the possible outcomes are:

1. C = A : The probability of this case is $1/(l-3)$. The probability is 0 for A = G since A = C.
2. C ≠ A : The probability of this case is $1 - 1/(l-3)$. The probability for A = G is $1/(l-3)$ since A can not be either C, D, or E.

Prob{A = G | B = E, D = H, A ≠ E, B ≠ G, C ≠ F} $= \{0 \cdot 1/(l-3) + [(l-4)/(l-3)] \cdot 1/(l-3)\} \cdot \{C(d_{ij}+1,\ 2) \cdot C(l-d_{ij}-1,\ 0) \cdot C(d_{ij}-1,\ 1) \cdot C(l-d_{ij}-1,\ 1)\ /\ [C(l,\ 2) \cdot C(l-2,\ 2)]\} = \{C(d_{ij}+1,\ 2) \cdot C(l-d_{ij}-1,\ 0) \cdot C(d_{ij}-1,\ 1) \cdot C(l-d_{ij}-1,\ 1)\ /\ [C(l,\ 2) \cdot C(l-2,\ 2)]\} \cdot (l-4)/(l-3)^2$.

So, Prob{B = E and A = G | D = H, A ≠ E, B ≠ G, C ≠ F} $= \{C(d_{ij}+1,\ 2) \cdot C(l-d_{ij}-1,\ 0) \cdot C(d_{ij}-1,\ 1) \cdot C(l-d_{ij}-1,\ 1)\ /\ [C(l,\ 2) \cdot C(l-2,\ 2)]\} \cdot \{1/(l-2) \cdot (l-4)/(l-3)^2\} = \{C(d_{ij}+1,\ 2) \cdot C(l-d_{ij}-1,\ 0) \cdot C(d_{ij}-1,\ 1) \cdot C(l-d_{ij}-1,\ 1)\ /\ [C(l,\ 2) \cdot C(l-2,\ 2)]\} \cdot (l-4)/[(l-2)(l-3)^2]$

REFERENCES

David, L., 1989, "Adapting Operator Probabilities in Genetic Algorithm," *Proceeding of the Third International Conference on Genetic Algorithms and Their Applications*, George Mason University, Virginia, pp. 61-69.

Davis, L., 1985, "Job Shop Scheduling with Genetic Algorithms," *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Carnegie-Mellon University, Pennsylvania, pp. 136-140.

Edward R. D., 1990, *Probability and Statistics Engineering, Computing, and Physical Sciences*, Prentics-Hall Inc., Englewood Cliffs, New Jersey.

Englander, A. C., 1985, "Machine Learning of Visual Recognition Using Genetic Algorithms," *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Carnegie-Mellon University, Pennsylvania, pp. 197-201.

Goldberg, D. E. and Segrest, P, 1987, "Finite Markov Chain Analysis of Simple Genetic Algorithm," *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, Massachusetts Institute of Technology, Massachusetts, pp. 1-8.

Golden, B. L., W. R. Stewart, 1985, "Empirical Analysis of Heuristics," *The Traveling Salesman Problem*, Chapter 7, John Wiley & Sons Ltd., New York, New York, p.p. 207-215

Grefenstette, J. J., R. Gopal, B. Rosmaita, and D. Van Gucht, 1985, "Genetic Algorithm for the Traveling Salesman Problem," *Proceeding of the First International Conference on Genetic Algorithms and Their Applications*, Carnegie-Mellon University, Pennsylvania, pp. 160-168.

Grefenstette, J. J., 1986, "Optimization of Control Parameters for Genetic Algorithm," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-16, No. 1, pp. 122-128.

Holland, J. H., 1975, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, Michigan.

Kemeny, J. G. and Snell, J. L., 1960, *Finite Markov Chains*, Van Nostrand, Princeton, New Jersey.

Koza, J. R., 1990, "Non-linear Genetic Algorithms for Solving Problems," United States Patent No. 4,935,877.

Lawer, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, 1985, *The Traveling Salesman Problem*, John Wiley & Sons Ltd, New York, New York.

Leu, Ming C., Hermean Wong, Zhiming Ji, Dec. 1993, "Planning of Component Placement/Insertion Sequence and Feeder Setup in PCB Assembly Using Genetic Algorithm," *Journal of Electronic Packaging*, Vol. 115, No. 4, pp. 424-432.

Mosteller, F., R. Rourke, 1973, *Sturdy Statistics*, Addision-Wesley, Reading, Massachusetts.

Nix, A. E. and Vose, M. D., 1992, "Modeling Genetic Algorithms with Markov Chains," *Annals of Mathematics and Artificial Intelligence*, Vol. 5, No. 1.

Oliver, I. M., D. J. Smith, and J. Holland, 1987, "A Study of Permutation Crossover Operations on the Traveling Salesman Problem," *Proceeding of the Second International Conference on Genetic Algorithms and Their Applications*, Massachusetts Institute of Technology, Massachusetts, pp. 224-230.

Pfaffenberger, R., J. Patterson, 1981, *Statistical Methods*, Irwin, Homewood, Illinois.

Shahookar, K. and P. Mazumder, 1990, "A Generic Approach to Standard Cell Placement Using Meta-genetic parameter optimation," *IEEE Transactions on Computer-Aided Design*, Vol 9 (5), pp. 500-511.

Suzuki, Joe, 1993, "A Markov Chain Analysis on A Genetic Algorithm," *Proceeding of the Fifth International Conference on Genetic Algorithms and Their Applications*, University of Illinois at Urbana-Champaign, Illinois, pp. 146-153.

Wong. Hermean, 1991, *Genetic Algorithm for Solving Printed Circuit Board Assembly Planning Problems*, Master Thesis, New Jersey Institute of Technology, Newark, New Jersey.

Wong, Hermean, Ming C. Leu, 1993, "Adaptive Genetic Algorithm for Optimal Printed Circuit Board Assembly Planning," *Annals of the CIRP*, Vol. 42, pp. 17 - 20.