This is to certify that the
dissertation entitled

Modeling Genetic Algorithm Dynamics for OneMax and Deceptive Functions

presented by

Bulent Buyukbozkirli

has been accepted towards fulfillment
of the requirements for the

Doctoral degree in Mathematics

Major Professor's Signature

August 26, 2004

Date

DATE DUE	DATE DUE	DATE DUE

MODELING GENETIC ALGORITHM DYNAMICS FOR ONEMAX AND DECEPTIVE FUNCTIONS

By

Bulent Buyukbozkirli

A DISSERTATION

Submitted to
Michigan State University
In partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Mathematics

2004

ABSTRACT

MODELING GENETIC ALGORITHM DYNAMICS FOR ONEMAX AND DECEPTIVE FUNCTIONS

By

Bulent Buyukbozkirli

In this dissertation, we develop a model predicting dynamics of the counting-ones (OneMax) and a form of deceptive function problems. The model describes the mean allele and, in the case of deceptive function, mean deceptive block values. The genetic algorithm (GA) that is being modeled consists of two-point crossover, fitness proportional selection and mutation operators. The model is developed to estimate the average GA dynamics, but it can also be used for an individual run of the GA.

First, we develop the model for the OneMax problem with very high crossover rates. Then, we modify the model by using statistics of very early generations from GA runs, to describe the complete dynamics for different (lower) crossover rates of the OneMax problem. In the development of the model, we introduce a new quantity that measures the effect of the crossover operation and is independent of generation, for practical purposes. Then, the model is generalized to cover other cases of the OneMax, such as weighted OneMax, as well as a form of deceptive function problem, for high enough crossover rates. The model is also modified to include Boltzmann selection with fixed or scaled selection pressures.

even

Boltz

Since

dece

a sui

algori

gener

simpl

nontr

to m

degr

diffe

the

it sh

equa

equ

The model can be applied to OneMax and deceptive function problems even when the crossover, mutation and the selection pressures (in the case of Boltzmann scaling) are changed at predetermined generations during a GA run. Since our model estimates the mean value (and, mean deceptive block value for deceptive function) at each locus at any generation, it can be used to determine a suitable migration time as well as the migration rate for parallel genetic algorithms (in the island model case) when migrations are allowed at any generation for our benchmark problems.

Although the problems for which our model proved successful were rather simple or idealized, they were often sufficiently involved to capture interesting nontrivial features of the GA dynamics. The author hopes to extend the approach to model solution of more representative real-world problems with various degrees of OneMax similarity and various amounts of deception.

At the end of the dissertation, an attempt to develop a stochastic differential equation model to predict the evolution of the fitness distribution for the OneMax problem is also presented. Although our attempt was not successful, it shows a strong connection between fitness evolution and certain diffusion equations and points toward the possibility of the existence of a diffusion-type equation that could describe the OneMax and maybe other type of GA dynamics.

The f

came

Dr. G

gene

this s

very

algori

the di

like to

McCl

resea

for re

My w

mom

befo

work

the g

ACKNOWLEDGEMENTS

The front page of this dissertation has the name of only one person; however it came into existence by the collective being of many others.

Dr. Goodman has introduced me to the interesting, open questions in the field of genetic algorithms. Through long and fruitful discussions with him, the details of this study have emerged. He was also very patient to read the manuscript in its very raw form and suggest many improvements. My involvement in the genetic algorithm theory has started by the suggestions and guidance of Dr. MacCluer in the direction of applying it to solve a practical power electronics problem. I would like to also express my thanks to the other members of my thesis committee: Dr. McCleer and Dr. Peng for their suggestions about the practical application of my research in the field of electrical power inverters, and Dr. Newhouse and Dr. Hall for reading my thesis and for their insightful questions.

My wife, Figen, has been the strongest source of inspiration and love at every moment of my studies. Our son, Eren Baray, who was born just a couple months before the completion of this dissertation, brought the energy and will to bring this work into its final form at its hardest stages. I am also grateful for the prayers and the good wishes of my parents, elders of my family and all the friends.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figur

Figur

Figur

Figur

Figu

Figu

Figu

Figu

Figu

Figu

Figu

Figu

Figu

Figur

Figur

Figur

Figur

Figur

Figur

Figur

Figur

Figur

Figur

Chapter 1

GENETIC ALGORITHMS: THEORY AND MODELS

1.1 Thesis Outline

Genetic algorithms (GA) are stochastic, heuristic search algorithms that have been applied to a wide range of problems. In Section 1.2, the simple genetic algorithm is described together with some of the ways its dynamics are observed. The counting-ones (OneMax) benchmark problem is also introduced in this section serving as a simple demonstrative example for GA operators. We present a real-world example in Section 1.3, in order to demonstrate a nontrivial problem in which GA can be applied to find an optimum configuration. Section 1.4 contains a review of some of the ways GAs are currently analyzed. Chapter 1 ends with a critique of the current theories in Section 1.5, and explains why we need a new model, which is the motivation in the development of this dissertation.

Chapter 2 starts with a detailed description of the OneMax problem. A new and practical model for the simple Genetic Algorithm dynamics of OneMax problem is developed in Section 2.2. The GA that is being modeled consists of two-point crossover, fitness proportional selection and mutation operators. For

low crossover rates, the model uses statistics of the early generations of GA runs to describe the dynamics of the problem for all time, using a variety of crossover and mutation rates. In the development of the model, we introduce a new quantity that measures the effect of the crossover operation and is independent of generation, for practical purposes. Then, the model is generalized, in Section 2.3 and 2.4, to cover the weighted OneMax and the Boltzmann selection with fixed or scaled selection pressures.

Chapter 3 follows a similar pattern as in Chapter 2, but for a type of deceptive function in stead of OneMax. The model in this chapter is developed only for high enough crossover rates. It estimates the simultaneous evolution of the mean allele and the mean all-1 deceptive blocks, as well as the mean fitness and fitness variance of the population. The model for lower crossover rates has still been under investigation by the author at the time this dissertation was written. In Section 3.1 the deceptive function is described and the modeling problem is formulated. The model for fitness proportional selection is developed in Section 3.2, which is followed by the extensions of the model to weighted deceptive function and Boltzmann selection in Sections 3.3 and 3.4, respectively.

In Chapter 4, the model that is developed in Chapter 2 and Chapter 3 is tested by comparing its computer simulations with the results with averaged GA runs. Section 4.1 presents the results for OneMax and Section 4.2 for the deceptive function.

An attempt to develop a stochastic differential equation model to predict evolution of fitness distribution for the OneMax problem is presented in Appendix. Although our attempt was not successful, it demonstrates a strong connection between fitness evolution and certain diffusion equations and points toward the possibility of the existence of a diffusion-type equation that could describe the OneMax and maybe other type of GA dynamics.

1.2 What is a Genetic Algorithm?

Genetic Algorithms (GAs) are stochastic search algorithms based on principles of natural selection and genetics. They were first developed by J. Holland in the 1960's, and presented, together with a large body of accompanying theory, in his 1975 book. There are three main operations in a genetic algorithm, namely selection, mutation and crossover (sometimes called recombination). Each operation has many different types which may be applied, depending on the specific nature of the optimization problem. At the beginning, an initial population that consists of a set of proposed solutions is chosen. The selection of the initial population is usually random, unless it is specifically required to search a more restricted region. Then, selection, mutation and crossover operations are applied to this initial population in a predetermined order to create the first generation (or second population), Figure 1.1. The second generation is obtained from the first one by the application of the same operators. This procedure continues until a "good enough" solution emerges within the population or until the whole population converges to the point that additional search is deemed unproductive.

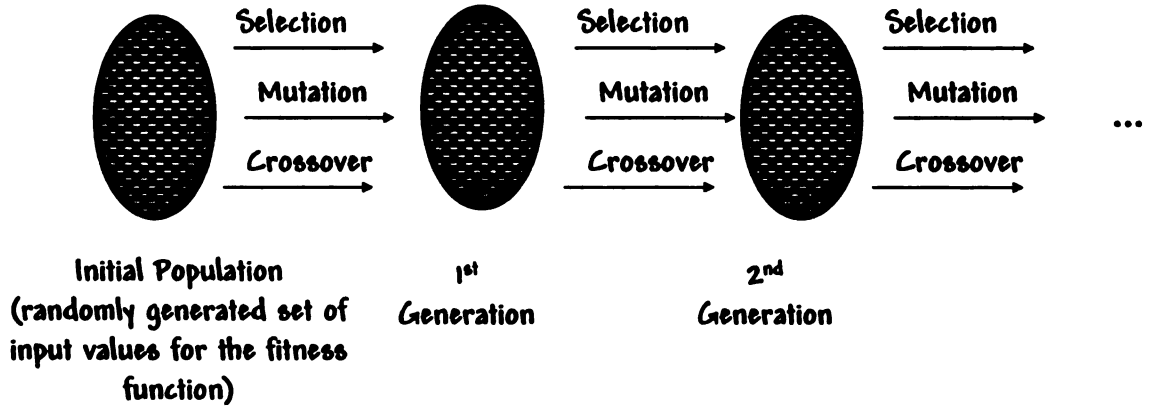


Figure 1.1 Genetic Algorithm creating new populations with selection, mutation and crossover operations

Let us consider a very simple example in order to illustrate the GA operations and define some GA-specific terms. In this example, we want to find the value of the input variables that yields the maximum value of the objective function f (called the 'fitness function' in the GA terminology)

$$f(a_1,a_2,a_3,a_4,a_5) = a_1 + a_2 + a_3 + a_4 + a_5, \tag{1.1}$$

where the input variables, a_1,a_2,a_3,a_4 and a_5, take discrete values of only 0 or 1. A problem of this form, of arbitrary dimension, is called a 'counting-ones' (or sometimes 'OneMax') problem, since the value of the function is equal to the number of 1's in the input values. It is obvious that the solution to the problem is $a_1 = 1, a_2 = 1, a_3 = 1, a_4 = 1, a_5 = 1$. It is not so obvious, but important, that even some real-world problems – those that can be solved by decomposition into

independent components – exhibit many aspects of the behavior of a counting ones problem, after suitable recoding of the inputs. Because of their simplicity, counting ones problems are often used in analyses of the dynamics of a genetic algorithm. A few examples of this kind of application can be found in [Goldberg, 1989], [Furutani 2002], [Prügel-Bennett, 2002] and in many other publications.

Now, let us see how a GA would solve this problem of find the optimum of the function f. First, select values for $(a_1, a_2, a_3, a_4, a_5)$ randomly, for example (1,0,1,0,0), etc.. To make the initial population, we generate a fixed-size of such number strings each of which is called a 'chromosome'. In Figure 1.2, we see an example of such a population with 4 chromosomes, showing their fitness values and the ratios of their fitness to the total fitness of the population (so-called "relative fitness values"). In this example, the 'chromosome length' is 5 – i.e., 5 variables – and the 'population size' is 4.

#	Chromosome	Fitness	% of Total
1	1-0-1-0-0	2	25
2	1-0-0-0-1	2	25
3	0-0-1-1-1	3	37.5
4	0-0-0-1-0	1	12.5
Total		8	100.0

Figure 1.2 An example of an initial population together with the fitnesses and relative fitnesses of their chromosomes

Given a population, the selection operation creates a new population with the same number of chromosomes by selecting some of its chromosomes. One of the most common types of selection is 'fitness proportional' selection, also called 'roulette wheel' selection. In this type of selection, the probability that each chromosome would be selected for the next generation is equal to the proportion of its fitness to the total fitness of the population (i.e., its relative fitness). In our example, Figure 1.2, the first two chromosomes have probability 0.25 of being selected for the next generation, while the third one has probability 0.375 and the last one, 0.125. The selection operation is performed with replacement. That is, the fittest chromosomes might be selected more than once, while some chromosomes might be lost during the selection procedure. A possible outcome of the selection is shown in Figure 1.3, in which two copies of the third chromosome are selected, while the fourth one is not present after selection.

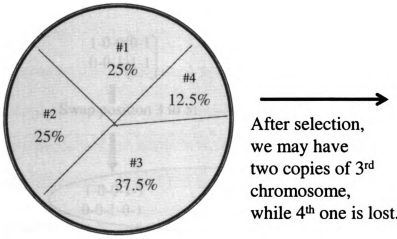


Figure 1.3 A possible outcome of fitness proportional selection

The crossover operation is the most complicated operation of the GA. One of the most common types of crossover is two-point crossover. In two-point crossover, the chromosomes are first paired randomly, using a uniform probability density distribution. Then, within each pair, the parts of the chromosomes lying between two randomly (with a uniform probability density distribution) selected locations are swapped. Not all pairs of chromosomes have to undergo the crossover operation, though. An important parameter of crossover operation, called the crossover probability, p_c, gives the probability at which a given pair of chromosomes are crossed. In Figure 1.4, first and third chromosomes pair off and their last two genes are swapped. Also, the second and fourth chromosomes pair off and their third and forth genes are swapped.

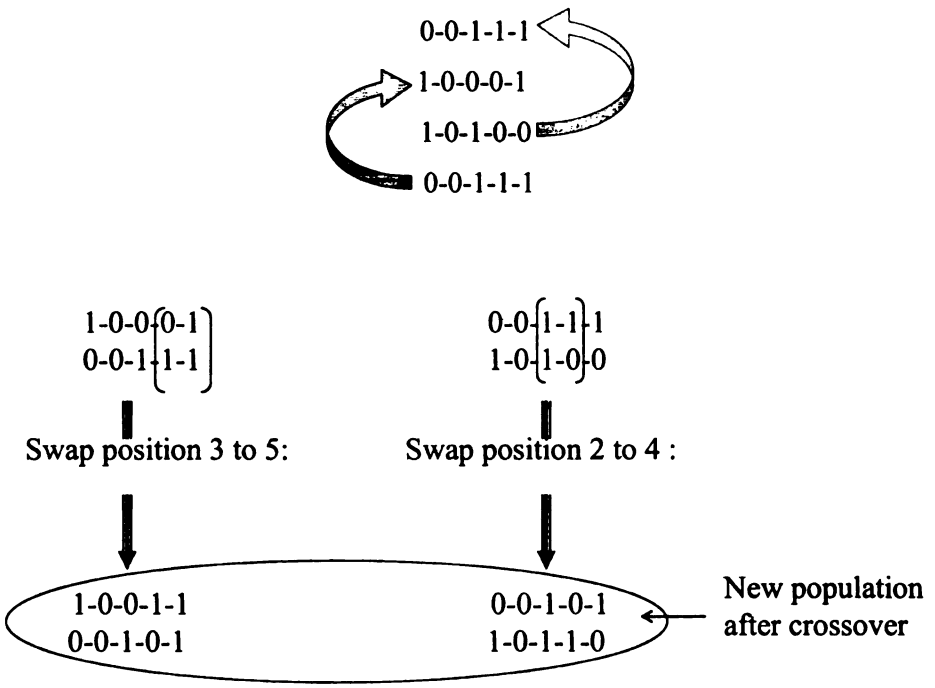


Figure 1.4 An example of two-point crossover

In bitwise mutation, each gene of each chromosome changes its value from 1 to 0 or 0 to 1 with mutation probability, p_m.For example, a mutation at the 4th gene of 0-0-1-0-0 gives us 0-0-1-1-0, Figure 1.5.

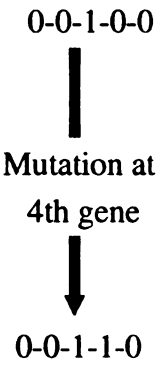


Figure 1.5 An example of mutation

Thus, after selection, crossover and mutation are applied, we get a new population, which is, on average, expected to be 'better' than the previous one.

Sometimes the relative fitness values of the chromosomes are replaced by some other scaled values of the fitness for the purposes of the selection operation. For example, if one wants to give disproportionately more chance of being selected for chromosomes with higher fitnesses, one can multiply their fitness values by some weights and use those values instead of their original fitness values. One of the common ways of fitness scaling is Boltzmann scaling, in which the fitness, f_c, of chromosome c is replaced by $\tilde{f}_c = e^{\beta f_c}$, where β is a constant, called the Boltzmann constant. The advantages of using a Boltzmann scaling is explained in Section 2.4.

Another common application of GA involves using multiple populations in parallel. This case is called parallel GAs. Each population (deme) evolves separately. However, at certain times some of the chromosomes, usually the best ones, of some populations are allowed to 'migrate' into other populations (typically being copied, without removal from the 'donor' population). Figure 1.6 shows a possible configuration for such an application. The main purpose of such an application is to reduce the risk of premature convergence.

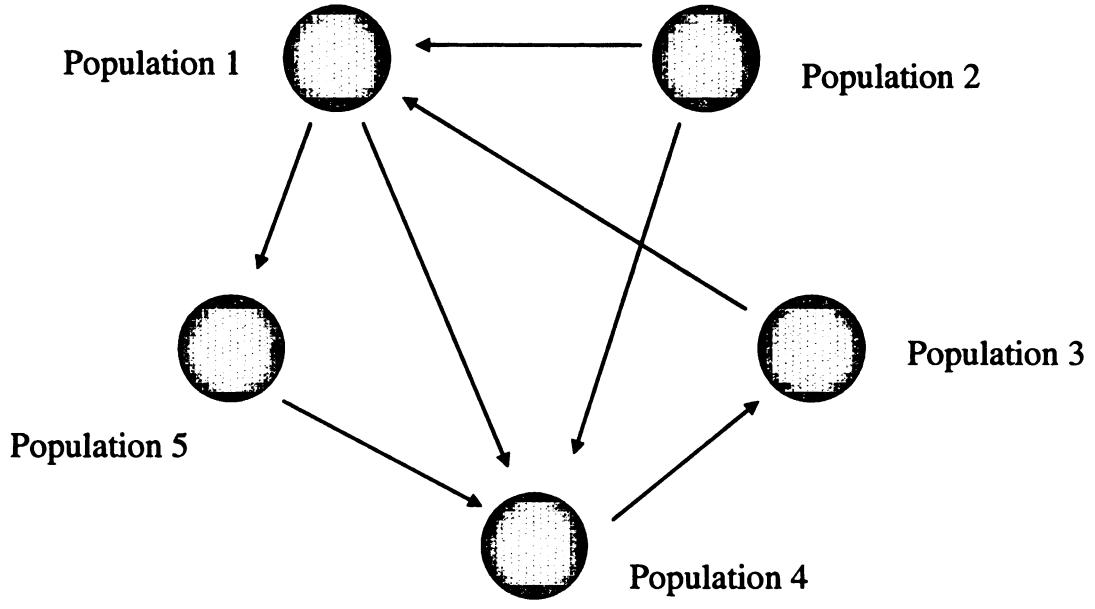


Figure 1.6 A GA with multiple populations, arrows showing the direction of migration

There are several advantages of GAs over other optimization and search procedures for some classes of search problems. First, GAs can climb many peaks in parallel. There is then a smaller probability that they will miss a peak that leads to a global optimum. Second, GAs use the value of a function rather

than performing operations on an explicit representation of the function (for example, formal differentiation). It is thus possible to find a good estimation of an optimum in a situation where the form of function itself is not known, but values of the function in particular situations are known, such as stock prices in the financial markets. GAs also produce effective solutions to problems with too many variables, in which other search procedures have hard time due to time limitations (computational complexity grows exponentially, for example, with problem order n). Since the information is exploited directly from a fitness function, no other information is necessary about the problem under consideration. In addition, GAs are better than a random search for almost any problem domains of significant interest, since they use directed randomness, which reduces the computation time by skipping areas that are not fruitful to search, directing the search effort towards areas where it is more likely to find the optimum.

1.2.1 Counting-Ones (OneMax)

In this section we study the counting-ones problem with a longer chromosome length. In this example, the chromosome length is 100. Thus the fitness function is

$$f(a_1, a_2, ..., a_{100}) = a_1 + a_2 + ... + a_{100} \ . \tag{1.2}$$

We apply the GA with population size $P = 50$, mutation rate $p_m = 0.01$, crossover rate $p_c = 1$, and Boltzmann scaling with Boltzmann constant $\beta = 0.2$.

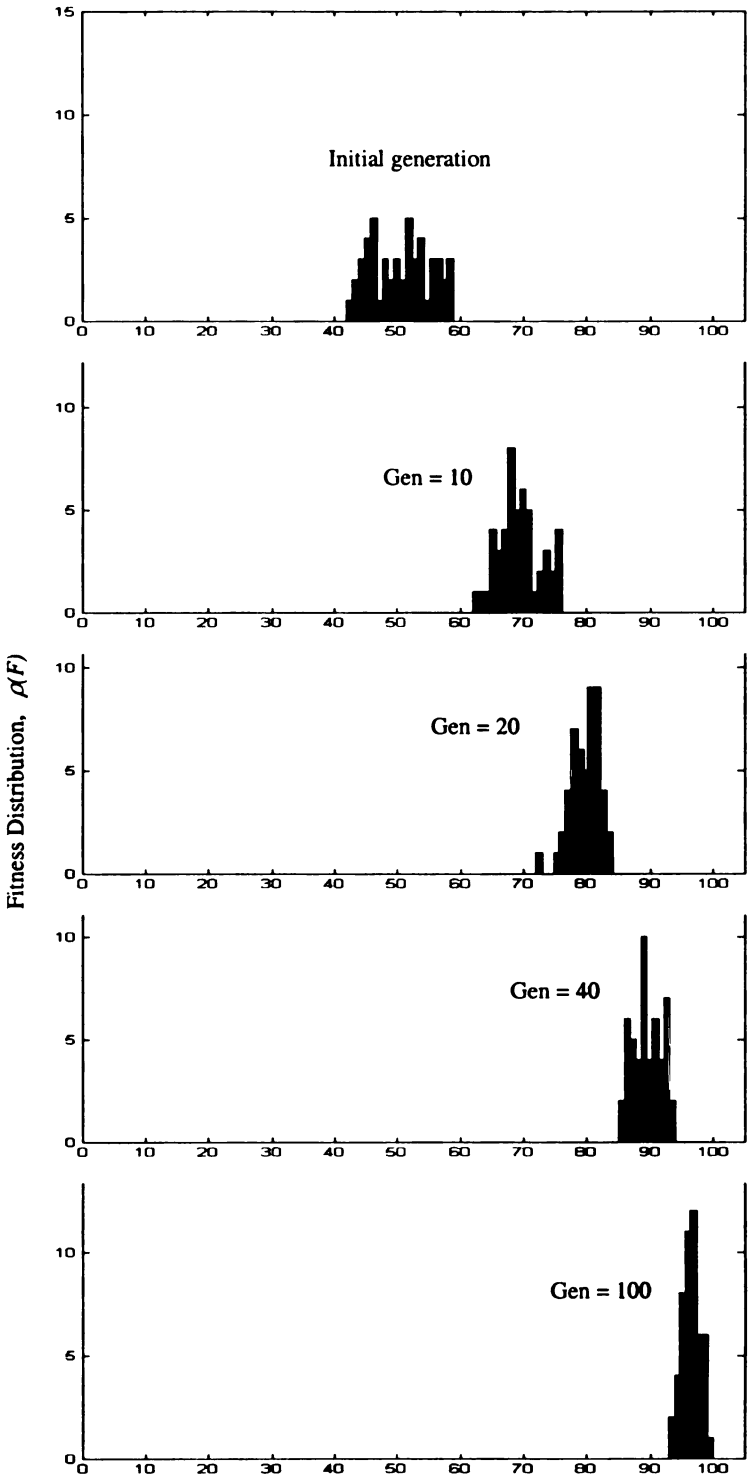


Figure 1.7 Evolution of mean fitness over 100 generations in a GA, for a counting-ones problem. Population size = 50, chromosome length = 100, $\beta = 0.2$

In Figure 1.7, we see the evolution of fitness distribution for a run of the GA. If we apply the GA 1000 times to this problem, starting with different randomly chosen initial populations, and take the average of the fitness distributions, then the picture looks as in Figure 1.8. This figure shows a typical evolution of the average fitness distribution of a GA.

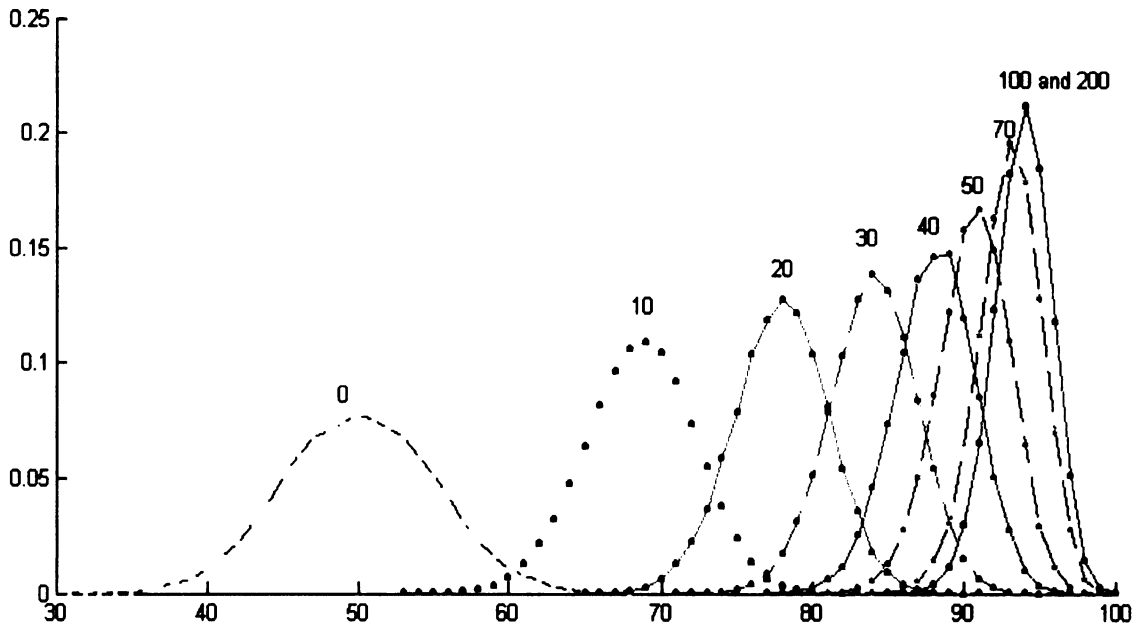


Figure 1.8 Evolution of mean fitness for a counting-ones problem over 200 generations, after the average is taken over 1000 GA runs. Population size = 50, chromosome length = 100, $\beta = 0.2$.

1.2.2 Cumulants as a Tool to Observe GA Evolution

The characteristic function of a fitness distribution $\rho(F)$ is defined by $\Phi(\omega) = \sum_F \rho(F) e^{i\omega F}$. Then the nth moment, μ_n, of this distribution is defined

by the coefficients of the series $\Phi(\omega) = \sum_{n=0} \frac{\mu_n \cdot (i\omega)^n}{n!}$. Similarly, the n^{th} cumulant, κ_n, of this distribution is defined as the coefficients obtained using the logarithm of the characteristic function

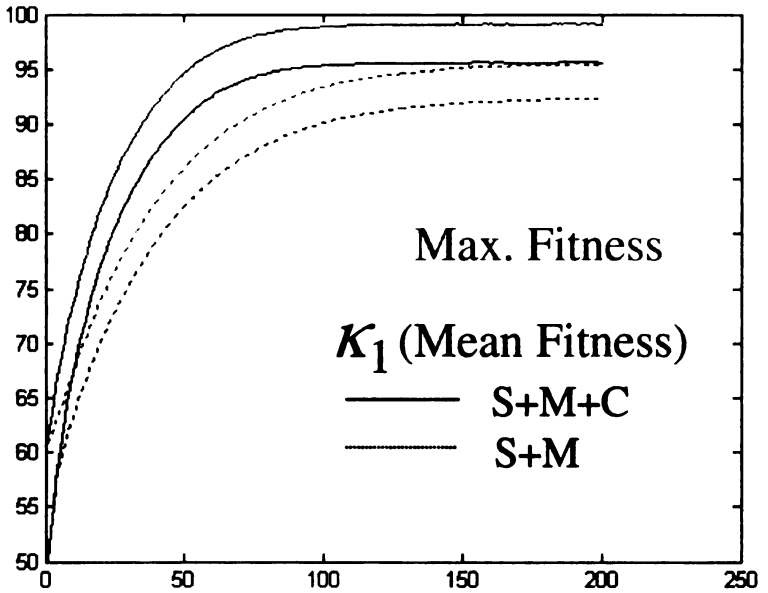
$$\log(\Phi(\omega)) = \sum_{n=1} \frac{\kappa_n \cdot (i\omega)^n}{n!} .$$

It follows from this definition that the first cumulant is just the mean fitness of the population, the second cumulant is the fitness variance, the third one is the skewness and the fourth one is the kurtosis. Cumulants are also related to the moments as

$$\begin{aligned}
\kappa_1 &= \sum_F \rho(F) F = \mu_1 \\
\kappa_2 &= \sum_F \rho(F)(F-\kappa_1)^2 = \mu_2 - \mu_1^2 \\
\kappa_3 &= \sum_F \rho(F)(F-\kappa_1)^3 = \mu_3 - 3\mu_2\mu_1 + 2\mu_1^3 \\
\kappa_4 &= \sum_F \rho(F)(F-\kappa_1)^4 = \mu_4 - 4\mu_3\mu_1 - 3\mu_2^2 + 12\mu_2\mu_1^2 - 6
\end{aligned} \tag{1.3}$$

One of the objectives of GA study is to understand and estimate how cumulants change over time. In Figure 1.10 and Figure 1.10, we have time evolution graphs of $\kappa_1, \kappa_2, \kappa_3$ and κ_4. The skewness and kurtosis are normalized by $\kappa_2^{3/2}$ and κ_2^2 in order to have a better representation of these quantities allowing comparison for populations with different variances. In these graphs, two cases

of a GA are studied. In the first one, selection, mutation and crossover are all applied. In the second case, only selection and mutation are applied in order to observe the effect of the crossover operation. The parameters of the GA are the same as in Section 1.2.1, --i.e., population size $P = 50$, mutation rate $p_m = 0.01$, crossover rate $p_c = 1$, and Boltzmann scaling with Boltzmann constant $\beta = 0.2$. Figures show that crossover operation, on average, increases the fitness variance little bit while giving a more uniform look to the skewness and kurtosis, and gives a better GA performance in terms of its mean fitness values, in the case of our specific problem.



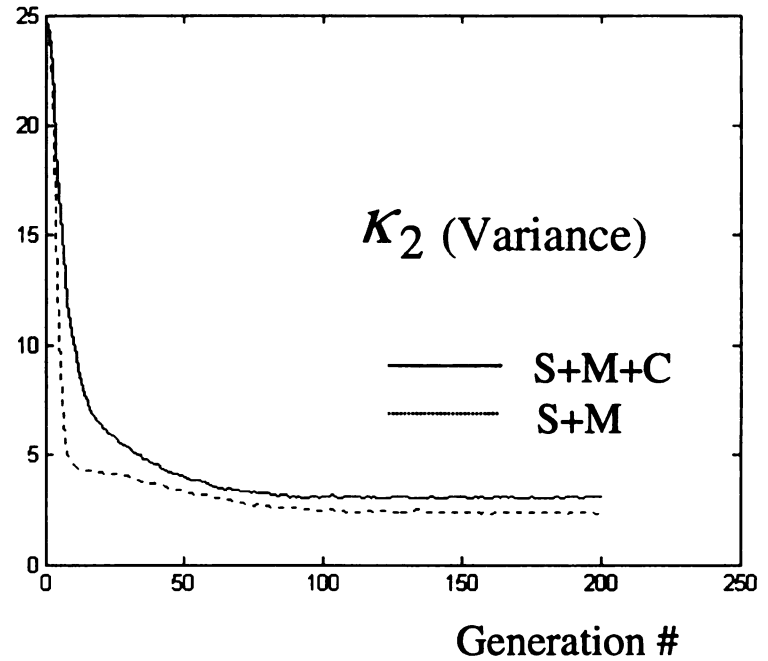
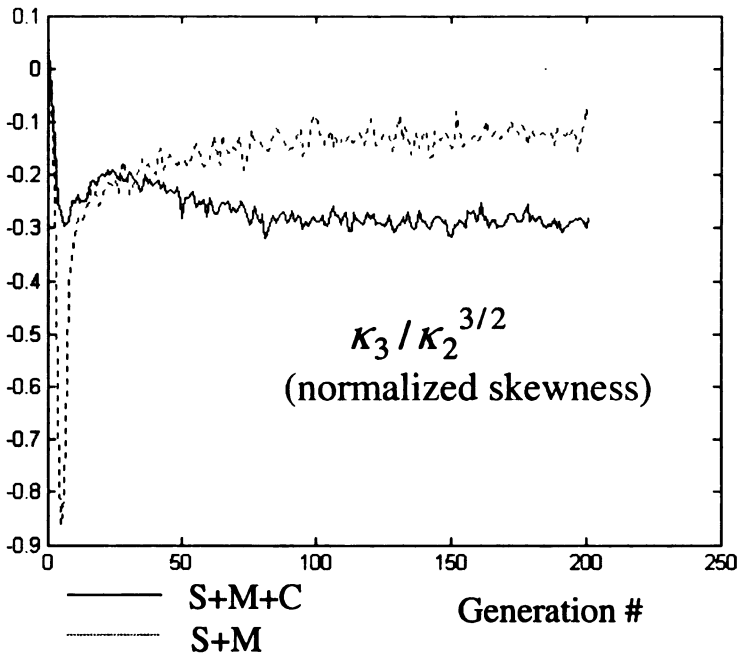


Figure 1.9 The time evolution of cumulants, κ_1 and κ_2, for two cases of GA. First with selection, mutation and crossover, the second with selection and mutation only. The graphs are obtained by averaging 100 GA runs



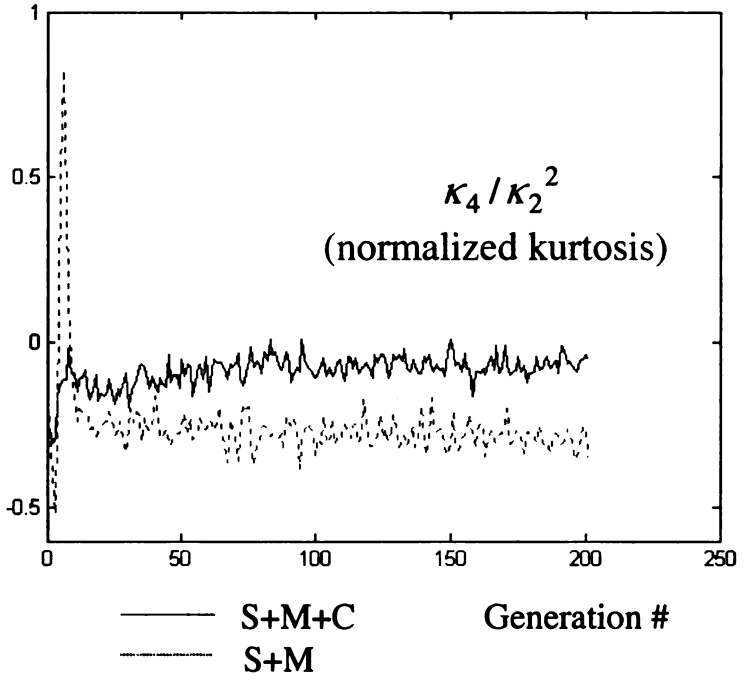


Figure 1.10 The time evolution of cumulants, κ_3 and κ_4, for two cases of GA. First with selection, mutation and crossover, the second with selection and mutation only. The graphs are obtained by averaging 100 GA runs

1.2.3 Counting-Ones and Migration

Although in practice, the migration between two populations which evolve simultaneously and independently from each other, usually occur at later generations when each population created "good" members possibly different than the other population's "good" members, it is still useful to look at the following two specially-structured experiments (CASE2 and CASE3), in which an artificial migration is applied at the very beginning. In the following comparison, the case in which the population evolves normally with no migration is called CASE1. In the second case (CASE2), for each run of the GA, we choose an arbitrary chromosome (migrant) with fixed fitness 65 and place it in the otherwise randomly chosen initial population, so that the migrant is different for each run but always with fitness 65. The third case (CASE3) is similar to CASE2 except that the chromosome with fitness 65 is not chosen randomly but is always the following special chromosome

$$\underbrace{1\ 1\ 1\ 1 \ldots 1}_{65\text{ of them}}\ 0\ 0 \ldots 0.$$

The population size is taken as 50 and the chromosome length is 100. GA includes two-point crossover, Boltzmann selection with scaled $\beta = 0.2$ (see Section 2.4 for a detailed description of this scaled-fitness selection) and no mutation.

In Figure 1.11, we see the fitness density distributions for the first 9 generations of these three cases of GA experiments, with average taken over 100 GA runs. The curves move to the right as time goes. The spike in the fitness distribution of the initial population, i.e. peak at the right of the leftmost curve, for CASE2 and CASE3 is due to the migrant with fitness 65. In these figures, we see how much faster the curves move when there is a migration in comparison to CASE1.

The mean fitness and maximum fitness evolutions of these three cases over 200 generations are shown in Figure 1.12. It shows that, for this specific example, each case converges to the same value but in different rates. In Figure 1.13, we compare the three cases in relation to the time when they reach mean fitness of 95. Table 1.1 shows these times when the mean fitness of each case reaches 95. As we see, both cases of this specially-structured "migration" perform better than the case with no migration, i.e. it takes less time for them to evolve their populations to a mean fitness of 95, on the average. The fact that CASE3 performs much better than CASE2 shows us that it is more beneficial to have long series of adjacent 1's (i.e. "correct" alleles) rather than one with same fitness but scattered 1's. This is because in CASE2 the part of the migrant starting from the first allele with value 1 and ending at the last allele with value 1 is longer than the same length in CASE3. Thus, the crossover is less likely to separate these 1's in CASE3 into different chromosomes, in comparison to CASE2.

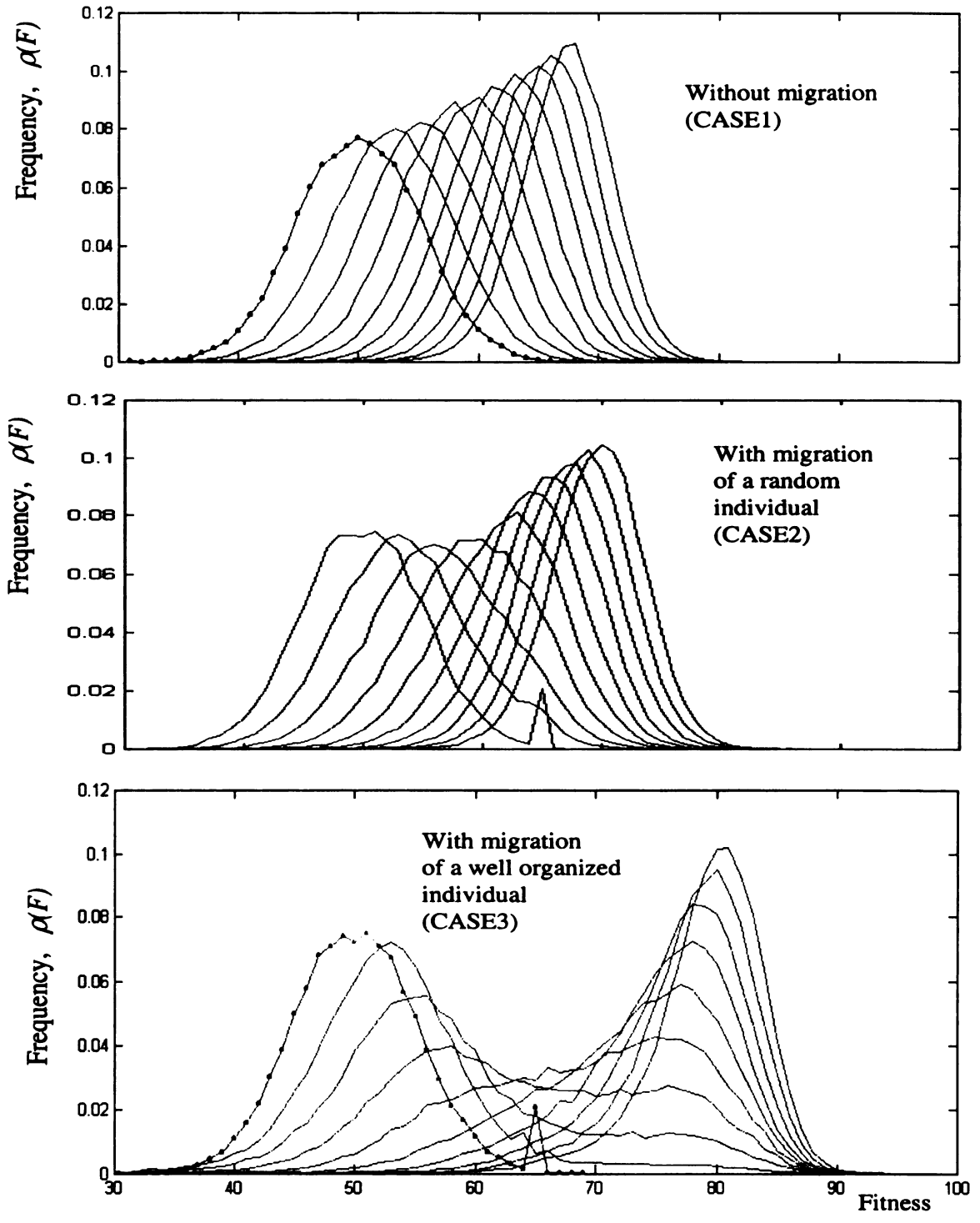


Figure 1.11 Evolution of fitness distributions for CASE1, CASE2 and CASE3

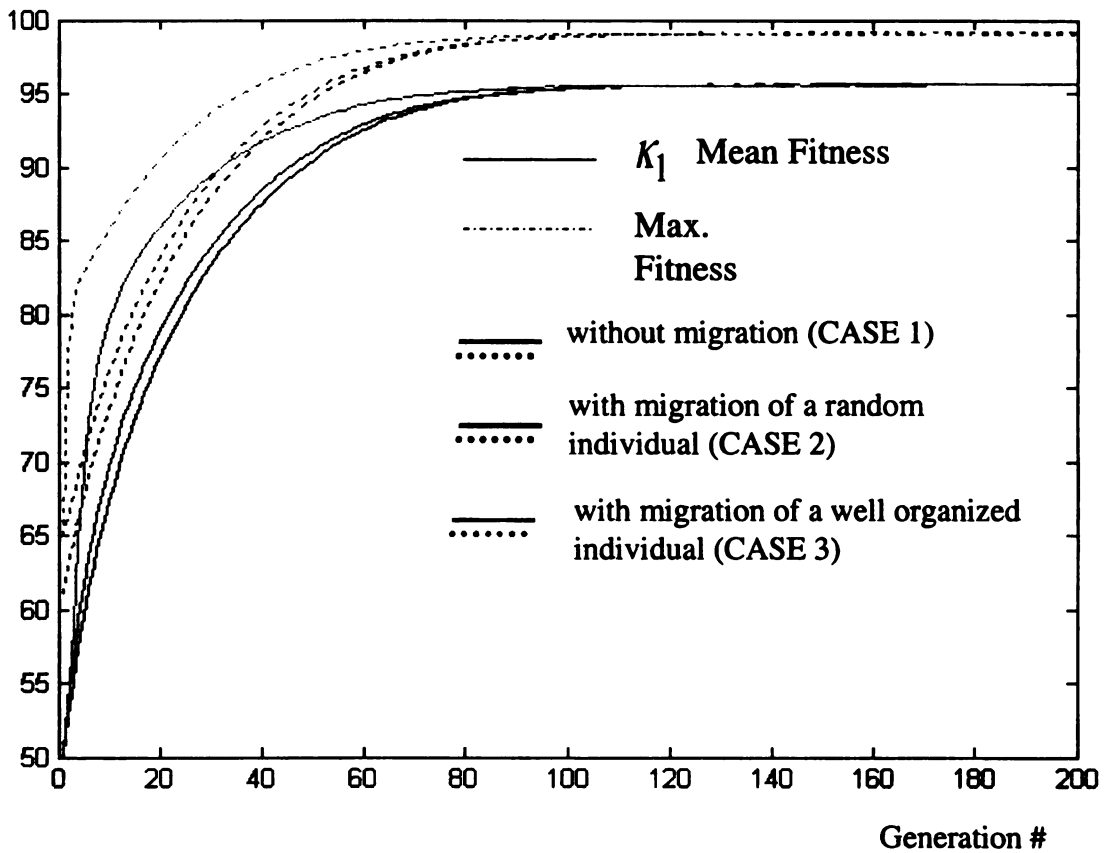


Figure 1.12 Evolution of mean and maximum fitness for CASE1, CASE2 and CASE3

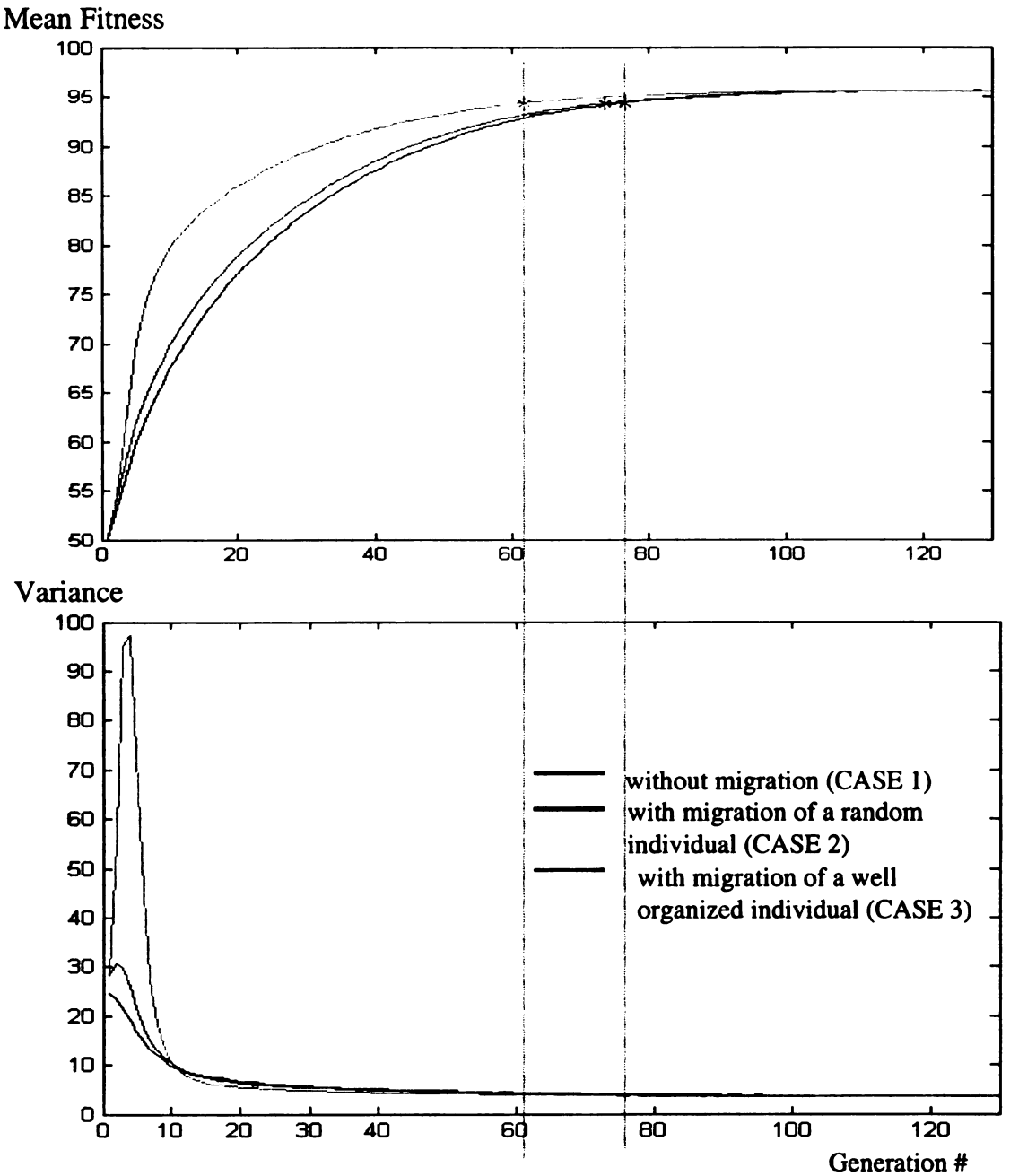


Figure 1.13 Evolution of mean fitness and fitness variance for CASE1, CASE2 and CASE3

Table 1.1 The three cases with the average generation numbers when the mean fitness of 95 is achieved

	Mean Gen #
CASE 1	**76.72**
CASE 2	**74.73**
CASE 3	**61.96**

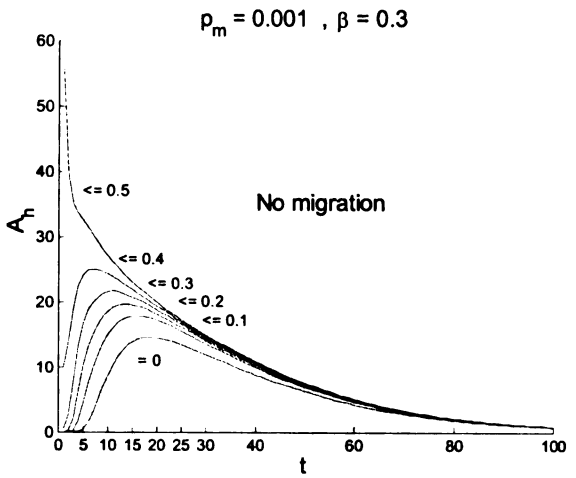
Now, let us look at a more realistic case of migration. In this counting-ones experiment, we have two populations, P1 and P2, of size 100, running simultaneously. We apply two-point crossover with 100% crossover rate, mutation with 0.1% rate, and Boltzmann selection with scaled $\beta = 0.3$ (see section 2.4 for a definition of Boltzmann selection). The case when no mutation is applied is called CASE-A. In CASE-B, 10 members with the highest fitness of P2 is migrated into P1 at generation 20. In CASE-C the migration procedure of CASE-B applied at generations both 20 and 30.

In order to observe the inner structure of the population P1, we count the number of loci in the population at which the mean allele values are less than or equal to a given number, *h*. In Section 2.1, we define these measures more formally and call them A_h. In particular, A_0 counts the number of loci at which all chromosomes in the population has 0 values. Figure 1.14 shows A_h curves for *h= 0, 0.1, ..., 0.5* as a function of time for each case. The migration times 20 and 30, in particular, are chosen by finding the time at which A_0 is maximum, in other

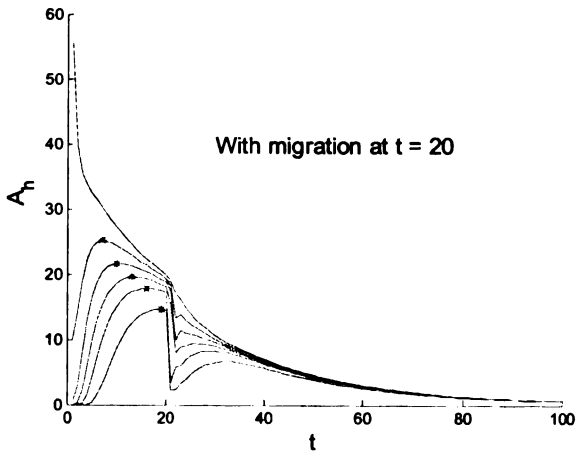
words by finding average times when the number of loci whose values completely converged to zero over all population members reaches to a local maximum. As seen in these graphs, application of each migration pulls down A_h curves towards 0. This means that, on the average, the number of loci with value 1 will increase faster with migration.

This example shows that the migration helps to make the population move or converge faster. In more complicated fitness function problems, this kind of migration will also prevent population converging to a false maximum, i.e. avoid premature convergence or convergence to a non-global local maximum, when the migration parameters are chosen correctly.

CASE-A



CASE-B



CASE-C

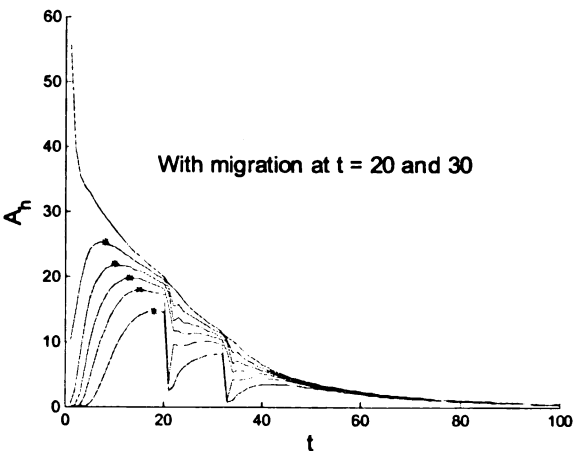


Figure 1.14 A_h curves for three cases. $p_m = 0.001$ and $\beta = 0.3$

1.3 GA with a Real-Life Problem

In this section, we will see an application of a GA to a real-world problem. We will not give the solution of this problem by a GA in this dissertation since our main subject is the modeling of the GA rather than its applications. However, it is illustrative to include this example here to see how complicated the GA problems could be. This problem, suggested by P.J. McCleer of McCleer Power Inc., introduced me to genetic algorithms for the first time. Considering how to solve this problem and trying to determine most appropriate GA parameters and structure drove me to seeking how to understand GA dynamics in a more general framework.

The problem involves the automotive 42-Volt DC electrical systems of hybrid cars. The system uses pulse-width-modulated inverters to convert the bus voltage to a 3-phase AC voltage with desired amplitude and frequency. In Figure 1.16, we see a simplified circuit of this system.

DC voltage can be converted to a sinusoidal AC voltage, either single phase or three phase, with desired amplitude and frequency, by means of electrical devices called "switch-mode inverters." The basic idea of these inverters is sketched in Figure 1.15. The DC voltage is converted to an AC voltage that alternates between values 0 and a certain fixed voltage. Only the fundamental component of this voltage is to be used as the output. Thus, the high frequency harmonics are filtered and the desired sinusoidal voltage is obtained.

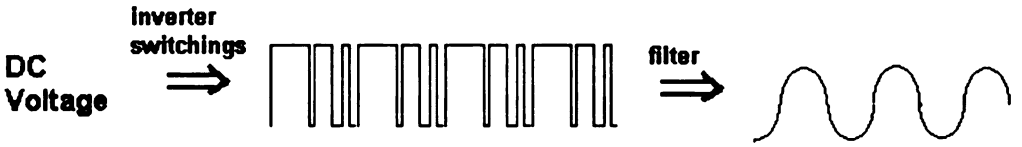


Figure 1.15 Passage from DC to AC voltage by switch-mode inverters

By turning the switches, which are the six transistors S_{1a}, S_{2a}, S_{3a}, S_{1b}, S_{2b} and S_{3b} in Figure 1.16 on and off, the desired output power specifications are achieved while providing an output voltage with a specific amplitude and frequency but without unnecessary harmonics. One restriction in the functioning of the switches is that, in a vertical pair, such as S_{1a} and S_{1b}, the transistors can not be both on or off at the same time. Otherwise, this would cause a short circuit. Thus, it is enough to determine the status of one transistor in each pair. The whole art of switching scheme design comes into play at this point.

The practice in automotive inverters is to control the switches by a central processing unit (CPU), which can be programmed to turn the switches on and off in any desired pattern. The switching is accomplished with high-current field-effect transistors (FET). The advantage of this method is the ability to control the fundamental component while eliminating some of the harmonics. However, one should be careful with the frequency of switching since, in practice, if a switch

turns off in an inverter leg, the turn-on of the other switch is delayed by a blanking time, which introduces low-order harmonics in the output.

On the other hand, ripples in the output current result in ripples in the current through the capacitors. This ripple current causes the capacitors to warm up over time, which is very undesirable since the temperature under the engine hood under normal operating conditions can approach 120°C. Such high temperatures can damage the capacitors.

Because of the high cost of these capacitors, any reduction in their number will reduce the cost of the inverter. Reducing the number of capacitors can be obtained by reducing the current i_c. Hence, the main focus of our design is to minimize i_c while still meeting the desired output voltage and/or current specifications. As a result, the question is "what is the best switching algorithm which will produce minimum current i_c through the capacitor C?"

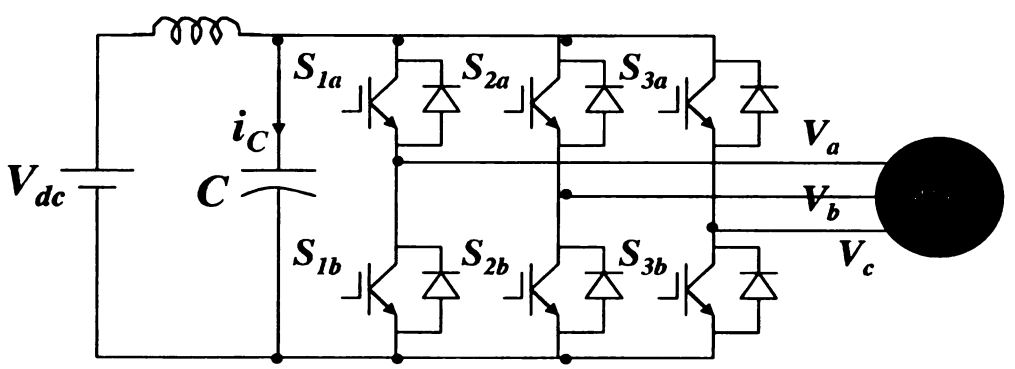


Figure 1.16 A simplified circuit of pulse-width-modulated inverter.

Let's say we want to determine the timing for S_{1a}, S_{2a} and S_{3a} (timing of S_{1b}, S_{2b} and S_{3b} are determined from them). A good solution to this problem can be achieved by a GA in the following way. Split the chromosome into 3 equal parts (one part for each vertical pair of transistors). For each part, the alleles (values) at successive loci (fields) of the chromosome are interpreted as the relative lengths of successive on and off intervals, of the corresponding transistor. We use discrete values for the time lengths. Each allele, let's say, can take on a 4-bit value, so the values 0,1,2,...,15 are the possible field values (alleles) at each locus. We assume a maximum of 101 ON and OFFs for each transistor during a period. So each part consists of 101 loci and the total chromosome length is, then, 3x101= 303 loci. We take the fitness function as the mean squared value of the current through the capacitor. The function is restricted to the domain in which we have the correct amplitude and frequency of the sinusoidal output (in a different application, the deviation from these constraints can also be put as a part of fitness function with negative weights.) Thus, we have a well-defined GA problem, whose application could produce switching algorithms that are more efficient than traditional methods.

Recently, GA is, in fact, successfully applied to power inverters to determine the switching angle, i.e. time delay between switching times of the transistors, eliminating high order harmonics of the output voltage, (Ozpineci, et al [2004].)

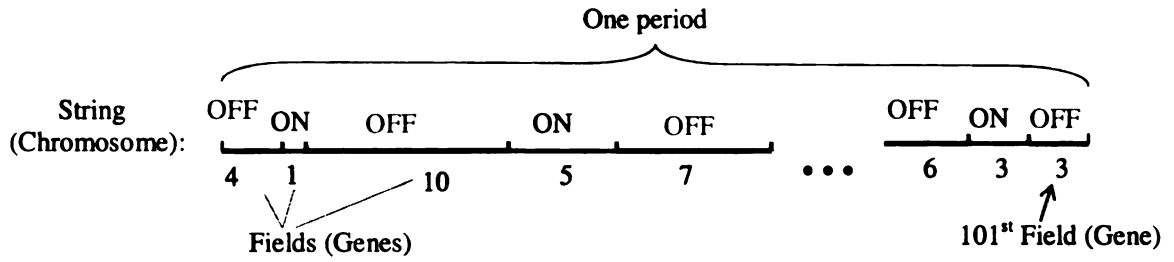


Figure 1.17 The part of the chromosome representing timing of a transistor pair during one period of the desired output voltage

1.4 Theoretical Models of GA

Some of the applications of GA may not converge to a desired value or sometimes would take months or years to achieve a good solution just because the design parameters are not chosen correctly. In practice, the fitness function is often very complicated and the crossover operation is a very complex mixing operator, which makes it quite difficult to analyze GAs theoretically. Several theories have been developed in order to understand why and how GAs work, and in order to choose the design parameters wisely. Theoretical models of Genetic Algorithms (GAs) fall into three main categories. The Markov chain model, as developed by Nix and Vose [1991], completely describes the probabilistic behavior of the GA. However, this model is too costly to implement computationally for problems with realistic population size and chromosome length. The statistical mechanics approach, developed by Prügel-Bennett, Shapiro [1994] and Rattray [1996], gives fairly good results in modeling the OneMax problem with Boltzmann scaling, for a crossover rate of 100%, however it is not developed for lower crossover rates or to handle other benchmark

problems of GA such as deceptive functions. The approach of modeling GAs by considering building blocks (Goldberg [1989] and Goldberg, Deb, Thierens [1993]), on the other hand, gives us a good idea about the appropriate population size or the convergence time of the OneMax and help us determine the failure boundaries in the "control maps". But the question of finding the most appropriate crossover or mutation rate is answered, so far, only by experimental results. We still lack a model that describes the behavior of the OneMax problem for different crossover and mutation rates together and allows us to choose the best parameters.

1.4.1 Schema Theory

In this theory, the search space is divided into subspaces called "schemata." The aim is to characterize the schemata using macroscopic quantities, such as the number of individuals within a given schema H at generation t, denoted by $m(H,t)$, average fitness of individuals in the schema and in the population, size of the search space, size of the schema, etc. Schema theorems model thematically how and why $m(H,t)$ varies from one generations to the next. Since GAs are non-deterministic, one can only predict the expected value, $E[m(H,t)]$, of $m(H,t)$. As an example, consider a search space consisting of 5-bit binary chromosomes. A special subset of this search space can be described by the template, H = (1 0 * * 1), which means that the first, second and the last loci are fixed as 1, 0 and 1, respectively, while the third and fourth

loci can take either of the binary values. Hence the schema H consists of 4 chromosomes, Figure 1.18.

$$H = (1\ 0\ *\ *\ 1) \longrightarrow \{1\ 0\ 1\ 1\ 1,\ \ 1\ 0\ 1\ 0\ 1,\ \ 1\ 0\ 0\ 1\ 1,\ \ 1\ 0\ 0\ 0\ 1\}$$

order : o(H) = 3 defining length : δ(H) = 5 - 1 = 4

Figure 1.18 Template H

The order, $o(H)$, of a schema H is defined to be the number of fixed digits within the schema. The defining length, $\delta(H)$, of H is defined to be the distance between the first and the last fixed string positions in the schema. For example, for H = (1 0 * * 1) above, we have $o(H)$=3 and $\delta(H)$ = 5-1 = 4.

The first schema theorem was developed by J. H. Holland in the 1960's, taught in his classes and used by his students in their theses, and made widely available in his 1975 book. Different versions of this theorem were later developed and published by Goldberg [1989], Whitley [1994], and Stephens and Waelbroeck [1997]. In one form, the schema theorem states that

$$E[m(H,t+1)] \geq M \cdot p(H,t)(1-p_m)^{o(H)}\left[1 - p_c \frac{\delta(H)}{N-1}\sigma\right] \tag{1.4}$$

where

M : population size,

N : string length,

$p(H,t)$: selection probability of individuals in H at generation t,

p_m : mutation probability,

p_c : crossover probability.

The term σ in Equation (1.4) is taken as $1-m(H,t)/H$ by Holland, as 1 by Goldberg, and as $1-p(H,t)$ by Whitley. In Equation (1.4), the term $M \cdot p(H,t)$ gives the expected number of population members which are instances of the schema H after the selection operation. For example, if fitness-proportional selection is applied, then we have

$$M \cdot p(H,t) = m(H,t)\frac{f(H,t)}{\overline{f(t)}} \tag{1.5}$$

where $f(H,t)$ is the average fitness of strings representing H within the population and $\overline{f(t)}$ is the average fitness of the whole population. After selection, we can calculate the probability that each individual of H within the population will survive the changes made by mutation and crossover. The terms $(1-p_m)^{o(H)}$ and $1-p_c\frac{\delta(H)}{N-1}\sigma$ in Equation (1.4) represent these survival probabilities respectively. Equation (1.4) gives only a lower bound for

$E[m(H,t+1)]$ since it does not consider the newly created instances of schema H after mutation and crossover. On the other hand, we gain some insight into why GAs work by examining this equation. For example, consider Goldberg's version of the schema theorem with fitness-proportional selection:

$$E[m(H,t+1)] \geq m(H,t) \cdot \underbrace{\frac{f(H,t)}{\overline{f(t)}}}_{\text{selection}} \underbrace{(1-p_m)^{o(H)}}_{\text{mutation}} \underbrace{\left[1-p_c \frac{\delta(H)}{N-1}\right]}_{\text{crossover}} \tag{1.6}$$

Equation (1.6) tells us that, if the average fitness of the schema, $f(H,t)$, is greater than the average fitness of the whole population — that is, if $\frac{f(H,t)}{\overline{f(t)}} > 1$, then, provided that $o(H)$ and $\delta(H)$ are small enough, the expected number of instances of H in the population increases exponentially over generations. By small enough values of $o(H)$ and $\delta(H)$, we mean that the values of $o(H)$, $\delta(H)$, p_m and p_c satisfy

$$\frac{f(H,t)}{\overline{f(t)}}(1-p_m)^{o(H)}\left[1-p_c \frac{\delta(H)}{N-1}\right] > 1 \tag{1.7}$$

By means of this observation, Goldberg, 1989, has defined the notion of Building Blocks (BB), which are low-order, low–defining-length schemata of above average fitness – in other words, those schemata satisfying the condition above. Hence, the well-known "Building Block Hypothesis" of Goldberg says that

"A GA works by combining BBs to form higher-order BBs until it converges to an optimum or near-optimum solution."

It is worthwhile to emphasize that Equation (1.4) works for all possible schemata independently and in parallel. Later, in 1997, Stephens and Waelbroeck, developed another schema theorem, which gives an exact equality for $E[m(H,t+1)]$ by the formula

$$E[m(H,t+1)/M] = (1-p_c)p(H,t) + \frac{p_c}{N-1}\sum_{i=1}^{N-1} p(L(H,i),t)p(R(H,i),t) \tag{1.8}$$

where $L(H,i)$ is the schema that is obtained by replacing the elements of H to the right of position *i* with *'s, and $R(H,i)$ is the schema that is obtained by replacing the elements of H to the left of position i with *'s.

Schema theory is applied to modeling of single populations and to determining good population sizes by G. Harik, D.E. Goldberg, , E. Cantu-Paz and B.L. Miller [1999] and later to modeling of parallel GAs by E. Cantu-Paz [2001].

1.4.2 Random Walk Model

The random walk model is used with schema theory to determine the appropriate population size N (G. Harik, D.E. Goldberg, , E. Cantu-Paz and B.L. Miller [1999]). Let x_0 be the initial number of BBs in the population and the variable x represent the number of BBs at any time. First, for a given

configuration of the population of size N, the probability, *p*, of producing one additional building block is calculated. Then one can visualize the dynamics of the number of BBs as a one-dimensional random walk as shown in Figure 1.19

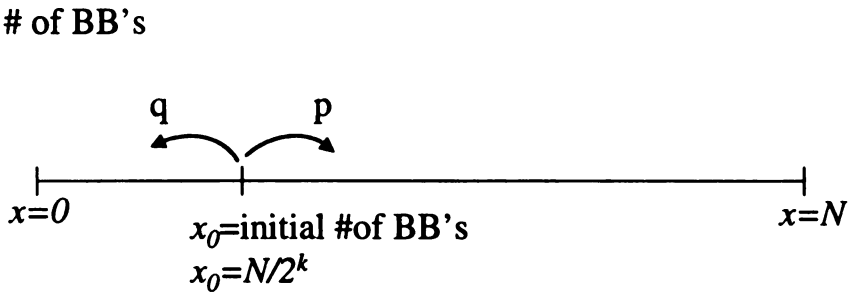


Figure 1.19 The Random Walk

Then the probability, *P*, that this random walk converges to $x = N$ is calculated. Specifying an expected value of *P* will determine the correct size of the population that would result in this expected value being met or exceeded.

1.4.3 Markov Chain Model

In this model, genetic operators are described by transition matrices acting on a vector describing the precise state of the population, (Nix and Vose [1991], Vose and Liepins [1991], Suzuki [1995].)

In a search space where there are *n* possible points (i.e., chromosomes), let $p = (p_0, p_1, \ldots, p_{n-1},)$ represent a population where p_k is the proportion of the population occupied by item *k*. Let $q = (q_0, q_1, \ldots, q_{n-1},)$ be the probabilities that each item is generated in the next population. Consider the next population as P (the population size) independent samples of the search space, using *q* as

a probability distribution. Representing the chromosome length by L, we have $n = 2^L$. Note that, the size of the state space, i.e. the number of different populations of size P, is given by $\binom{P+2^L-1}{P}$, which can be approximated by $\frac{2^{LP}}{P!}$ when P very small compared to 2^L. This number easily reaches to billions even for very small values of P and L.

Think of the action of a GA as a map $G : \Lambda \to \Lambda$, where

$$\Lambda = \left\{ x \in \Re^n : x_k \geq 0, \sum x_k = 1 \right\} \tag{1.9}$$

Then the probability that population *q* follows population *p* is given by a multinomial distribution

$$P! \prod_{j=0}^{n-1} \frac{\left(G(p)_j\right)^{Pq_j}}{(Pq_j)!} \tag{1.10}$$

where $G(p)$ is the expected next population, $G(p)_j$ is the j^{th} entry of the probability vector $G(p)$, i.e. the probability that the j^{th} chromosome will be selected for the next generation.

Iterating *G* will produce a sequence of points. G describes the limiting behavior as the population size grow large. Then, we have the following theorem [Vose, 1999].

Theorem: Given an initial population *p*, let *q* be the actual population observed after *t* generations. Then, for any $\varepsilon > 0$ and $0 < \delta < 1$, there exists a number *K* such that if the population size is bigger than *K*, then the probability that $\left\| G^t(p) - q \right\| < \varepsilon$ is greater than $1 - \delta$.

1.4.4 Statistical Mechanics Model

The population is described by a small set of macroscopic parameters under the assumption that microscopic details are not of critical importance, by A. Prügel-Bennett, J.L. Shapiro [1994,1997], and M. Rattray [1996]. They represent the fitness distribution by its cumulants and determine the effect of selection, mutation and crossover on each of the cumulants. Cumulants are more natural to use in this kind of representation instead of moments of the fitness distribution since they are *self-averaging*, i.e. their average represent a *typical* member of possible distributions, while moments are not self-averaging, (see Prügel-Bennett, [2002] for a discussion of this comparison.) Using these results, the shape of the next population fitness distribution is determined, (see Figure 1.20.)

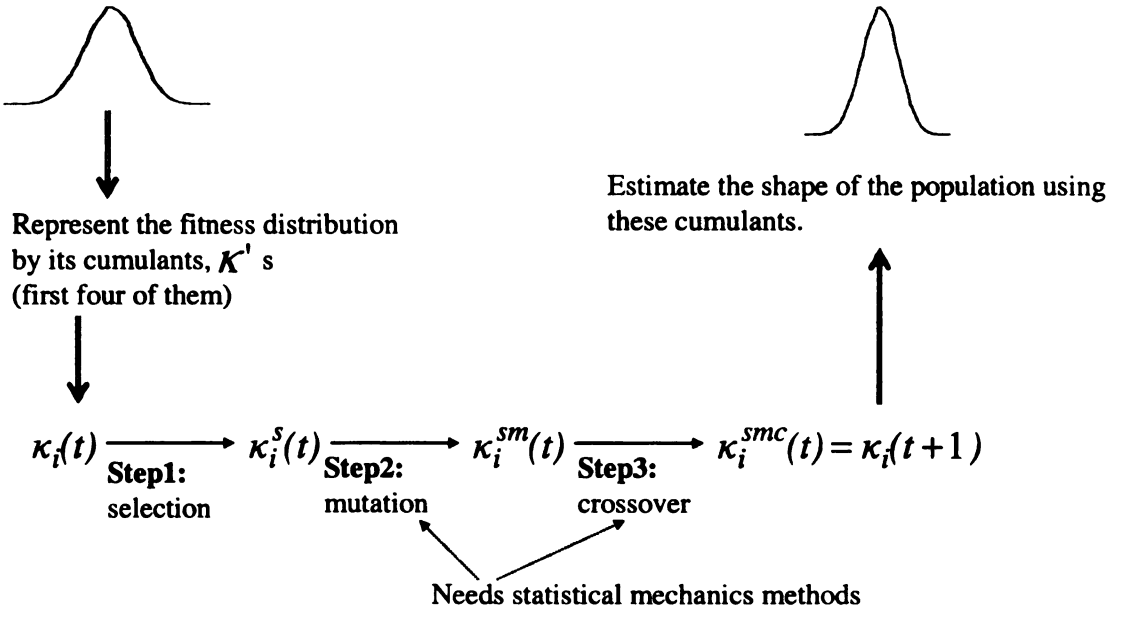


Figure 1.20 The statistical mechanics model applied to cumulants

The objective is to determine average allele per site, average correlation per site, etc. (terms like $< a_i^j >_j$, $< a_i^j a_i^k >_{j \neq k}$). The gene variables are assumed to be free to fluctuate subject to the constraint that the macroscopic quantities chosen are satisfied. Then, the distribution of the allele values is assumed to be the one which maximizes the entropy. As an example of their calculations, consider the mean allele value at the i[th] locus

$$\alpha_i = \left\langle \tau_i^j \right\rangle_j = \frac{1}{P} \sum_{j=1}^{P} \tau_i^j \ .$$

Let $N(\alpha_i)$ be the number of different ways that would give α_i as mean allele value at the i^{th} locus. Then, the entropy for this locus would be

$S(\alpha_i) = \log(N(\alpha_i))$. Introduce Lagrange multipliers, x and y, to enforce constraints on mean fitness and correlation

$$y\sum_{j=1}\sum_{i} a_i^j = yP\sum_i \alpha_i = yP\kappa_1 \qquad \text{: for mean fitness}$$

$$\frac{x^2}{2}\sum_j\sum_k\sum_i a_i^j a_i^k = \frac{x^2}{2}P^2\sum_i \alpha_i^2 \qquad \text{: for allele correlation}$$

Then, the probability distribution for a $\{\alpha_i\}$ configuration is

$$\mathrm{P}(\{\alpha_i\}) = \prod_{i=1}^{L} \mathrm{p}(\alpha_i) = \prod_{i=1}^{L} e^{S(\alpha_i) + yP\alpha_i + (Px\alpha_i)^2/2}$$

The maximal value of $\mathrm{p}(\alpha_i)$ with respect to α_i gives the maximum entropy distribution for α_i. So, from derivatives with respect to α_i, one gets a relationship between α_i and the Lagrange variables x and y. Using this expression of α_i in terms of x and y one can write down the previously chosen macroscopic variables, such as mean fitness, κ_1, and fitness variance, κ_2. When the defining equations of κ_1 and κ_2 are written for the counting-ones problem, and the average over all possible crossover operations and mutation operations is taken, one sees that average mean fitness or the fitness variance does not depend on the mean allele values. Thus, the expected values of κ_1 and κ_2 can be calculated after mutation or crossover, and, we can find the values of x and y using them. Finally, using these values of Lagrange multipliers, one finds the

values of mean allele, α_i. Then, α_i's are used to estimate κ_3 and κ_4 values. Finally, the shape of the fitness distribution for the next generation is estimated using the first four cumulants.

Recently, the maximum entropy technique is applied by Whitley [2004], also to determine the shape of the fitness distribution from schema frequencies.

1.5 The Need for a New Model

Studying the OneMax problem is important not for solution of that problem, per se, but because many real-world problems solved via genetic algorithms consist of a set of separable sub-problems for which the optimum is to optimize each individually, which is reminiscent of OneMax. When we look at all the models mentioned in this chapter, we have the following picture:

The Markov Chain Model uses huge matrices to model the dynamics of a GA and is not applicable in practice, although it gives a good idea about some general features of GA evolution when applied to simple cases with too small population size and chromosome lengths. Its application to infinite populations give an exact description of this case, but, as pointed out by Prügel-Bennett [2002], only very large population size gives results close to infinite population case. This model, yet, is not applicable to problems with typical population sizes. It also requires the full knowledge of the fitness function.

Schema Theory assumes that the building blocks are already (partially) known. Assembly of building blocks is useful to observe to understand GA performance for many problems. This theory is hard to generalize to more realistic cases since it has many simplifying assumptions, and loses its applicability as populations lose their initial random character.

The Statistical Mechanics Method requires explicit knowledge of the fitness function. This method makes very good predictions about the evolution of the fitness distribution. However, it is applicable in practice only to problems with very simple fitness functions. Effects of crossover are particularly hard to estimate using this method for most classes of real-world problems.

So far, none of these models have been successfully applied to the analysis of parallel genetic algorithm behavior that involves migration before the population converges. Estimation of optimal times for performing such migrations, for real-world problems such as that of McCleer Power, Section 1.3, was among the initial motivators for the models developed in this thesis research. The examples in Section 1.2.3, illustrate that a model that predicts the mean allele evolutions can be applied to migration analysis.

Although the models mentioned above are applicable to some cases of OneMax problem, there is no complete model predicting dynamics of deceptive functions (a class of functions that misleads the GA in finding the optimum. See Section 3.1 for a type of deceptive function).

As a result, we need a new model that can be applied to cover crossover rates lower than 100%, and also be used for migration analysis. Moreover, this model should also be simple enough, both theoretically and computationally, to potentially apply to a real-life problem.

Chapter 2

A MODEL OF GA DYNAMICS FOR THE ONEMAX PROBLEM

2.1 Problem Description and a Visual Representation of the OneMax Dynamics

In this section, we study the OneMax problem. We consider the simple genetic algorithm in which two-point crossover, fitness-proportional selection and mutation are applied in the order given. Note that, "canonical" GAs typically apply selection before or after the crossover and mutation are applied. However, considering selection in between crossover and mutation does not make much of a difference in estimating the long term behavior of GAs since the essential difference between them is only at the very beginning or very end of the GA run, as illustrated in Figure 2.1. Also note that the order of mutation and crossover can be changed without making any difference in GAs dynamics since each gene remains in the population during these two operations and is equally likely to be changed by mutation before or after crossover.

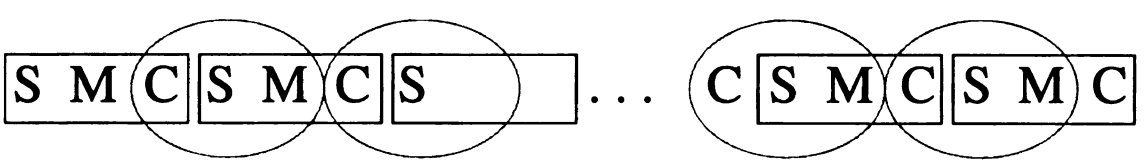


Figure 2.1 Selection-mutation-crossover versus crossover-selection-mutation

In this chapter, we develop a model of genetic algorithm behavior on the OneMax problem with a population consisting of *P* chromosomes of length *L* (see also Buyukbozkirli and Goodman [2004].) Let *S(t)* be the set of all chromosomes at time *t*, *chrom* an element of this set, and *chrom(i)* the allele at the i^{th} locus of this chromosome, in other words, for $chrom_k = (a_1^k, a_2^k, \ldots, a_L^k)$ the i^{th} allele is $chrom_k(i) = a_i^k$ .The fitness of a chromosome, *chrom*, will be denoted as $f(chrom)$. So, for the OneMax problem,

$$f(chrom_k) = \sum_i chrom_k(i) = a_1^k + a_2^k + \ldots + a_L^k, \tag{2.1}$$

where the values of a_i^k's are 1 or 0.

The population-level variables that we are interested in are the mean fitness $\kappa_1(t)$, the variance of the fitness $\kappa_2(t)$, and the set of means of the alleles at each locus *i* $\{\alpha_i(t)\}_{i=1,\ldots,L}$, at time *t*. They are given by the formulae

$$\begin{aligned}
\kappa_1(t) &= \frac{1}{P}\sum_{k=1}^{P} f(chrom_k) = \frac{1}{P}\sum_{k=1}^{P}\sum_{i=1}^{L} a_i^k, \\
\kappa_2(t) &= \frac{1}{P}\left(\sum_{k=1}^{P} f(chrom_k)^2\right) - \kappa_1(t)^2, \\
\alpha_i(t) &= \frac{1}{P}\sum_{k=1}^{P} chrom_k(i) = \frac{1}{P}\sum_{k=1}^{P} a_i^k(t).
\end{aligned} \tag{2.2}$$

In the case of the "weighted fitness function" for a OneMax problem, each allele *i* of the chromosome is weighted by a constant weight, w_i'. In this case, the fitness of a chromosome is given by the weighted sum

$$f(chrom_k) = \sum_i w_i \cdot chrom_k(i) = w_1 a_1^k + w_2 a_2^k + ... + w_L a_L^k. \tag{2.3}$$

Figure 2.2 shows a sample population at time *t* together with definitions of some of its parameters.

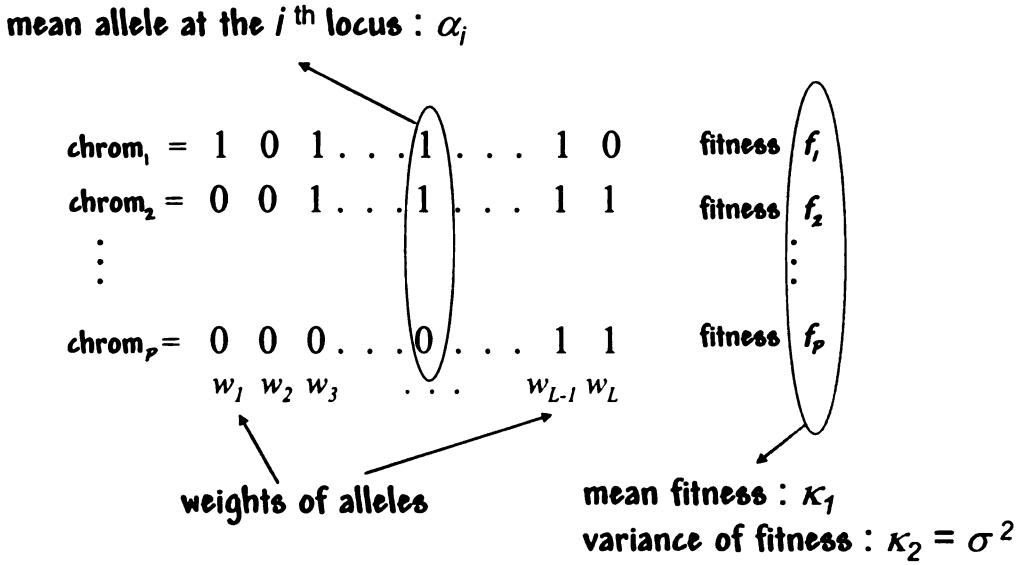


Figure 2.2 The chromosomes in the population, mean allele and other notations

In Table 2.1, we see a list of the notations used in this chapter in the development of the model.

Table 2.1 Notations

P : population size

L : chromosome length

$a_i^k(t)$: the i^{th} allele of the k^{th} chromosome at generation t

$\alpha_i(t)$: the mean allele at the i^{th} locus at generation t

$f\ (chrom_k)$: the fitness of the k^{th} chromosome at generation t

$\kappa_1(t)$: the mean fitness

$\kappa_2(t)$: the variance of the fitness

p_m : mutation probability of each allele, in decimal form

p_c : crossover rate, in decimal form

$S(t)$: the set of all chromosomes at time t

$p_i(t)$: the probability that a chromosome that is selected randomly at time t with the probability scheme of the selection, has 1 at its i^{th} locus

When we study a particular run of a GA it is useful to look at its mean allele values for each locus and observe the way they change at each generation. In Figure 2.3, we see the mean allele values at times 0, 10, 20, 30, 50 and 100. The GA parameters of this OneMax problem are: chromosome length L=100, population size P=50, crossover rate p_c = 0.25 and mutation rate p_m = 0.001.

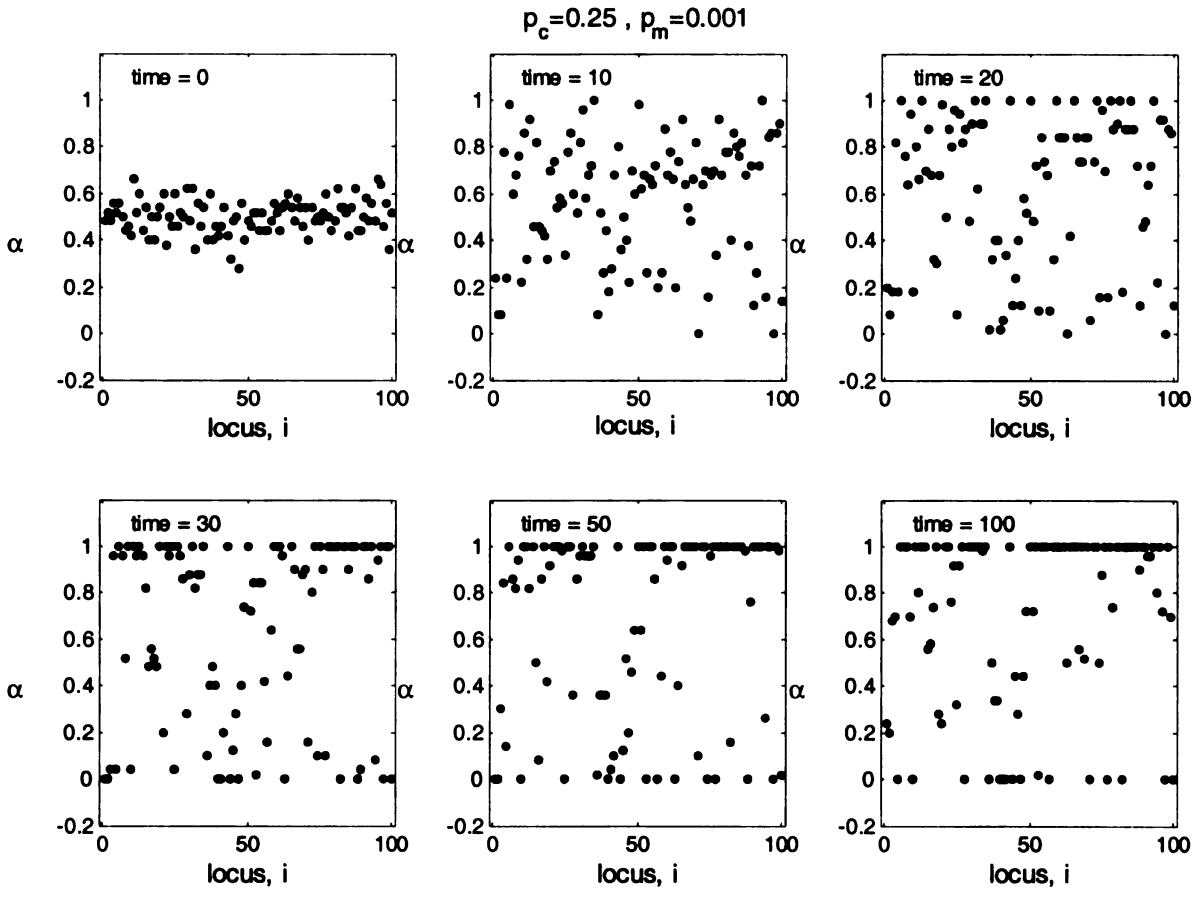


Figure 2.3 The mean allele values, α_i's at each locus *i* for times t =0, 10, 20, 30, 50 and 100 of a GA run with p_c=0.25 and p_m=0.001.

Define $A_h(t)$ to be the number of $\alpha_i(t)$'s whose values are less than or equal to *h*,

$$A_h(t) = \#\left\{ \alpha_i(t) \,\middle|\, \alpha_i(t) \le h, i = 1,\ldots,L \right\} \tag{2.4}$$

By this definition, for example, $A_0(t)$ gives the number of loci where all of the chromosomes have value 0, while $A_{0.6}(t)$ gives the number of loci where at

most 60% of the chromosomes have value 1. For instance, in Figure 2.3, we have $A_0(30) = 9$; i.e., at the 30th generation, at 9 of the loci, all the chromosomes have value 0. So, for this example of the GA, at generation 30, the allele value 1 is completely lost at 9 positions of the chromosomes. $A_1(t)$, by definition, is always equal to *L*.

In Figure 2.4, we see how $A_{0.4}(t)$ is defined and the time evolution of $A_h(t)$ for *h*=0, 0.1, 0.2, ... 0.9. The time evolution graphs of $A_h(t)$ in Figure 2.4 are obtained by taking the average of 100 GA runs.

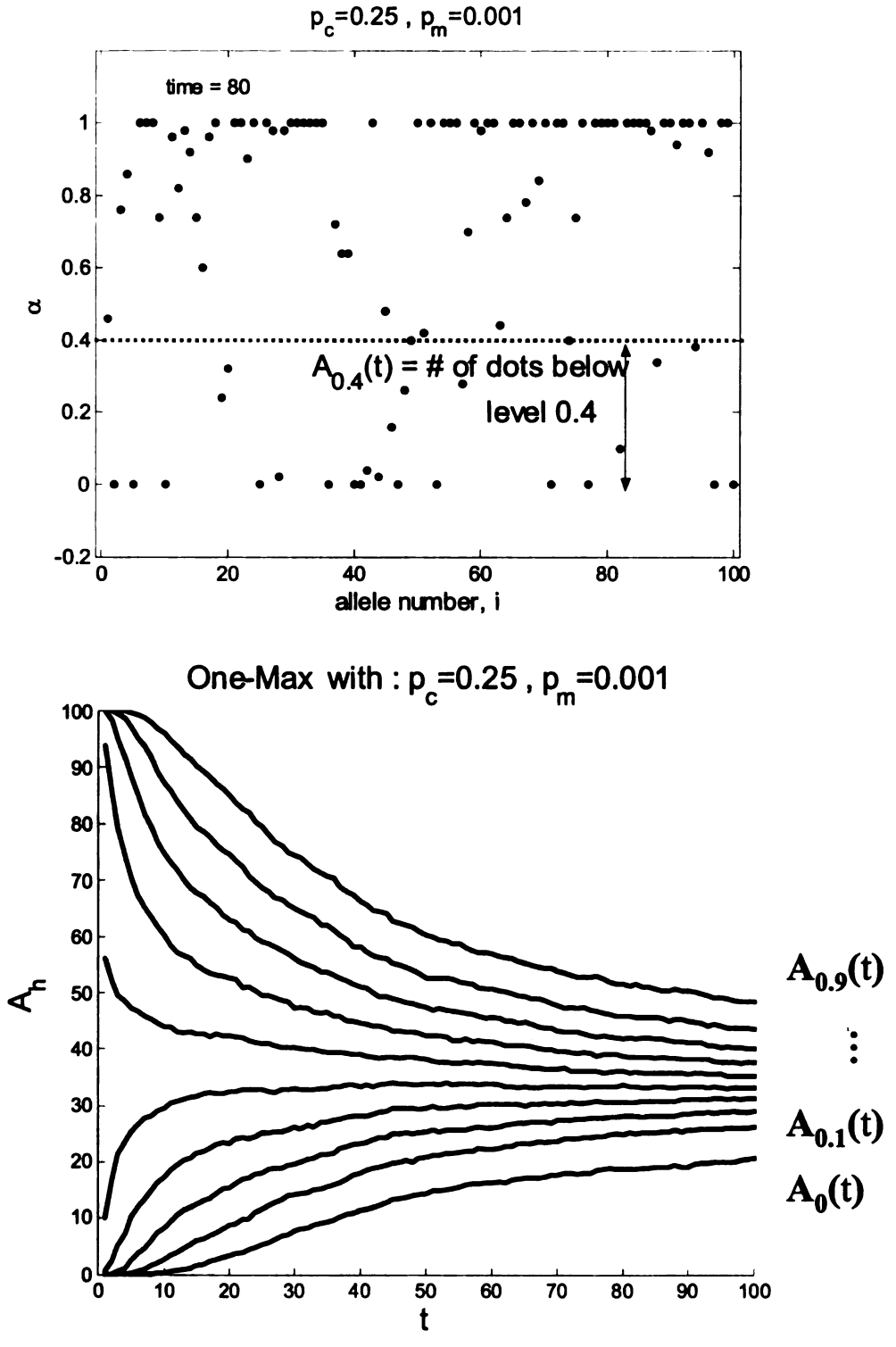


Figure 2.4 Definition of $A_h(t)$

The values of the variables $(\kappa_1(t), \kappa_2(t), \{\alpha_i(t)\}_{i=1,..,L})$ and $A_h(t)$ change from one GA run to another even if we have the same initial population. In terms of experimental results, we run a GA, with fixed parameters of selection, mutation and crossover, many times. For each run of the GA, we measure these quantities at each generation and take the average over all of the runs. The goal of our model is to estimate average values of these variables, hence the average behavior of the GA. In order to simplify the notation, we will use the same symbols $(\kappa_1(t), \kappa_2(t), \{\alpha_i(t)\}_{i=1,..,L})$ and $A_h(t)$ for values of a specific run of a GA, or for an experimental average of these values, or for the estimated theoretical average in our model. Which one is denoted will be clear from the context. We will use the superscripts *c, cs* or *csm* in order to distinguish these variables after crossover, selection or mutation is applied, respectively. So, $\alpha_i^{cs}(t)$ represents the mean of alleles at the i^{th} locus at the t^{th} generation after crossover and selection have been applied, and $\alpha_i^{csm}(t)$ equals $\alpha_i(t+1)$. Note that the crossover operation does not change the mean allele values in OneMax problem. Thus, $\alpha_i(t)$ equals $\alpha_i^c(t)$.

The study of the time evolution of $A_h(t)$'s for several values of *h* gives a very practical insight into the behavior of the GA. Figure 2.5 shows the graphs of $A_h(t)$ for $h = 0, 0.1, \ldots 0.9$, where the crossover rate is 25%, the mutation rate is 0.1%, and fitness-proportional selection is used. The population size is taken as 50 chromosomes and the chromosome length is 100 genes. Each time slice of such a graph can be seen as a "bar graph" of the mean allele distribution at the

given instant. In other words, the vertical distance between two curves gives the number of gene locations at which the mean allele is between the corresponding values, averaged across runs. For example, at t=100, at about 18 gene locations, none of the chromosomes (i.e. h=0) have value 1; at about 5 locations from 1 to 5 chromosomes (i.e., more than 0% and up to 10% of the population, i.e. $0<h\leq 0.1$) have a 1 and the rest have a 0; and at about 50 locations, from 45 to 50 chromosomes (i.e., 91% to 100% of the population, i.e., $0.9<h\leq 1$) have a 1 and the rest have a 0, etc. The closer the curves are to each other, the smaller the variation in the population. We observe that although the population converges to a more-or-less stable configuration after 100 generations, there is still some variation within the population, due to the existence of mutation, which has the potential of creating new chromosomes. Thus, for example, at the right-hand side of the graph, about 18% of the loci are "fixed" at 0, about 50% are "fixed" at 1, and about 32% of the loci have a mixture of 0's and 1's. The number of these "mixed" loci typically increases as the mutation rate increases.

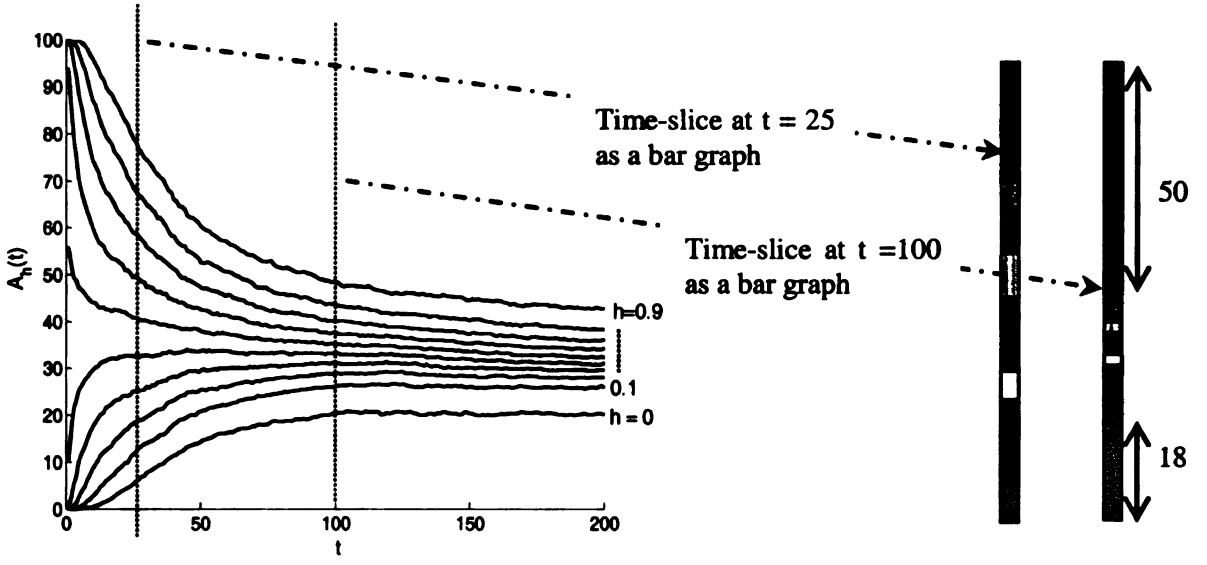


Figure 2.5 The experimental average values of $A_h(t)$ as a function of time for h =0, 0.1, 0.2, ... 0.9 , and the bar graph interpretation. The population size is 50 and the chromosome length is 100 genes. The GA parameters are $p_c = 0.25$, $p_m = 0.001$. The average is taken over 100 experiments

When h is quantized with a gap of 0.1 between two consecutive values as above, we get 10 regions formed between the curves, including the region above the top curve. We will use the index h' to count these regions, $h' = 1, 2, \ldots, 10$, given by

$$R_{h'} = \left\{ (y,t) \,\middle|\, A_{(h'-1)/10}(t) \le y \le A_{h'/10}(t) \right\} \tag{2.5}$$

where $A_1(t)$ is defined as the constant function L.

2.2 The Model for OneMax with Fitness Proportional Selection

The model is developed first for the case with a "high enough" crossover rate. "High enough" here means sufficiently high that the alleles at any locus are distributed essentially randomly among the chromosomes. Then it is modified to include cases with lower crossover rates. The first case involves three main steps. First, mean alleles after crossover and selection are estimated assuming that the crossover rate is "high enough". Then, the effect of mutation on the mean allele is determined. The last step involves the estimation of fitness variance given the mean allele values.

The second case, in which the crossover rate takes more realistic values, is modeled by observing some statistical properties of the GA at early generations. This is an aspect in which this method differs strongly from earlier theoretical models, but which, it is hoped, will allow simple models to be developed that are applicable to a variety of interesting problems, adapting to the behavior of the crossover and fitness functions on a problem-specific basis.

Note that at any GA stage, the mean fitness, κ_1, is always the sum of the mean alleles across loci at that moment, i.e.

$$\kappa_1(t) = \sum_i \alpha_i(t) \tag{2.6}$$

This gives us our first simulation formula:

***M 2.1**: At any stage of the GA for OneMax the mean fitness is*

$$\kappa_1(t) = \sum_i \alpha_i(t)$$

2.2.1 Selection and Crossover with "High Enough" Crossover Rate

First, consider the case in which the crossover rate is so high that the alleles at any locus are distributed essentially randomly among the chromosomes. We will call this crossover rate a "high enough" crossover rate. In fitness-proportional selection, each chromosome has a selection probability proportional to its relative fitness within the population. If we denote as q_j the probability of selecting the j^{th} chromosome, then

$$q_j = \frac{f_j}{\sum_{k=1}^{P} f_k} \tag{2.7}$$

where f_k is the fitness of the k^{th} chromosome.

Let p_i be the probability that a chromosome that is selected randomly with the above probability scheme after the application of crossover, has 1 at its i^{th} locus. As with the other symbols, we will use the notation $p_i(t)$ for values of a specific run of a GA at time t, or of an experimental average of these values at time t, or of the estimated theoretical average in our model, depending on the context. It is easy to estimate $p_i(t)$ theoretically in terms of $\kappa_1(t)$ and $\alpha_i(t)$ when the crossover rate is "high enough". Let $S(t)$ be the population after crossover is

applied with high enough crossover rate to the population of generation $t-1$. Define the subsets $S_0^i(t)$ and $S_1^i(t)$ of $S(t)$ as

$$S_0^i(t) = \{ chrom \in S(t) \mid chrom(i) = 0 \}$$

and

$$S_1^i(t) = \{ chrom \in S(t) \mid chrom(i) = 1 \}. \tag{2.8}$$

Then, we have

$$\overline{\sum_{chrom \in S_0^i(t)} f(chrom)} = P(1-\alpha_i(t))(\kappa_1(t) - \alpha_i(t))$$

and

$$\overline{\sum_{chrom \in S_1^i(t)} f(chrom)} = P\alpha_i(t)(1+\kappa_1(t) - \alpha_i(t)), \tag{2.9}$$

where the bar over the summation means the average over all possible configurations of gene distributions, in which we assume that the genes are distributed randomly satisfying the given mean allele values, since the crossover rate is "high enough". Thus, the estimated average value of $p_i(t)$ is

$$p_i(t) = \frac{P\alpha_i(t)(1+\kappa_1(t)-\alpha_i(t))}{P\alpha_i(t)(1+\kappa_1(t)-\alpha_i(t)) + P(1-\alpha_i(t))(\kappa_1(t)-\alpha_i(t))}, \tag{2.10}$$

which simplifies to

$$p_i(t) = \alpha_i(t) + \frac{(1-\alpha_i(t))\alpha_i(t)}{\kappa_1(t)} \ . \tag{2.11}$$

In the process of fitness proportional selection, we apply selection of chromosomes P times with replacement. Each time, the probability that the selected chromosome has 1 as its i^{th} allele, is $p_i(t)$. So, the expected number of 1's at the i^{th} locus, after the selection is over, can be obtained by using a binomial distribution. Let $B(n,P,p_i)$ denote the probability of having n successes after P trials, when the success probability is p_i for each trial. Then, the expected theoretical value of $\alpha_i^{cs}(t)$ is

$$\alpha_i^{cs}(t) = \frac{1}{P}\sum_{n=1}^{P} n \cdot B\big(n,P,p_i(t)\big) \ , \tag{2.12}$$

when the crossover rate is "high enough".

In summary, we have the following formula that is used in the code simulating the model:

> ***M 2.2**: **(a)** After crossover with high enough rate is applied, the probability that a randomly selected chromosome that is drawn with the probability scheme of fitness proportional selection, has a 1 at its i^{th} locus is* $p_i(t) = \alpha_i(t) + \frac{(1-\alpha_i(t))\alpha_i(t)}{\kappa_1(t)}$.

(b) *The mean allele values after crossover and selection for high enough crossover rates are given by the formula*

$$\alpha_i^{cs}(t) = \frac{1}{P}\sum_{n=1}^{P} n \cdot B\left(n, P, p_i(t)\right) .$$

2.2.2 Mutation

In this section, we want to estimate $\alpha_i^{csm}(t)$ given the values of $\alpha_i^{cs}(t)$. Each gene of a chromosome has the probability p_m of changing its value from 1 to 0 or from 0 to 1 by mutation. When we consider the possible changes at the i^{th} locus only, the expected number, N, of total allele changes due to mutation can be found by using a binomial distribution as

$$N = \sum_{n=1}^{P} n \cdot B\left(n, P, p_m\right) .$$

(2.13)

Since the percentage of 1's at the i^{th} locus is $\alpha_i^{cs}(t)$, $\alpha_i^{cs}(t)N$ of these changes are going to be from 1 to 0, and $(1-\alpha_i^{cs}(t))N$ of the changes are from 0 to 1, on the average. This means that the number of 1's at the i^{th} locus, which is $P\alpha_i^{cs}(t)$, will become $P\alpha_i^{cs}(t) - \alpha_i^{cs}(t)N + (1-\alpha_i^{cs}(t))N$ after the mutation. Simplifying this quantity and dividing by P gives the mean allele for the next generation as

$$\alpha_i(t+1) = \alpha_i^{csm}(t) = \alpha_i^{cs}(t) + \frac{1-2\alpha_i^{cs}(t)}{P} N .$$

(2.14)

As a result we have the following recipe to use in the model simulation:

M 2.3: *The mean allele after mutation is given by* $\alpha_i(t+1) = \alpha_i^{csm}(t) = \alpha_i^{cs}(t) + \frac{1-2\alpha_i^{cs}(t)}{P} N$, *where* $N = \sum_{n=1}^{P} n \cdot B(n, P, p_m)$.

2.2.3 Fitness Variance for "High Enough" Crossover Rates

The fitness variance by definition is

$$\kappa_2(t) = \frac{1}{P}\left(\sum_{k=1}^{P} f(chrom_k)^2\right) - \kappa_1(t)^2 \ . \tag{2.15}$$

If we write the fitness of $chrom_k$ as the sum of its gene values a_i^k and change the order of summation after expanding the square sign above, we obtain

$$\kappa_2(t) = \kappa_1(t) + \frac{1}{P}\sum_{i \neq j}^{L} \sum_{k=1}^{P} a_i^k a_j^k - \kappa_1(t)^2 \ . \tag{2.16}$$

The term $\sum_{k=1}^{P} a_i^k a_j^k$ in Equation (2.16), counts the number of chromosomes in which loci i and j both contain 1's. In the case of "high enough" crossover rates, this count is estimated by using α_i^{cs} and α_j^{cs} as follows. The probability, $p(i, j, n)$, that locations i and j have n common 1's is given by the formula

$$p(i,j,n) = \binom{P\alpha_i^{cs}}{n} \times \binom{P - P\alpha_i^{cs}}{P\alpha_j^{cs} - n} \div \binom{P}{P\alpha_j^{cs}},$$ where n could take any value

between $\max(0, P\alpha_i^{cs} + P\alpha_j^{cs} - P)$ and $\min(P\alpha_i^{cs}, P\alpha_j^{cs})$ and the product of P with α's is rounded to the nearest integer in order to calculate the combinations. Thus, the estimation of the fitness variance in the case of "high enough" crossover rates is found by using

$$\kappa_2^{cs}(t) = \kappa_1^{cs}(t) + \frac{1}{P}\sum_{i\neq j}^{L}\sum_{n} n \cdot p(i,j,n) - \kappa_1^{cs}(t)^2 \quad .$$

(2.17)

The estimation of fitness variance after mutation is done by Prügel-Bennett and Shapiro, [1997]. Their formula gives us

$$\kappa_2^{csm} = (1-2p_m)^2 \kappa_2^{cs} + \left(1-\frac{1}{P}\right) p_m (1-p_m) \sum_{i=1}^{L} w_i^2 \quad ,$$

(2.18)

where w_i is the weight of the i^{th} locus. In other words, the fitness of a chromosome $(a_1, a_2, ..., a_L)$ is calculated by the weighted summation $\sum w_i a_i$. In our special case, the values of w_i's are all 1. So, we use the formula

$$\kappa_2^{csm}(t) = (1-2p_m)^2 \kappa_2^{cs}(t) + \left(1-\frac{1}{P}\right)(p_m - p_m^2) \sum_{i=1}^{L} 1$$

$$= (1-2p_m)^2 \kappa_2^{cs} + L\left(1-\frac{1}{P}\right)(p_m - p_m^2) = \kappa_2(t+1) \quad ,$$

(2.19)

to estimate the fitness variance after mutation.

As a result we have the following two statements to use in the model simulations:

***M 2.4**: The fitness variance after crossover and selection is found by using* $\kappa_2^{cs}(t) = \kappa_1^{cs}(t) + \frac{1}{P}\sum_{i\neq j}^{L}\sum_{n} n \cdot p(i,j,n) - \kappa_1^{cs}(t)^2$, *where*

$p(i,j,n) = \begin{pmatrix} P\alpha_i^{cs} \\ n \end{pmatrix} \times \begin{pmatrix} P - P\alpha_i^{cs} \\ P\alpha_j^{cs} - n \end{pmatrix} \div \begin{pmatrix} P \\ P\alpha_j^{cs} \end{pmatrix}$. *The same formula can be used to estimate* $\kappa_2(0)$ *using* $\kappa_1(0)$ *since the initial values of the allele are chosen randomly which makes it equivalent to a highly mixed population.*

***M 2.5**: The fitness variance after mutation is found by using* $\kappa_2^{csm} = (1-2p_m)^2 \kappa_2^{cs} + L\left(1-\frac{1}{P}\right)\left(p_m - p_m^2\right)$.

2.2.4 Lower Crossover Rates

Equation (2.11) gives the probability p_i when the crossover rate is very high. In such a case, as in Section 2.2.1, we are able to treat the 1's at a fixed locus of different chromosomes as identical to each other in terms of their roles in selection because of the high mixing rate of the crossover operator, which makes chromosomes look similar to each other, on the average. However, for lower and more realistic crossover rates, there will be some correlation between alleles within a chromosome and Equation (2.11) will no longer hold. Let's keep the

usage of notation $p_i(t)$ for the probability of selecting a 1 at the i^{th} locus in the case of the "high enough" crossover rate and denote the corresponding probability in the case of a lower crossover rate by $\tilde{p}_i(t)$. To remedy this situation and estimate $\tilde{p}_i(t)$ correctly, we consider artificial weights, $c_i(t)$, for each locus in order to reflect the average change in the role of 1's played in the selection process due to correlation between alleles. The correction weights, $c_i(t)$, are defined implicitly by

$$\tilde{p}_i(t) = \alpha_i(t) + \frac{(1-\alpha_i(t))\alpha_i(t)c_i(t)}{\kappa_1(t)} . \tag{2.20}$$

The reason why we defined the correction weights as in the equation above is because if we write Equation (2.11) for a fitness function of the form $f(chrom) = \sum_i w_i \cdot chrom(i)$, with weights w_i, we would get an equation exactly like Equation (2.20) with c_i replaced by w_i. Our correction weights play a similar role at each locus as w_i's would, except that c_i's change over time.

The next step will be to estimate the c_i's statistically by means of some data gathered from experiments. In order to do this, the GA is run with fixed rates of p_m and p_c up to a pre-selected generation, say t_0. Let us call this generation G_0. The crossover operation with the current rate, p_c, is applied to G_0 many times. Each time, $\tilde{p}_i(t)$ values are calculated from the experimental data for each locus i, and the corresponding c_i values are found using Equation (2.20).

This process is repeated for many runs of the GA to obtain statistical measures. It is observed that the value of c_i strongly depends on the values of α_i, as expected. Because of this dependence, it makes more sense to group the c_i according to their corresponding α_i values before finding the statistics of the data gathered from the experiments. So, define $C_k^h(t_0)$ as the set $\{ c_i$ values of the k^{th} experiment of GA such that $h \le \alpha_i(t_0) < h + 0.1\}$, for h = 0, 0.1, ...0.9 . The mean of the correction weights is obtained by finding $\mu(h', t_0) = \underset{k}{mean}\left(mean\left(C_k^h(t_0)\right)\right)$, h' = 1, 2, ...,10, where the relationship between the index h and h' is given by h = (h' -1)/10, to be consistent with definition (2.4). In order to measure how much the correction weights vary from one experiment to another, we also calculate the standard deviation $\sigma(h', t_0) = \underset{k}{std}\left(mean\left(C_k^h(t_0)\right)\right)$.

The experimental results show that, when p_c is not too low (below about 4%), μ and σ remain more or less at the same value regardless of the time, t_0. Moreover, μ shows a linear-like behavior while σ shows a quadratic-like behavior as a function of h'. This behavior of the crossover operator allows us to use the linear approximation of $\mu(h',5)$ to predict $\tilde{p}_i(t)$ for the following generations. Figure 2.6 shows the graphs of μ for two different rates of crossover with t_0 at generations 5, 15 and 30. We have observed that the inclusion of μ in our model is good enough for describing the effects of c_i distributions and the information coming from σ does not play a significant role in the counting-ones problem.

However, for other problems, such as OneMax with Boltzmann scaling, σ might be needed in the model. The deviations from the linear behavior, in Figure 2.6, at h'=1 or 10 are due to effects of statistical averaging in which there were not enough data points available for these border values.

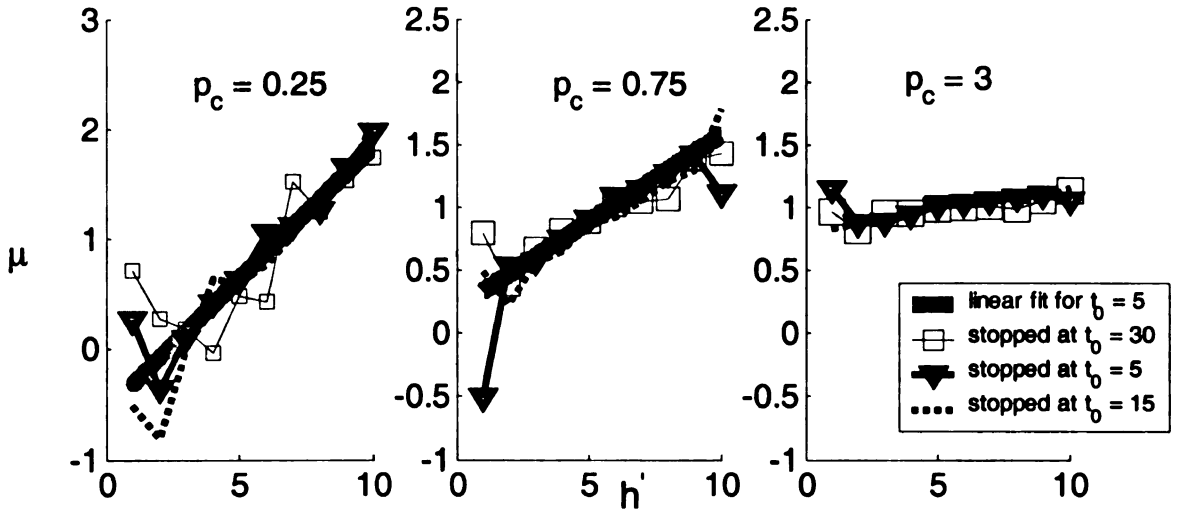


Figure 2.6 The mean value of the correction weights as a function of mean allele levels, h', for crossover rates pc = 0.25, 0.75 and 3. The statistical average is found over 100 runs of the GA. Population size is 50 and chromosome length is 100 genes

***M 2.6**: The mean allele values after crossover and selection for a crossover rate p_c are given by the formula*

$$\alpha_i^{cs}(t) = \frac{1}{P}\sum_{n=1}^{P} n \cdot B\left(n, P, \tilde{p}_i(t)\right), \quad \text{where} \quad \tilde{p}_i(t) = \alpha_i(t) + \frac{\left(1-\alpha_i(t)\right)\alpha_i(t)c_i(t)}{\kappa_1(t)}$$

and the values of $c_i(t)$ are determined as described above for the corresponding crossover rate.

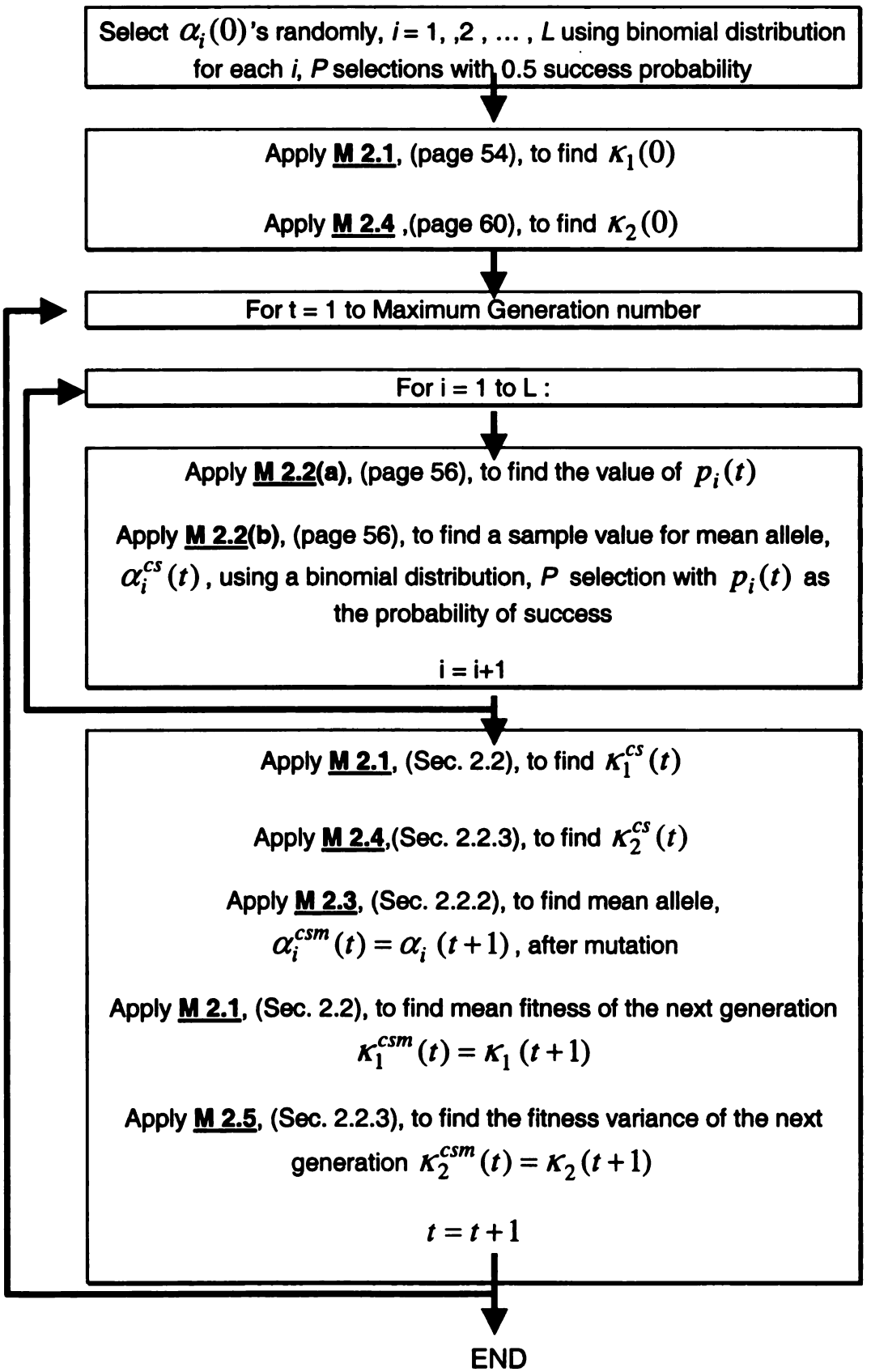
2.2.5 The Algorithm of the Model

Diagram 1, page 65, shows the flowchart of the algorithm that is used in the simulation of the model that has been developed in the previous sections. We apply this algorithm *N* times. The estimations of mean fitness and fitness variance are found by taking the averages over these *N* runs. $\alpha_i(t)$ values of each run are used to find $A_h(t)$ values for that particular run, *h* = 0, 0.1, 0.2, ... 0.9, Taking the average of these measures for each t, in turn, gives us the actual $A_h(t)$ evolutions, that describe average GA behavior.

The simulation of the model for high enough crossover rates starts with selecting a set of $\alpha_i(0)$ values chosen by considering a binomial distribution for each locus in which we have P selections with a 50% chance of selecting a 1 each time. **M 2.2** (page 56) and **M 2.3** (page 58) are applied to estimate the mean alleles after the crossover, selection and mutation operations. This process is iterated for each generation to obtain a dynamic simulation of the mean allele. At any moment, the mean fitness is estimated by **M 2.1** (Section 2.2), and the fitness variance in the case of "high enough" crossover rates is estimated using **M 2.4** or **M 2.5** (Section 2.2.3), depending on whether we are considering the variance right after the selection process or after the mutation, respectively.

In the case of lower crossover rates, **M 2.2** is replaced by **M 2.6**, in which the c_i-values are pre-determined by the linear approximation of the data gathered at the 5th generation of a set of GA runs as described in Section 2.2.4, Figure 2.6.

Diagram 1 The Flowchart of the Simulation Algorithm



2.3 Weighted OneMax Fitness Function

Now, we will consider the case in which each allele contributes to the fitness with different weights. In other words the fitness of a chromosome $chrom_k = (a_1^k, a_2^k, \ldots, a_L^k)$ is

$$f(chrom_k) = \sum_i w_i \cdot a_i^k \,. \tag{2.21}$$

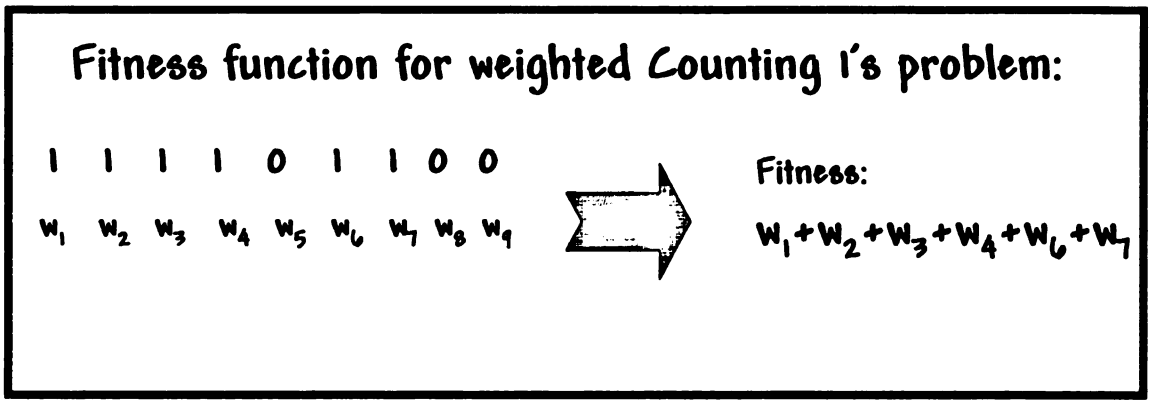


Figure 2.7 Fitness with weights

In Figure 2.7, we see an example of the weighted fitness of a chromosome of length 9.

When we study the GA with the weighted fitness function of Equation (2.21) at time *t*, for the i^{th} locus, Equation (2.9) would change to

$$\overline{\sum_{chrom \in S_0^i(t)} f(chrom)} = P(1-\alpha_i(t))(\kappa_1(t) - w_i\alpha_i(t))$$

and

$$\overline{\sum_{chrom \in S_1^i(t)} f(chrom)} = P\alpha_i(t)(\kappa_1(t) + (1-\alpha_i(t))w_i).$$

(2.22)

Then, Equation (2.11) would be replaced by

$$p_i(t) = \alpha_i(t) + \frac{(1-\alpha_i(t))\alpha_i(t)w_i}{\kappa_1(t)}.$$

(2.23)

This would change our simulation model formula as

M 2.2': ***(a)*** *After crossover with high enough rate is applied, the probability that a randomly selected chromosome that is drawn with the probability scheme of fitness proportional selection, has a 1 at is* i^{th} *locus is* $p_i(t) = \alpha_i(t) + \frac{(1-\alpha_i(t))\alpha_i(t)w_i}{\kappa_1(t)}$.

(b) *The mean allele values after crossover and selection for high enough crossover rates are given by the formula* $\alpha_i^{cs}(t) = \frac{1}{P}\sum_{n=1}^{P} n \cdot B(n, P, p_i(t))$.

The existence of weights changes Equation (2.16) as

$$\kappa_2(t) = \kappa_1(t) + \frac{1}{P}\sum_{i\neq j}^{L}\sum_{k=1}^{P} w_i a_i^k w_j a_j^k - \kappa_1(t)^2 .$$

(2.24)

Then, Equation (2.17) is replaced by

$$\kappa_2^{cs}(t) = \kappa_1^{cs}(t) + \frac{1}{P}\sum_{i\neq j}^{L}\sum_{n} n \cdot w_i w_j p(i,j,n) - \kappa_1^{cs}(t)^2 ,$$

(2.25)

where $p(i,j,n)$ remains the same as

$$p(i,j,n) = \binom{P\alpha_i^{cs}}{n} \times \binom{P - P\alpha_i^{cs}}{P\alpha_j^{cs} - n} \div \binom{P}{P\alpha_j^{cs}} ,$$

(2.26)

as in Section 2.2.3.

The fitness variance after mutation is already given by Equation (2.18) for the weighted case.

The modification for the lower crossover rates is straightforward. The defining equation for the correction weights, Equation (2.20) is now

$$\tilde{p}_i(t) = \alpha_i(t) + \frac{\big(1-\alpha_i(t)\big)\alpha_i(t) w_i c_i(t)}{\kappa_1(t)} .$$

(2.27)

2.4 OneMax with Boltzmann Scaling

One of the most common ways to scale the fitness values so that the chromosomes with higher fitness have more chance to survive the selection operation compared to the usual fitness-proportional selection is Boltzmann scaling. In this scaling, the fitness value $f(chrom_i)$ of the i[th] chromosome is replaced by

$$\tilde{f}(chrom_i) = e^{\beta f(chrom_i)}, \tag{2.28}$$

where β, which controls the pressure of the selection, might be a function of some parameters of the population or it might be a constant. Then, the probability, q_i that the i[th] chromosome would be selected by the Boltzmann-scaled fitness proportional selection is given by the quotient

$$q_i = \frac{e^{\beta f(chrom_i)}}{\sum_{j=1}^{P} e^{\beta f(chrom_j)}}. \tag{2.29}$$

The advantage of using a Boltzmann scaling is that the value of q_i is shift invariant. In other words, if the fitness values of all chromosomes present in the population at any time t are shifted by a value *c*, then q_i would remain unchanged for $chrom_i$. One can see this property easily from

$$q_i' = \frac{e^{\beta(f(chrom_i)+c)}}{\sum_{j=1}^{P} e^{\beta(f(chrom_i)+c)}}$$

$$= \frac{e^{\beta c} e^{\beta f(chrom_i)}}{e^{\beta c} \sum_{j=1}^{P} e^{\beta f(chrom_i)}}$$

$$= \frac{e^{\beta f(chrom_i)}}{\sum_{j=1}^{P} e^{\beta f(chrom_i)}} = q_i$$

In the case of scaled β, the selection pressure, β, is scaled at each generation inversely proportional to the standard deviation, $\sigma = \sqrt{\kappa_2}$ of the fitness. So, $\beta_S = \frac{\sqrt{2\ln(P)}}{\sigma}\beta$ is used instead of β and the fitness values, $f(chrom)$, are replaced by

$$\tilde{f}(chrom) = e^{\frac{\sqrt{2\ln(P)}}{\sigma}\beta f(chrom)}. \tag{2.30}$$

Then q_i is

$$q_i = \frac{\tilde{f}(chrom_i)}{\sum_{j=1}^{P} \tilde{f}(chrom_j)} = \frac{e^{\frac{\sqrt{2\ln(P)}}{\sigma}\beta f(chrom_i)}}{\sum_{j=1}^{P} e^{\frac{\sqrt{2\ln(P)}}{\sigma}\beta f(chrom_j)}}.$$

When β is scaled like this, then the value of $\tilde{f}(chrom_j)$ is invariant under multiplication of fitness values by a constant. This is because when all the fitness values in a population are multiplied by *c*, then the new variance of the population would be $c^2\kappa_2$, where κ_2 is the variance of the original population. Then,

$$\tilde{f}(chrom)' = e^{\frac{\sqrt{2\ln(P)}}{\sqrt{c^2\kappa_2}}\beta cf(chrom)} = e^{\frac{\sqrt{2\ln(P)}}{c\sqrt{\kappa_2}}\beta cf(chrom)} = \tilde{f}(chrom).$$

Hence, using a Boltzmann selection with scaled β, the selection operation would be invariant under constant multiples or shifts of fitness values.

We will study two types of Boltzmann selection for the high enough crossover rates only. In the first one, β is kept constant throughout the whole GA run. We call this case the 'Boltzmann selection with fixed β'. In the second one, β changes inversely proportional to the standard deviation of the fitness distribution of the current population. This case will be called 'Boltzmann selection with scaled β.'

Using the same notation as in Section 2.2.1, we define the sets

$$S_0^i(t) = \{ chrom \in S(t) \mid chrom(i) = 0 \}$$

and

$$S_1^i(t) = \{ chrom \in S(t) \mid chrom(i) = 1 \},$$

where $S(t)$ is the set of all chromosomes at time t.

Let us consider an instance of a GA run and study this instance for the i[th] locus. We define probability p_i, similar to that in Section 2.2.1, as the probability that a chromosome that is selected randomly with the Boltzmann probability scheme after the application of crossover has 1 at its i[th] locus. Then, we have

$$p_i(t) = \frac{\sum_{chrom \in S_1^i(t)} e^{\beta f(chrom)}}{\sum_{chrom \in S_1^i(t)} e^{\beta f(chrom)} + \sum_{chrom \in S_0^i(t)} e^{\beta f(chrom)}} . \tag{2.31}$$

We want to find the expected value of the probability, $p_i(t)$, when the crossover rate is 'high enough'. If the i[th] locus values of all chromosomes in this population are replaced by 1's then we would have a fitness distribution whose mean fitness is $\kappa_1 + (1 - \alpha_i)$, where κ_1 is the mean fitness of the original population. At this point, we assume that the variance of this new population is almost the same as the variance of the original population. Further, assuming that all the fitness distributions under consideration are normal distributions, we propose a model distribution for the set $S_1^i(t)$:

***M 2.7**: Fitness values of $S_1^i(t)$ are assumed to have a normal distribution with mean $\kappa_1 + (1 - \alpha_i)$ and variance κ_2.*

Similarly, for the fitness values of $S_0^i(t)$ we have:

***M 2.8**: Fitness values of $S_0^i(t)$ are assumed to have a normal distribution with mean $\kappa_1 - \alpha_i$ and variance κ_2.*

Chapter 3

A MODEL OF GA DYNAMICS FOR THE DECEPTIVE FUNCTION PROBLEM

GAs, with the help of the selection operator, have a tendency to treat the direction in which an increase in fitness is observed when a gene value is changed as the direction to go to seek the optimum value (essentially, hill climbing). However, not all such changes, in general, lead toward a gene value that is part of the optimum chromosome. In particular, there are some test functions that are especially designed to mislead GAs that take advantage of this weakness of the algorithm. An important class of such functions is called "deceptive functions". In this chapter, we will describe a form of deceptive function and develop a model for the GA with this function.

3.1 Deceptive Function

First, we define an N-bit deceptive function. For this function, the chromosome is made up of some number of N-bit blocks and the fitness of the chromosome is found by adding up the fitness contribution of each block. The fitness contribution of an N-bit block is given by

$$f(a_1, a_2, \ldots, a_N) = \begin{cases} A & \text{if } a_i = 1 \text{ for all } i \\ Bm & \text{if some } a_i\text{'s are 0, where } m = \#\text{ of zeros} \end{cases} \tag{3.1}$$

The chromosome consists of many such N-bit blocks and its fitness is the sum of each of their contributions given by Equation (3.1). The constants *A* and *B* are such that *A>BN*. In other words, a deceptive block with the highest fitness contribution would have all 1's as its alleles.

Figure 3.1 shows an example of a 3-bit deceptive fitness function. In this example the fitness of (111 101 100) is A+1B+2B = A+3B. We see that the fitness of the chromosome would increase if the last deceptive block, 100, would change to 000. However, it is clear that the optimum chromosome is the one with all 1's. This is the reason why these functions are called deceptive: From the GA-selection operator's point of view 111 101 000 is better than 111 101 100, while the later is bitwise closer to the optimum.

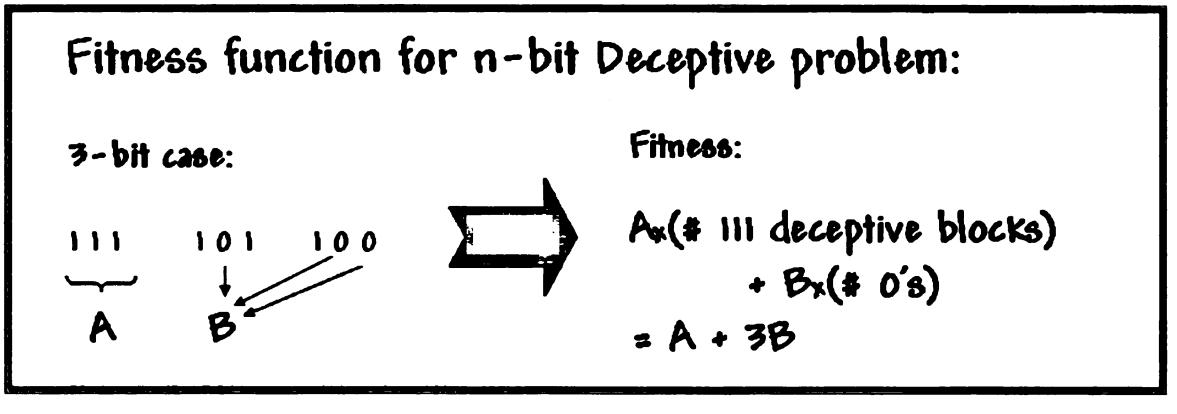


Figure 3.1 The definition of the fitness of a 3-bit deceptive function

3.2 A Model for the Deceptive Function with Fitness-Proportional Selection

When a GA is applied to find the optimum value of the deceptive function defined in the previous section, it tries to increase the number of both 0's and 1's at the same time. So, the problem involves a counting-0's and a counting-all-1-deceptive blocks (i.e., the N-bit blocks with all their allele values equal to 1) problem, competing with each other. Both the counting-0's and counting-all-1-deceptive blocks work similarly to the OneMax problem modeled in Chapter 2. In this section, we apply this idea and develop a model for the deceptive function, using the results of the OneMax model.

We use the index letter *j* to index the deceptive blocks, and the index letter *i* to index the locus in the chromosome. Similarly to the mean allele values defined in Section 2.1, Figure 2.2, we define

$$\alpha_j = \frac{\#\text{ of 0's at the } j^{\text{th}} \text{ locus of all chromosomes}}{P}$$

and

$$\gamma_i = \frac{\#\text{ of all-1 deceptive blocks at the } i^{\text{th}} \text{ N-bit deceptive loci in the population}}{P} \tag{3.2}$$

where P is the population size. Figure 3.2 shows the definitions of α and γ for a 3-bit deceptive function.

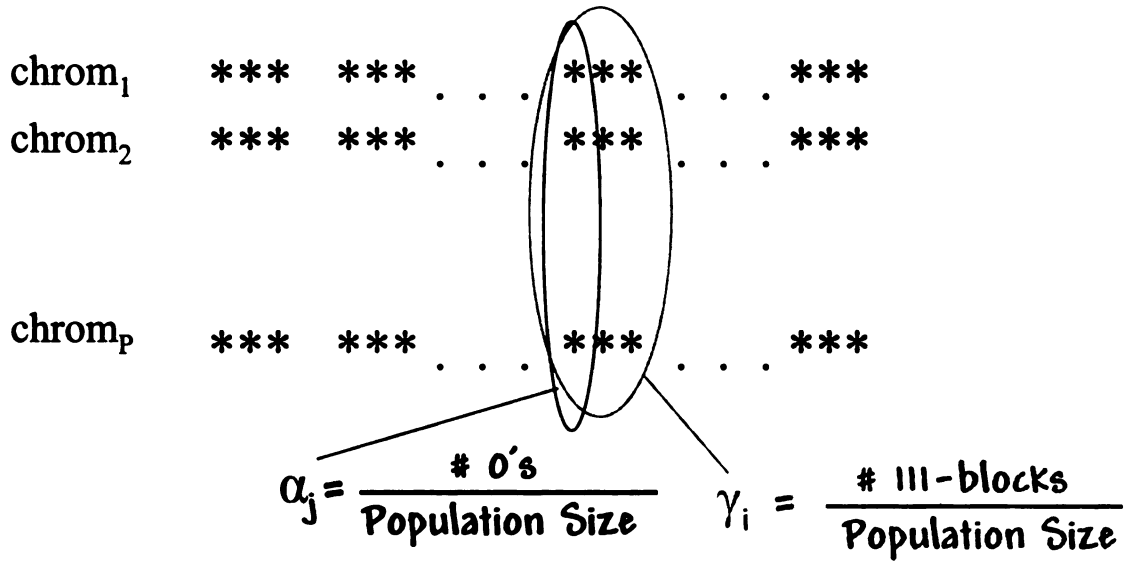


Figure 3.2 Variables α and γ in a 3-bit deceptive function problem

Similarly to Equation (2.4), Figure 2.4, define

$$A_h^{\alpha}(t) = \#\left\{\alpha_j(t) \,\middle|\, \alpha_j(t) \le h,\ j = 1,\ldots,L\right\}$$

and

$$A_h^{\gamma}(t) = \#\left\{\gamma_i(t) \,\middle|\, \gamma_i(t) \le h,\ i = 1,\ldots,\frac{L}{N}\right\}, \tag{3.3}$$

where L is an integer multiple of N.

In the following subsections, we develop a model to estimate the average expected values of $\alpha_j(t)$ and $\gamma_i(t)$ from their previous values.

3.2.1 Selection and Crossover with "High Enough" Crossover Rate

Similarly to Section 2.2.1, we first consider the case in which the crossover rate is so high that the alleles at any locus are distributed essentially randomly among the chromosomes. We will call this crossover rate a "high enough" crossover rate.

We have the following simple equation, following directly from the definition of the mean fitness

M 3.1 *At any stage of the model the mean fitness is given by*

$$\kappa_1 = A\sum_{i}^{L/3}\gamma_i + B\sum_{j=1}^{L}\alpha_j \ .$$

Define $p_j^{\alpha}(t)$ as the probability at generation *t* that the j^{th} locus of a chromosome that is selected randomly with the fitness proportional selection scheme after the application of crossover, has 1 as its gene value. $p_i^{\gamma}(t)$ is defined as the probability at generation *t* that the i^{th} N-bit deceptive block of a chromosome that is selected randomly with the fitness proportional selection scheme after the application of crossover, has all 1's as its alleles. In Figure 3.3, we see a pictorial definition of $p_j^{\alpha}(t)$ and $p_i^{\gamma}(t)$ for a 3-bit deceptive function.

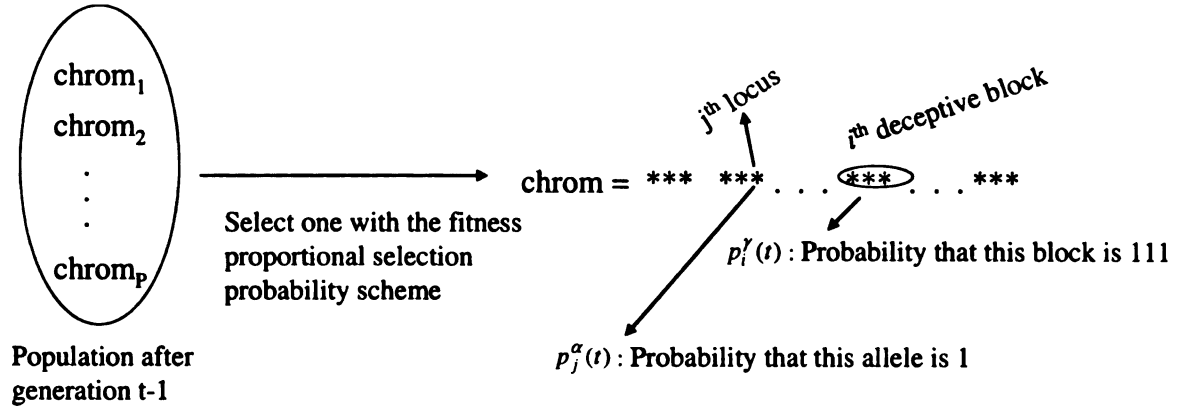


Figure 3.3 Definitions of $p_j^{\alpha}(t)$ and $p_i^{\gamma}(t)$ for a 3-bit deceptive function

Let $S(t)$ be the population after crossover is applied with a high enough crossover rate to the population of generation $t-1$. Define the subsets ${}^{\gamma}S_0^i(t)$, ${}^{\gamma}S_1^i(t)$, ${}^{\alpha}S_0^j(t)$ and ${}^{\alpha}S_1^j(t)$ of $S(t)$ as

$$ {}^{\gamma}S_1^i(t) = \left\{ chrom \in S(t) \,\middle|\, i^{th} \text{ deceptive block of } chrom \text{ is } 11...1 \right\}, $$

$$ {}^{\gamma}S_0^i(t) = S(t) - {}^{\gamma}S_1^i(t), $$

$$ {}^{\alpha}S_1^j(t) = \left\{ chrom \in S(t) \,\middle|\, j^{th} \text{ locus of } chrom \text{ is } 1 \right\}, $$

$$ {}^{\alpha}S_0^j(t) = S(t) - {}^{\alpha}S_1^j(t). $$

(3.4)

Then, by definition of $p_j^{\alpha}(t)$ and $p_i^{\gamma}(t)$ we have

$$p_j^{\alpha}(t) = \frac{\sum_{chrom \in {}^{\alpha}S_0^j(t)} f(chrom)}{\sum_{chrom \in {}^{\alpha}S_0^j(t)} f(chrom) + \sum_{chrom \in {}^{\alpha}S_1^j(t)} f(chrom)} = \frac{\sum_{chrom \in {}^{\alpha}S_0^j(t)} f(chrom)}{\kappa_1^c}$$

and

$$p_i^{\gamma}(t) = \frac{\sum_{chrom \in {}^{\gamma}S_1^i(t)} f(chrom)}{\sum_{chrom \in {}^{\gamma}S_1^i(t)} f(chrom) + \sum_{chrom \in {}^{\gamma}S_0^i(t)} f(chrom)} = \frac{\sum_{chrom \in {}^{\gamma}S_1^i(t)} f(chrom)}{\kappa_1^c} \quad . \tag{3.5}$$

In order to simplify notation, from now on we assume that the deceptive function is 3-bit. All the arguments below can easily be generalized to a deceptive function with any number of bits.

Then, for the set ${}^{\gamma}S_1^i(t)$ we have

$$\overline{\sum_{chrom \in {}^{\gamma}S_1^i(t)} f(chrom)} = P\left[A + \left(\kappa_1^c(t) - A\gamma_i^c(t) - B\left(\alpha_{j+1}(t) + \alpha_{j+2}(t) + \alpha_{j+3}(t)\right)\right)\right], \tag{3.6}$$

where the values of *i* and *j* are related to each other by $j = N(i-1)$, ($N = 3$ in our case).

For the set ${}^{\alpha}S_1^j(t)$, we have

$$\overline{\sum_{chrom \in {}^{\alpha}S_0^j(t)} f(chrom)} = P\left[B + \left(\kappa_1^c(t) - A\gamma_i^c(t) - B\alpha_j(t)\right)\right], \tag{3.7}$$

where the values of *i* and *j* are related to each other by $i = \text{int}(j/N)+1$, ("int" is the function giving the integer value of its argument and $N = 3$ in our case).

In both Equations (3.6) and (3.7), the bar over the summation means the average over all possible configurations of gene distributions, in which we assume that the genes are distributed randomly satisfying the given mean allele values, since the crossover rate is "high enough". The superscript "*c*" in both equations means the values of the corresponding variables after crossover is applied. Note that, in the case of the deceptive function, the crossover operation changes the value of mean fitness, $\kappa_1(t)$, as well as $\gamma_i(t)$, while $\alpha_j(t)$ remains unchanged. In the OneMax problem, the mean fitness was not changed by crossover.

Equations (3.6) and (3.7) give us the probabilities $p_j^{\alpha}(t)$ and $p_i^{\gamma}(t)$ as

$$p_i^{\gamma} = \frac{\gamma_i^c\left(A + \left(\kappa_1^c - A\gamma_i^c - B(\alpha_{j+1} + \alpha_{j+2} + \alpha_{j+3})\right)\right)}{\kappa_1^c}, \text{ where } j = 3(i-1) \tag{3.8}$$

and

$$p_j^{\alpha} = \frac{\alpha_j\left(B + (\kappa_1^c - A\gamma_i^c - B\alpha_j)\right)}{\kappa_1^c}, \text{ where } i = \text{int}(j/3)+1. \tag{3.9}$$

In order to use equations (3.8) and (3.9), we need to estimate the values of κ_1^c and γ_i^c – i.e., the mean fitness and deceptive block percentages at each

deceptive locus after crossover is applied. For high enough crossover rates, they are easy to estimate. γ_i^c is determined by

$$\gamma_i^c = \frac{\hat{G}\left(P(1-\alpha_{j+1}), P(1-\alpha_{j+2}), P(1-\alpha_{j+3})\right)}{P}, \tag{3.10}$$

where $\hat{G}(n_1,n_2,n_3)$ is the function that gives the expected number of 111's in the population when the genes are completely shuffled with n_k being the number of 1's at the corresponding locus of all the chromosomes. Then, κ_1^c is given by

$$\kappa_1^c = A\sum_{i}^{L/3}\gamma_i^c + B\sum_{j=1}^{L}\alpha_j . \tag{3.11}$$

In the process of fitness-proportional selection, we apply selection of chromosomes P times with replacement. Each time, the probability that the selected chromosome has 1 as its j^{th} allele, is p_j^{α} and the probability that the i^{th} deceptive block is 111 is p_i^{γ}. So, we use a binomial distribution to find the expected number of 1's at the j^{th} locus and expected number of all-1-deceptive blocks at the i^{th} deceptive place. We use the notation $B(n,P,r)$ to denote the probability of having n successes after P trials, when the success probability is r for each trial. Then, the expected theoretical value of $\alpha_j^{cs}(t)$ is

$$\alpha_j^{cs}(t) = \frac{1}{P}\sum_{n=1}^{P} n \cdot B\left(n, P, p_j^{\alpha}(t)\right),$$

(3.12)

when the crossover rate is "high enough". Similarly, the expected theoretical value of $\gamma_i^{cs}(t)$ is

$$\gamma_i^{cs}(t) = \frac{1}{P}\sum_{n=1}^{P} n \cdot B\left(n, P, p_i^{\gamma}(t)\right).$$

(3.13)

Then, in the model simulation, we have

> ***M 3.2**: The values of $\alpha_j^{cs}(t)$ after crossover and selection for high enough crossover rates are given by Equation (3.12), where $p_j^{\alpha}(t)$ is given by (3.9).*

Having found the values of $\alpha_{j+1}^{cs}(t)$, $\alpha_{j+2}^{cs}(t)$ and $\alpha_{j+3}^{cs}(t)$ in the simulation, we need to modify Equation (3.13), in order to find $\gamma_i^{cs}(t)$, where *j =3(i-1),* as follows. Since the values of $\alpha_{j+1}^{cs}(t)$, $\alpha_{j+2}^{cs}(t)$ and $\alpha_{j+3}^{cs}(t)$ are already determined, this means that we have only

$$\tilde{P} = \min\left(P(1-\alpha_{j+1}^{cs}(t)),\quad P(1-\alpha_{j+2}^{cs}(t)),\quad P(1-\alpha_{j+3}^{cs}(t))\right),$$

(3.14)

chromosomes left in the population which have the possibility of having 111-deceptive blocks at their i^{th} deceptive places. Moreover, the probability $p_i^{\gamma}(t)$ would be modified as

$$\tilde{p}_i^{\gamma}(t) = \frac{p_i^{\gamma}}{1 - p_k^{\alpha}}, \tag{3.15}$$

where k is the index of α for which we get the minimum value in $\left\{P(1-\alpha_{j+1}^{cs}(t)),\ P(1-\alpha_{j+2}^{cs}(t)),\ P(1-\alpha_{j+3}^{cs}(t))\right\}$. Thus, the expected theoretical value of $\gamma_i^{cs}(t)$ is now given by

$$\gamma_i^{cs}(t) = \frac{1}{P}\sum_{n=1}^{P} n \cdot B\left(n, \tilde{P}, \tilde{p}_i^{\gamma}(t)\right). \tag{3.16}$$

As a result, we have

> ***M 3.3*** *After $\alpha_{j+1}^{cs}(t)$, $\alpha_{j+2}^{cs}(t)$ and $\alpha_{j+3}^{cs}(t)$ are determined by M 3.2, the values of $\gamma_i^{cs}(t)$ are given by Equation (3.16), where $\tilde{P}$ and $\tilde{p}_i^{\gamma}(t)$ are determined by Equations (3.14) and (3.15), respectively.*

3.2.2 Mutation

The modeling of mutation is done exactly as in Section 2.2.2.

Thus we have

M 3.4 *The mean allele after mutation is given by*

$$\alpha_j(t+1) = \alpha_j^{csm}(t) = \alpha_j^{cs}(t) + \frac{1-2\alpha_j^{cs}(t)}{P} N, \qquad \textit{where}$$

$$N = \sum_{n=1}^{P} n \cdot B(n, P, p_m).$$

After the values of $\alpha_j(t+1)$ are determined as above, the values of $\gamma_i(t+1)$ are estimated from them by using the function $\hat{G}$, which is defined in Section 3.2.1,

$$\gamma_i(t+1) = \frac{\hat{G}\left(P(1-\alpha_{j+1}), P(1-\alpha_{j+2}), P(1-\alpha_{j+3})\right)}{P}. \tag{3.17}$$

3.2.3 Fitness Variance

Fitness variance by definition is

$$\kappa_2(t) = \frac{1}{P}\left(\sum_{k=1}^{P} f(chrom_k)^2\right) - \kappa_1(t)^2 . \tag{3.18}$$

In order to write the fitness of a chromosome more easily in the calculations, define variables g_i and a_l as

$$g_i = \begin{cases} 1 \text{ if the } i^{th} \text{ deceptive block is } 111 \\ 0 \text{ otherwise} \end{cases}$$

$$a_l = \begin{cases} 1 \text{ if the } l^{th} \text{ allele is 0} \\ 0 \text{ otherwise} \end{cases}$$

Then the fitness of a chromosome is

$$f(chrom_k) = \sum_{i}^{L/3} \left(Ag_i^k + B(a_{3(i-1)+1}^k + a_{3(i-1)+2}^k + a_{3(i-1)+3}^k) \right). \tag{3.19}$$

Then,

$$\frac{1}{P}\left(\sum_{k=1}^{P} f(chrom_k)^2 \right) = \frac{1}{P} \sum_{j,i}^{L/3} \sum_{k} \left(Ag_j^k + B(a_{3(j-1)+1}^k + a_{3(j-1)+2}^k + a_{3(j-1)+3}^k) \right) * \left(Ag_i^k + B(a_{3(i-1)+1}^k + a_{3(i-1)+2}^k + a_{3(i-1)+3}^k) \right) \tag{3.20}$$

Separating the terms with *i=j* from the rest of the summation yields an estimation of the above expression, given by

$$\frac{1}{P}\left(\sum_{k=1}^{P} f(chrom_k)^2\right) \approx \sum_{i=1}^{L/3}\Big[A^2\gamma_i + (3B)^2\xi_i$$
$$+ (2B)^2\alpha_{3(i-1)+1}\cdot\alpha_{3(i-1)+2}\cdot(1-\alpha_{3(i-1)+3})$$
$$+ (2B)^2\alpha_{3(i-1)+1}\cdot(1-\alpha_{3(i-1)+2})\cdot\alpha_{3(i-1)+3}$$
$$+ (2B)^2(1-\alpha_{3(i-1)+1})\cdot\alpha_{3(i-1)+2}\cdot\alpha_{3(i-1)+3}$$
$$+ B^2\alpha_{3(i-1)+1}\cdot(1-\alpha_{3(i-1)+2})\cdot(1-\alpha_{3(i-1)+3})$$
$$+ B^2(1-\alpha_{3(i-1)+1})\cdot\alpha_{3(i-1)+2}\cdot(1-\alpha_{3(i-1)+3})$$
$$+ B^2(1-\alpha_{3(i-1)+1})\cdot(1-\alpha_{3(i-1)+2})\cdot\alpha_{3(i-1)+3}\Big]$$
$$+ \sum_{i\neq j}^{L/3}\Big[\Big(B\big(\alpha_{3(i-1)+1}+\alpha_{3(i-1)+2}+\alpha_{3(i-1)+3}\big)+A\gamma_i\Big)*$$
$$\Big(B\big(\alpha_{3(j-1)+1}+\alpha_{3(j-1)+2}+\alpha_{3(j-1)+3}\big)+A\gamma_j\Big)\Big] \quad (3.21)$$

In the estimation above, ξ_i is the estimated percentage of deceptive blocks which are 000 at the i^{th} deceptive place.

As a result, we have

M 3.5 *The estimation of fitness variance for high enough crossover rates is given by Equation (3.18), where the first term is estimated using Equation (3.21).*

3.2.4 The Algorithm of the Model

Diagram 2, page 89, shows the flowchart of the algorithm that is used in the simulation of the model developed in the previous sections. We apply this algorithm several times (~10 times). The estimations of mean fitness and fitness variance are found by taking their averages over these runs. Then, $\alpha_j(t)$ and $\gamma_i(t)$ values of each run are used to find $A_h^{\alpha}(t)$ and $A_h^{\gamma}(t)$ values, h = 0, 0.1, 0.2, ... 0.9, whose averages, in turn, give us the average $A_h(t)$ evolutions.

Diagram 2 The Flowchart of the Simulation Algorithm

Select $\alpha_j(0)$'s randomly, j = 1, ,2 , ... , L using binomial distribution for each i, P selections with 0.5 success probability

Estimate $\gamma_i(0)$ by Equation (3.10), i=1, ... , L/3

Apply **M 3.1**, (page 78), to find $\kappa_1(0)$

Apply **M 3.5**,(page 87), to find $\kappa_2(0)$

For t = 1 to maximum generation number do the following

For each n = 1 to L/3 :

Estimate $\gamma_n^c(t)$ by Equation (3.10), n=1, ... , L/3

Apply **M 3.1**, (page 78), to find $\kappa_1^c(t)$

For each i = 1 to L/3 :

Apply **M 3.2**, (page 83), to find the values of $\alpha_j^{cs}(t)$,
for $j = \{3(i-1)+1, \quad 3(i-1)+2, \quad 3(i-1)+3\}$

Apply **M 3.3**, (page 84), to find to find the value of $\gamma_i^{cs}(t)$,

Apply **M 3.1**, (page 78), to find $\kappa_1^{cs}(t)$

Apply **M 3.5**,(page 87), to find $\kappa_2^{cs}(t)$

Apply **M 3.4**, (page 85), to find mean allele, $\alpha_i^{csm}(t) = \alpha_i(t+1)$, after mutation

Apply **M 3.3**, (page 84), to find to find the value of $\gamma_i^{csm}(t)$,

Apply **M 3.1**, (page 78), to find mean fitness of the next generation $\kappa_1^{csm}(t) = \kappa_1(t+1)$. Apply **M 3.5**,(page 87), to find the fitness variance of the next generation $\kappa_2^{csm}(t) = \kappa_2(t+1)$

3.3 Weighted Deceptive Fitness Function

We define the weighted fitness function for an N-bit deceptive problem by assigning weights both for each locus and for each N-bit deceptive block. Let's use the notation w_j for the weight of j^{th} locus, and v_i for the weight of the i^{th} deceptive N-bit block. Define variables g_i and a_l as

$$g_i = \begin{cases} 1 \text{ if the } i^{th} \text{ deceptive block is all 1's} \\ 0 \text{ otherwise} \end{cases}$$

$$a_l = \begin{cases} 1 \text{ if the } l^{th} \text{ allele is 0} \\ 0 \text{ otherwise} \end{cases}$$

Then, the fitness of a chromosome for a 3-bit case, for example, would be given by

$$f(chrom_k) = \sum_{i}^{L/3} \left[A w_i g_i^k + B\left(v_{3(i-1)+1} a_{3(i-1)+1}^k + v_{3(i-1)+2} a_{3(i-1)+2}^k + v_{3(i-1)+3} a_{3(i-1)+3}^k \right) \right]. \tag{3.22}$$

Figure 3.4 shows an example fitness calculations for the chromosome 111101100, with a 3-bit weighted deceptive function.

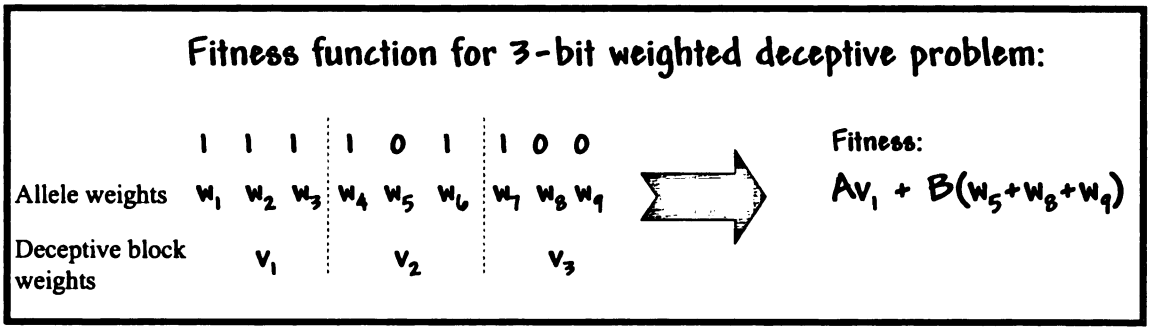


Figure 3.4 An example of the weighted fitness function for a 3-bit deceptive problem

For the mean fitness, in this case, we have

M 3.1' *At any stage of the model the mean fitness is given by*

$$\kappa_1 = A\sum_{i}^{L/3} v_i\gamma_i + B\sum_{j=1}^{L} w_j\alpha_j .$$

The Equations (3.6) and (3.7) would be changed as

$$\frac{\sum f(chrom)}{chrom \in {}^{\gamma}S_1^i(t)}$$

$$= P\left[Av_i + \kappa_1^c(t) - Av_i\gamma_i^c(t) - B\left(w_{j+1}\alpha_{j+1}(t) + w_{j+2}\alpha_{j+2}(t) + w_{j+3}\alpha_{j+3}(t)\right)\right]$$

$$\text{where } j = 3(i-1), \tag{3.23}$$

and

$$\overline{\sum_{chrom \in {}^{\alpha}S_0^j(t)} f(chrom)} = P\left[Bw_j + \left(\kappa_1^c(t) - Av_i\gamma_i^c(t) - Bw_j\alpha_j(t)\right)\right], \tag{3.24}$$

where $i = \text{int}(j/N) + 1$.

These modifications give us the probabilities $p_j^{\alpha}(t)$ and $p_i^{\gamma}(t)$ as

$$p_i^{\gamma} = \frac{\gamma_i^c\left[Av_i + \kappa_1^c(t) - Av_i\gamma_i^c(t) - B\left(w_{j+1}\alpha_{j+1}(t) + w_{j+2}\alpha_{j+2}(t) + w_{j+3}\alpha_{j+3}(t)\right)\right]}{\kappa_1^c} \tag{3.25}$$

where $j = 3(i-1)$, and

$$p_j^{\alpha} = \frac{\alpha_j\left[Bw_j + \left(\kappa_1^c(t) - Av_i\gamma_i^c(t) - Bw_j\alpha_j(t)\right)\right]}{\kappa_1^c}, \tag{3.26}$$

where $i = \text{int}(j/3) + 1$.

3.4 Deceptive Function with Boltzmann Scaling

In Section 2.4, we have defined the Boltzmann scaling for two cases, one with fixed β and the other with β adjusted inversely proportional to the standard deviation of the fitness distribution. The development of the model of the deceptive function with Boltzmann scaling is similar to Section 2.4. Using the

definitions of $p_j^{\alpha}(t)$ and $p_i^{\gamma}(t)$ and the sets ${}^{\gamma}S_0^i(t)$, ${}^{\gamma}S_1^i(t)$, ${}^{\alpha}S_0^j(t)$ and ${}^{\alpha}S_1^j(t)$, in Section 3.2.1, the equation corresponding to (3.5) would be

$$p_j^{\alpha}(t) = \frac{\sum_{chrom \in {}^{\alpha}S_0^j(t)} e^{\beta f(chrom)}}{\sum_{chrom \in {}^{\alpha}S_0^j(t)} e^{\beta f(chrom)} + \sum_{chrom \in {}^{\alpha}S_1^j(t)} e^{\beta f(chrom)}}$$

and

$$p_i^{\gamma}(t) = \frac{\sum_{chrom \in {}^{\gamma}S_1^i(t)} e^{\beta f(chrom)}}{\sum_{chrom \in {}^{\gamma}S_1^i(t)} e^{\beta f(chrom)} + \sum_{chrom \in {}^{\gamma}S_0^i(t)} e^{\beta f(chrom)}} . \tag{3.27}$$

In the model, we need to estimate expected values of $\sum_{chrom \in {}^{\alpha}S_1^j(t)} e^{\beta f(chrom)}$, $\sum_{chrom \in {}^{\alpha}S_0^j(t)} e^{\beta f(chrom)}$, $\sum_{chrom \in {}^{\gamma}S_1^i(t)} e^{\beta f(chrom)}$ and $\sum_{chrom \in {}^{\gamma}S_0^i(t)} e^{\beta f(chrom)}$. When the crossover rate is high enough, we have for the γ counting

$$\overline{\sum_{chrom \in {}^{\gamma}S_1^i(t)} f(chrom)} = P\left[A + \left(\kappa_1^c(t) - A\gamma_i^c(t) - B\left(\alpha_j(t) + \alpha_{j+1}(t) + \alpha_{j+2}(t)\right)\right)\right]$$

$$\overline{\sum_{chrom \in {}^{\gamma}S_0^i(t)} f(chrom)} = P\left(\kappa_1^c(t) - A\gamma_i^c(t) + B\frac{\left(\alpha_j(t) + \alpha_{j+1}(t) + \alpha_{j+2}(t)\right)}{1 - \gamma_i^c(t)}\right), \tag{3.28}$$

and for the α counting

$$\overline{\sum_{chrom \in {}^{\alpha}S_1^j(t)} f(chrom)} =$$

$$P\left[\kappa_1^c(t) - B\alpha_j(t) + A\left(Common\left(1-\alpha_{j+1}(t), 1-\alpha_{j+2}(t)\right) - \gamma_i^c(t)\right)\right]$$

$$\overline{\sum_{chrom \in {}^{\alpha}S_0^j(t)} f(chrom)} = P\left[B + \left(\kappa_1^c(t) - A\gamma_i^c(t) - B\alpha_j(t)\right)\right].$$

(3.29)

Then, for the model simulation we have the assumptions

M 3.6 *The fitness distributions of the sets ${}^{\gamma}S_0^i(t)$ and ${}^{\gamma}S_1^i(t)$ are normal with the mean values given by Equation (3.28).*

M 3.7 *The fitness distributions of the sets ${}^{\alpha}S_0^j(t)$ and ${}^{\alpha}S_1^j(t)$ are normal with the mean values given by Equation (3.29).*

Chapter 4

COMPARISON OF THE MODEL WITH GA EXPERIMENTS

In this chapter, we explore the model for OneMax and deceptive functions developed in Chapter 2 and 3, with various sets of GA parameters, and compare the model predictions with the data obtained from GA runs. This way, we both examine the GA behavior under different circumstances and also test the validity of the model assumptions. The exploration of the OneMax function includes both "high enough" and lower crossover rates, while for the deceptive function, we have only the results for "high enough" crossover rates since the model for lower rates is a subject for future research. In all the graphs of this chapter comparing the model with the actual GA runs, the black lines represent the experimental GA results and the thick gray lines represent the results obtained from our model. For notation purposes only, in some of the graphs, we denote the fitness proportional selection without the Boltzmann scaling as $\beta = -1$, although it is clear that there is no such β variable in this case.

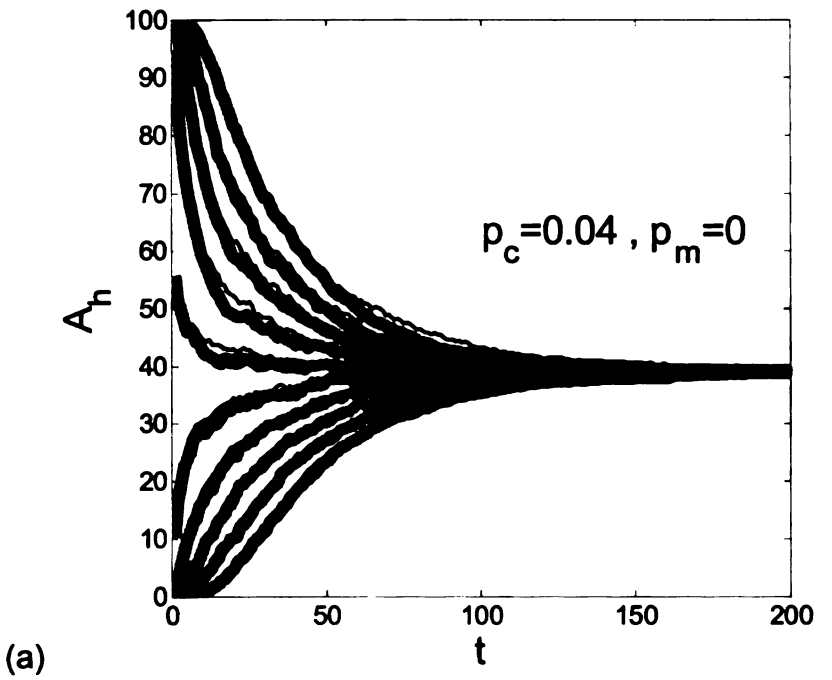
4.1 OneMax Problem

4.1.1 Fitness Proportional Selection

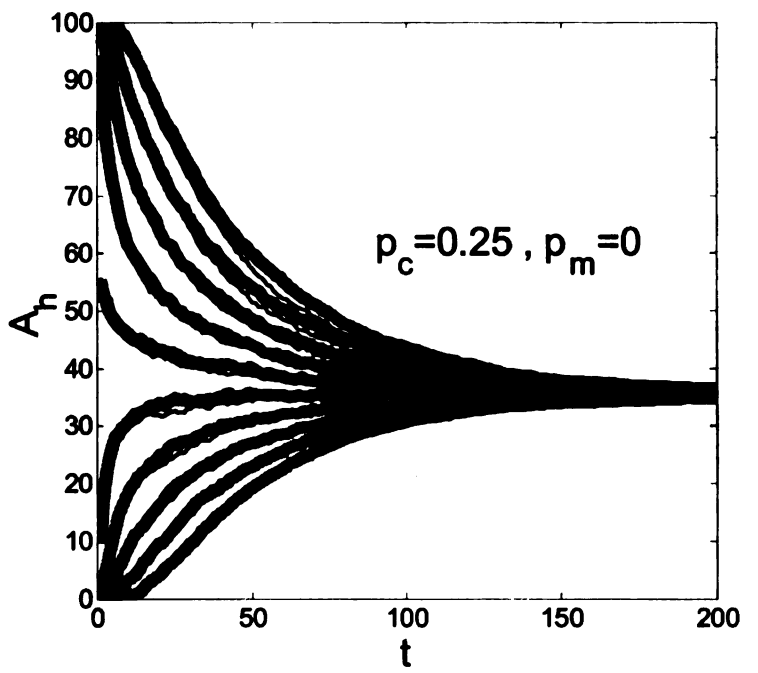
In this section we apply the model developed in Sections 2.2.1 through 2.2.5. The population size is 50 and the chromosome length is 100 genes. The experimental results are obtained by averaging 100 runs of the GA. The "high enough" crossover rate in our case is p_c = 300%, which means that 100% crossover is applied 3 times in a row before the selection. This rate of crossover is verified experimentally as "high enough" by observing that there is no significant change in the graphs of $A_h(t)$, $\kappa_1(t)$ and $\kappa_2(t)$ if a higher value of p_c is used. It can also be verified from Figure 2.6, since the correction weights, when p_c = 300%, are all very close to 1. In Figure 4.2 and Figure 4.2, the time variation of A_h is shown for crossover rates of p_c = 4%, 25%, 75% and 300%. We observed in our experiments with the model that when the crossover rate is too low, such as p_c = 4%, the statistics of correction weights taken only from the 5th generation are not enough and we needed adjustment by using the statistics at the 15th generation. Figure 4.2(a) shows the graph with this adjustment. For p_c = 25% and 75%, the statistics from only the 5th generation are used. The simulation for p_c = 300% is obtained by taking all c_i's as 1. In all four cases, no mutation is applied — i.e., p_m = 0. The graphs look quite similar to each other, except that there is some variation in the value to which the lines converge as time approaches 200 generations. We see that the limit value decreases from around 40 to 30 as p_c increases from 4% to 300%. This slight decrease observed in the experimental graphs is well captured by the model simulations. Such a change in

the converged values has a significant effect in the final fitness values of the population as a whole.

In Figure 4.3, the time evolution of A_h is shown for two different mutation rates, in both of which p_c is kept constant at 50%. In the first case the mutation rate is very low at p_m = 0.1%, while in the second case it is p_m = 2%. In both cases, the model predicts the mean allele behavior very well.

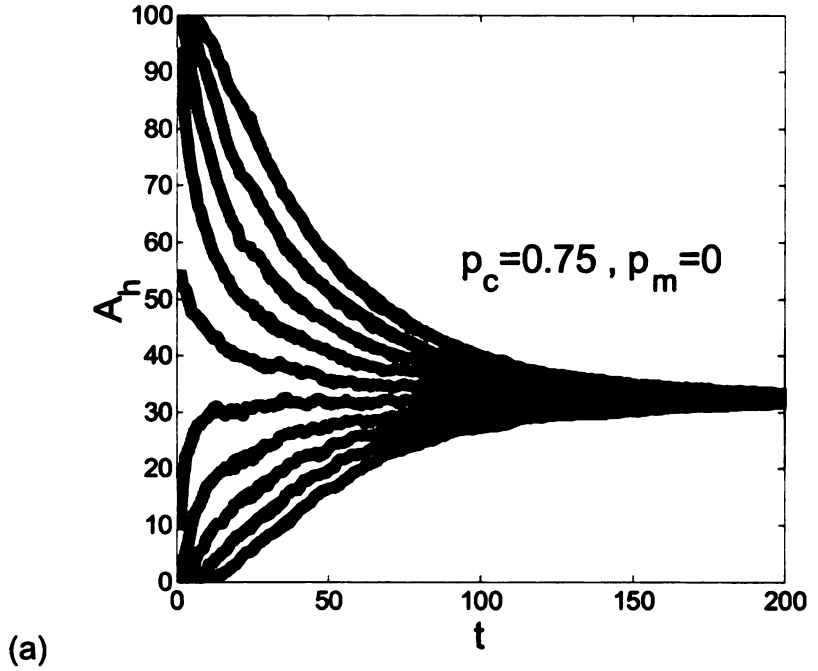


(a)

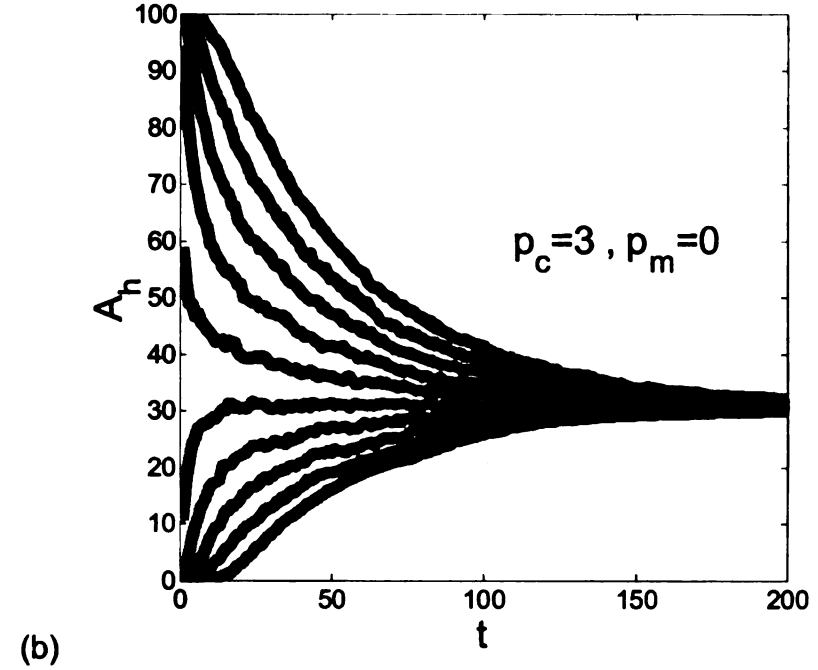


(b)

Figure 4.1 A_h as a function of time for four different cases where the crossover rate is 4 and 25 percent, respectively. There is no mutation. Black lines are the experimental averages over 100 GA runs and thick gray lines are the results obtained by model simulations. Population size is 50 and the chromosome length is 100 genes

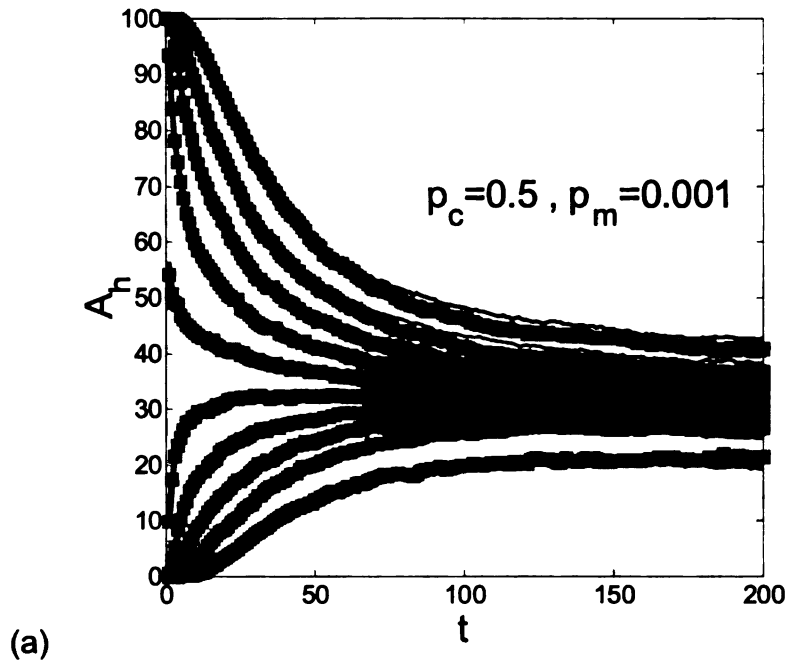


(a)

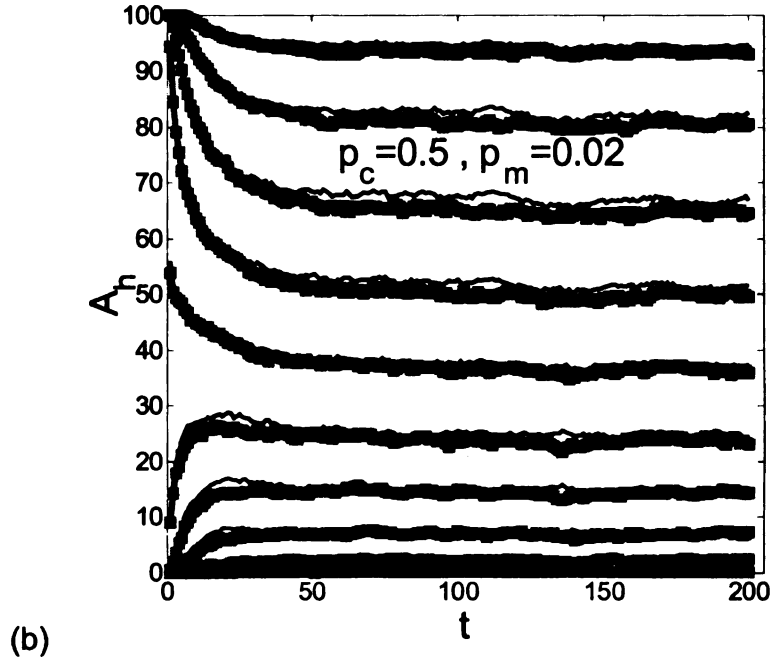


(b)

Figure 4.2 A_h as a function of time for four different cases where the crossover rate is 75 and 300 percent, respectively. There is no mutation. Black lines are the experimental averages over 100 GA runs and thick gray lines are the results obtained by model simulations. Population size is 50 and the chromosome length is 100 genes



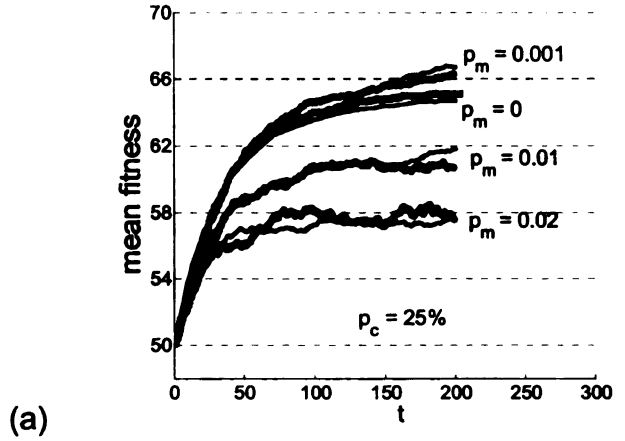
(a)



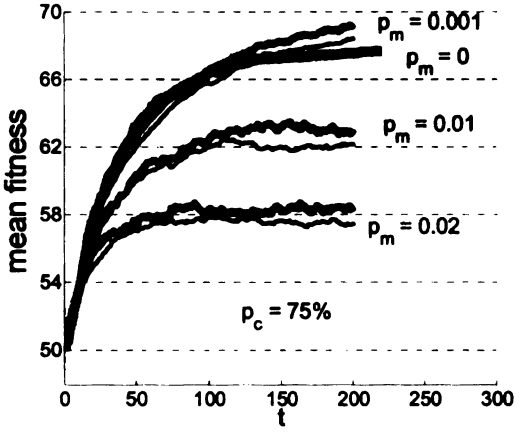
(b)

Figure 4.3 A_h as a function of time for two different cases where the mutation rate is 0.1 and 2 percent, respectively. The crossover rate is 50% in both cases. Black lines are the experimental averages over 100 GA runs and thick gray lines are the results obtained by model simulations

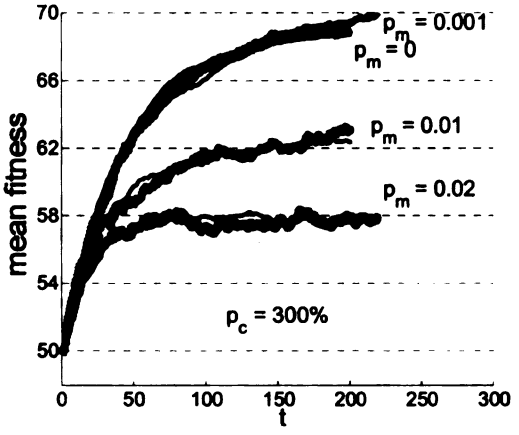
The experimental results and the model estimations of mean fitness for several values of p_m with crossover rates of p_c = 25%, 75% and 300% are shown in Figure 4.4. The impact on the mean fitness of changing p_c from 300% to 25% is more visible when p_m is low, around 0 or 0.1%. The effect of higher mutation rates dominates the dynamics of mean fitness evolution, and decreases the amount by which different crossover rates affect the mean fitness. The estimation of the fitness variance when p_c = 300% is shown in Figure 4.5 for various mutation rates, together with experimental averages. In all these graphs, we see that these dynamics of mean fitness and fitness variance are well captured by the simulation of the model.



(a)



(b)



(c)

Figure 4.4 The mean fitness for p_c = 25%, 75% and 300%. In each figure four different mutation rates, pm = 0%, 0.1%, 1% and 2%, are shown. Black lines are the experimental averages obtained by averaging over 100 GA runs and thick gray lines are the results obtained by model simulations

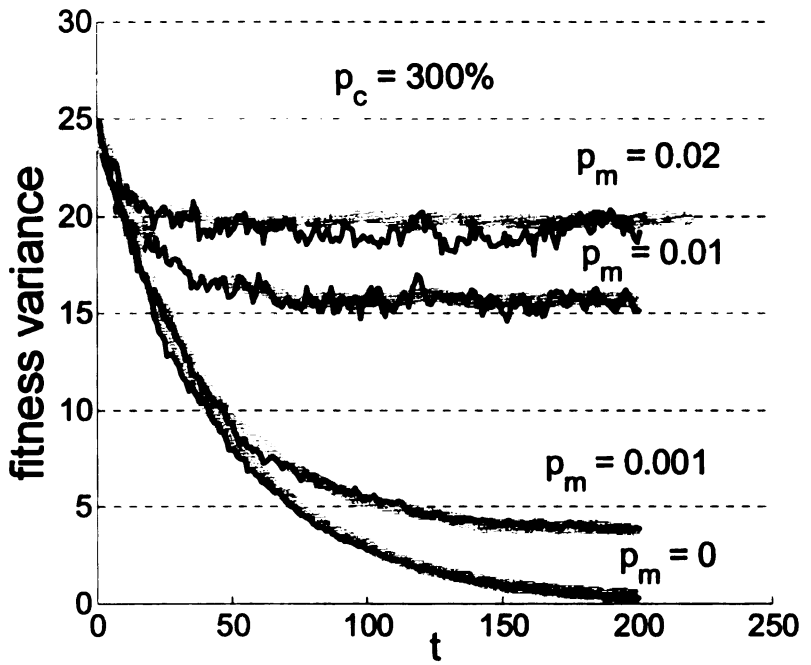


Figure 4.5 The fitness variance for four different rates of mutation, p_m = 0%, 0.1%, 1% and 2%, with crossover rate at 300%. Black lines are the experimental averages obtained by averaging over 100 GA runs and thick gray lines are the results obtained by model simulations

4.1.2 Boltzmann Scaling

In this section, we examine the model for β =0.1, 0.3 and 0.6 for both fixed and scaled β. The "high enough" crossover rate for Boltzmann selection is observed to be $p_c \geq 3000\%$, in other words, full crossover (i.e. 100%) is applied at least 30 times in the crossover operation.

In Figure 4.6 and Figure 4.7, we have the A_h curves. Figure 4.6 shows the case when no mutation is applied. The β values are fixed at 0.1 or 0.6.

Figure 4.7 has the graphs for 0.1%, 1% and 2% mutation rates with β=0.1 selection pressure. The corresponding mean fitness graphs are shown in Figure 4.9 and Figure 4.9. This figure also contains the mean fitness curves for the corresponding GA parameters when β is scaled. The comparison of mean fitness graphs for fixed-β Boltzmann selection for three different selection pressures, $\beta = 0.1, 0.3, 0.6$, without any mutation are seen in Figure 4.10. The corresponding fitness variance graphs are seen in Figure 4.11.

When the selection pressure, β is scaled, the A_h curves look as in Figure 4.12. Lastly, the comparison of fitness variance curves for different mutation rates of $\beta = 0.1$ and $\beta = 0.6$ are seen in Figure 4.13.

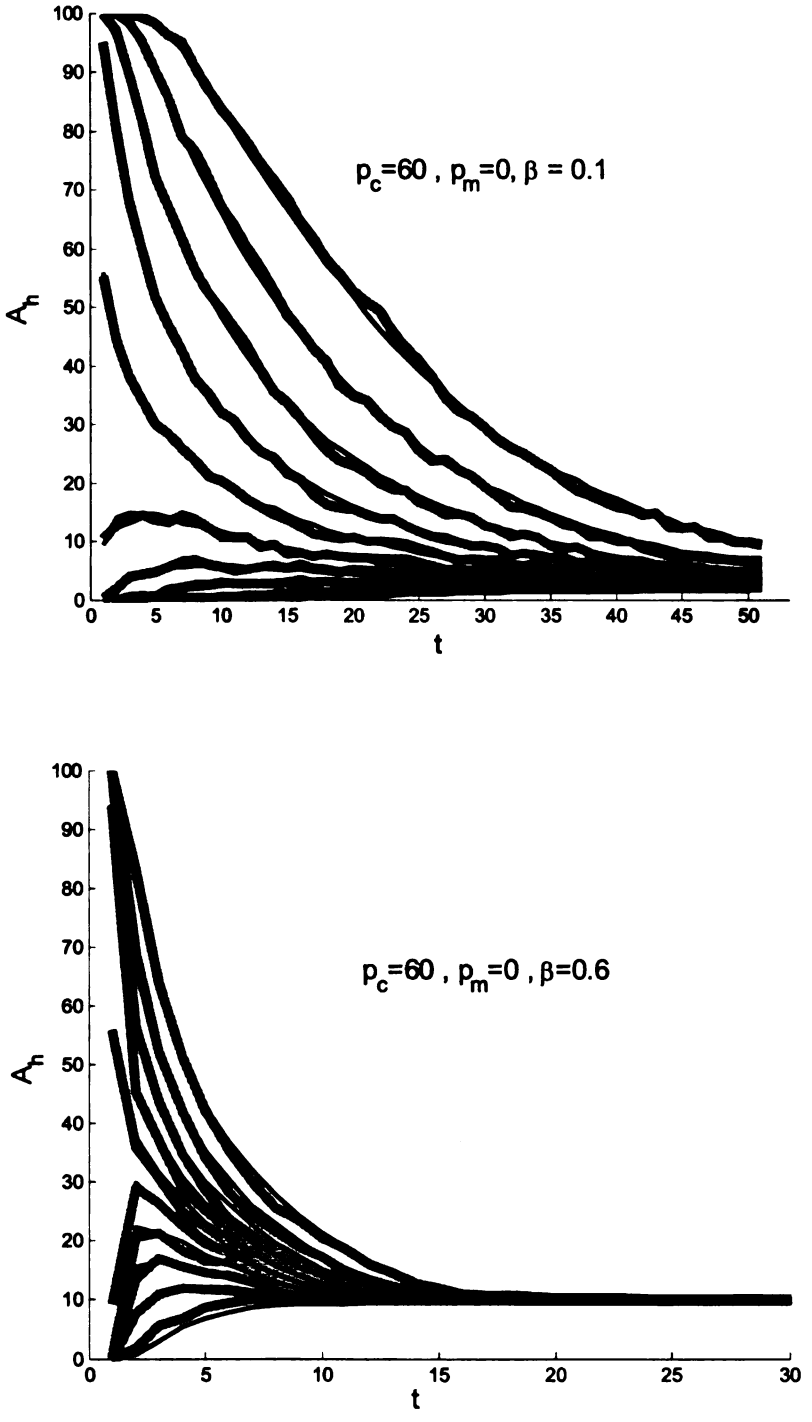


Figure 4.6 A_h curves for fixed β Boltzmann selection. $p_m = 0, p_c = 60$. The first graph is for $\beta = 0.1$, the second, for $\beta = 0.6$

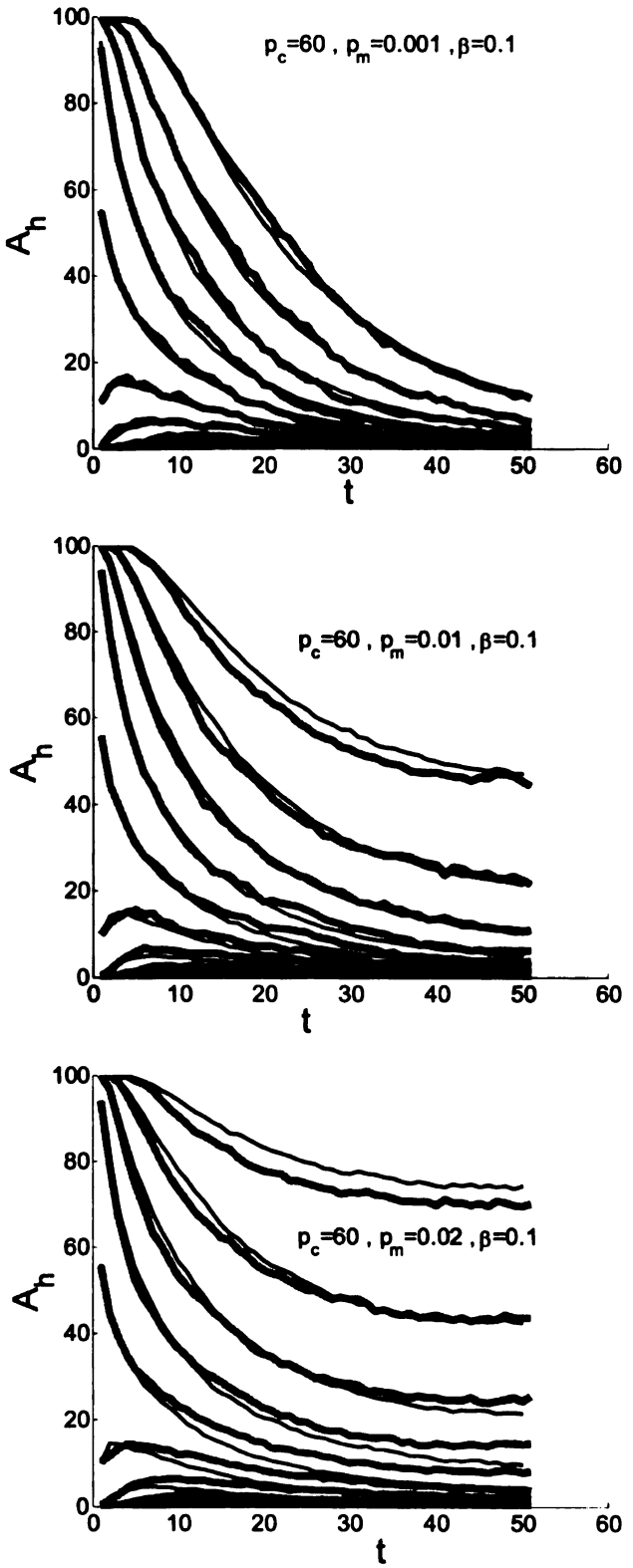


Figure 4.7 A_h curves for fixed $\beta = 0.1$ Boltzmann selection.

$p_m = 0.01, 0.1, 0.2$, $p_c = 60$

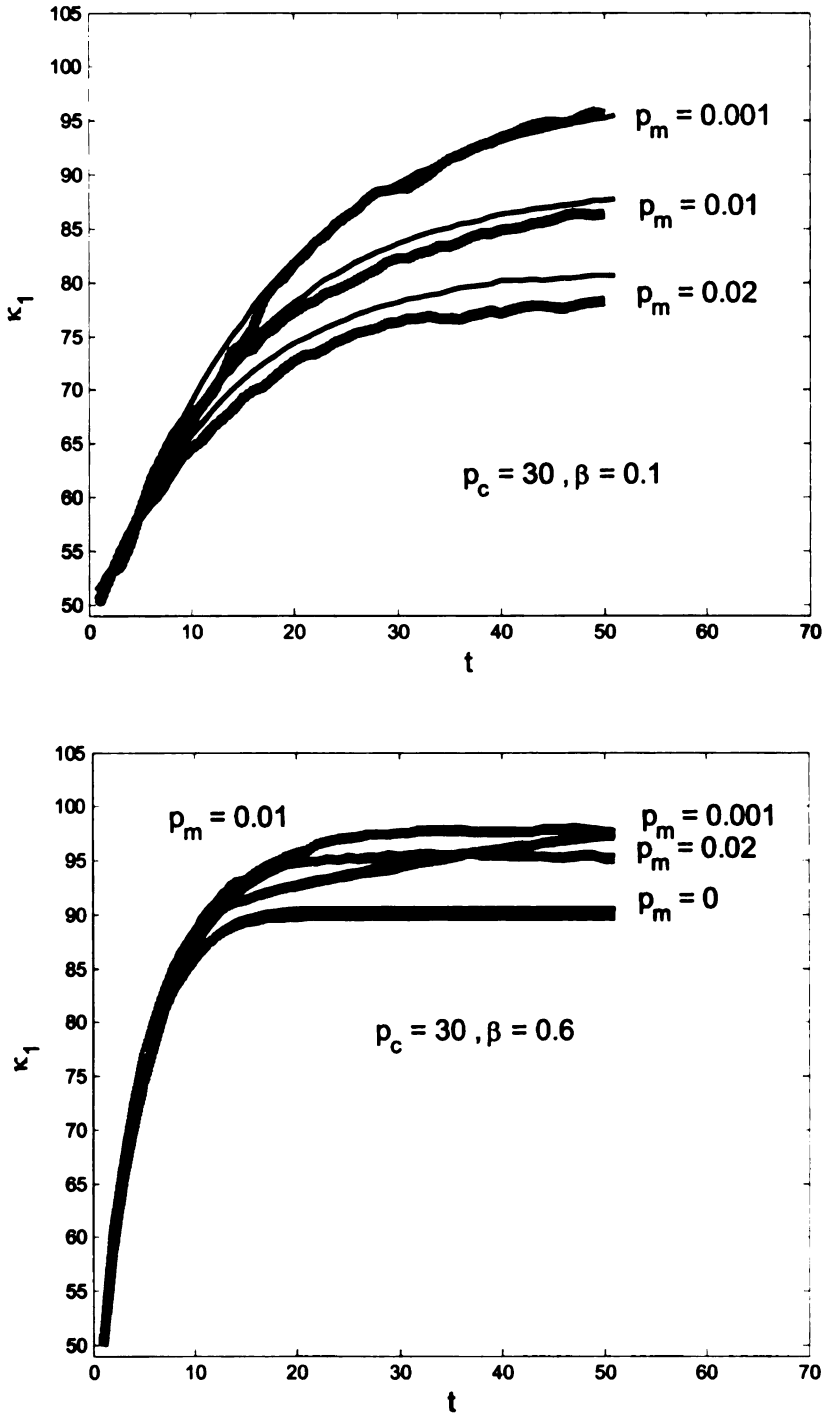


Figure 4.8 The mean fitness graphs for fixed-β Boltzmann selection. The graphs show mean fitness for different mutation rates with $\beta = 0.1$ or $\beta = 0.6$

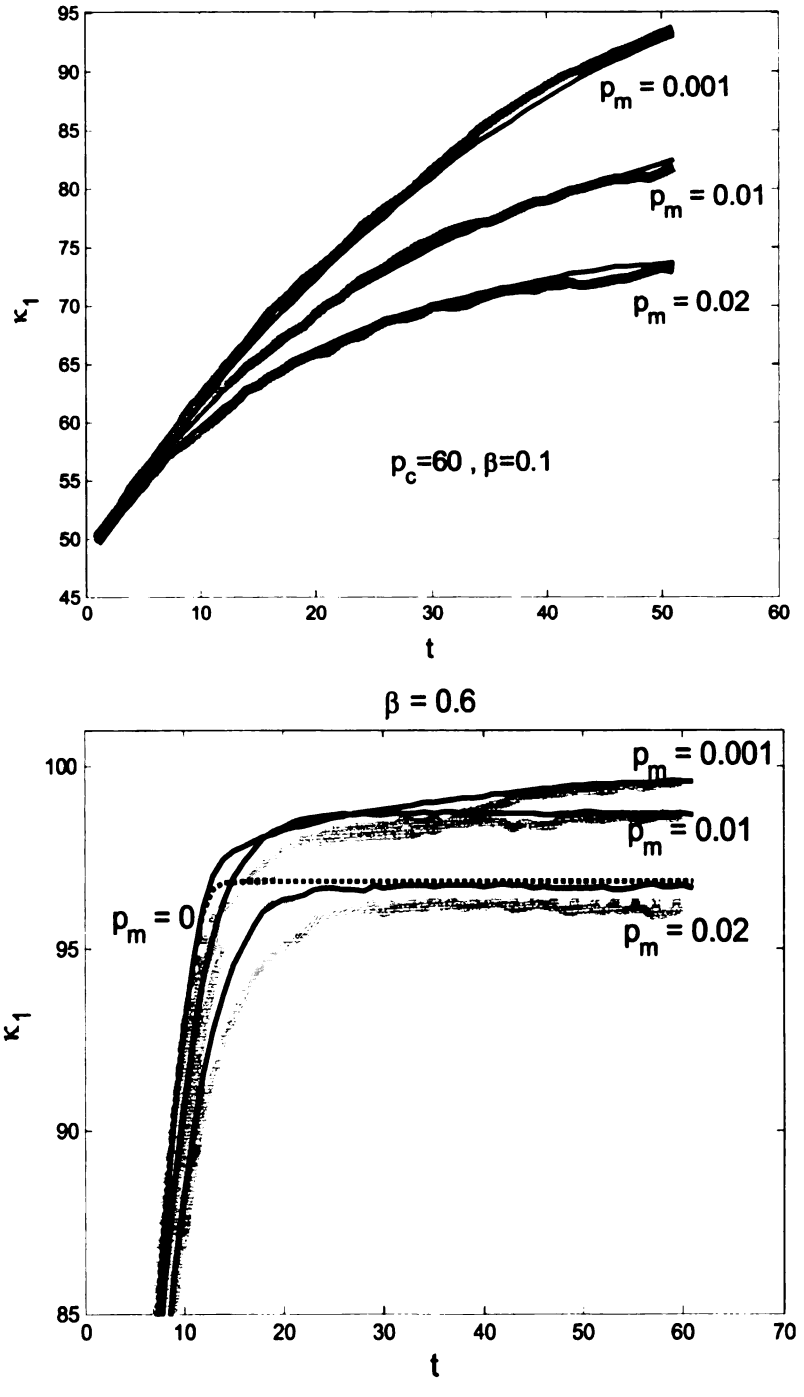


Figure 4.9 The mean fitness graphs for scaled-β Boltzmann selection. The graphs show mean fitness for different mutation rates with $\beta = 0.1$ or $\beta = 0.6$

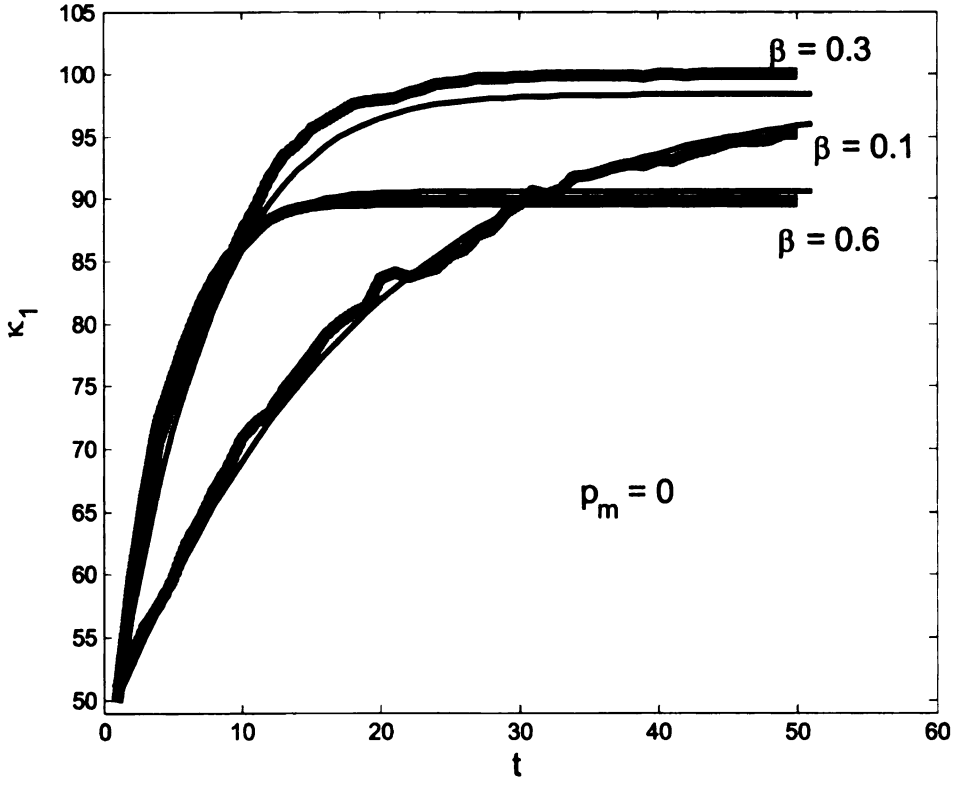


Figure 4.10 The mean fitness graphs for fixed β Boltzmann selection for three different selection pressures $\beta = 0.1, 0.3, 0.6$ with $p_m = 0$

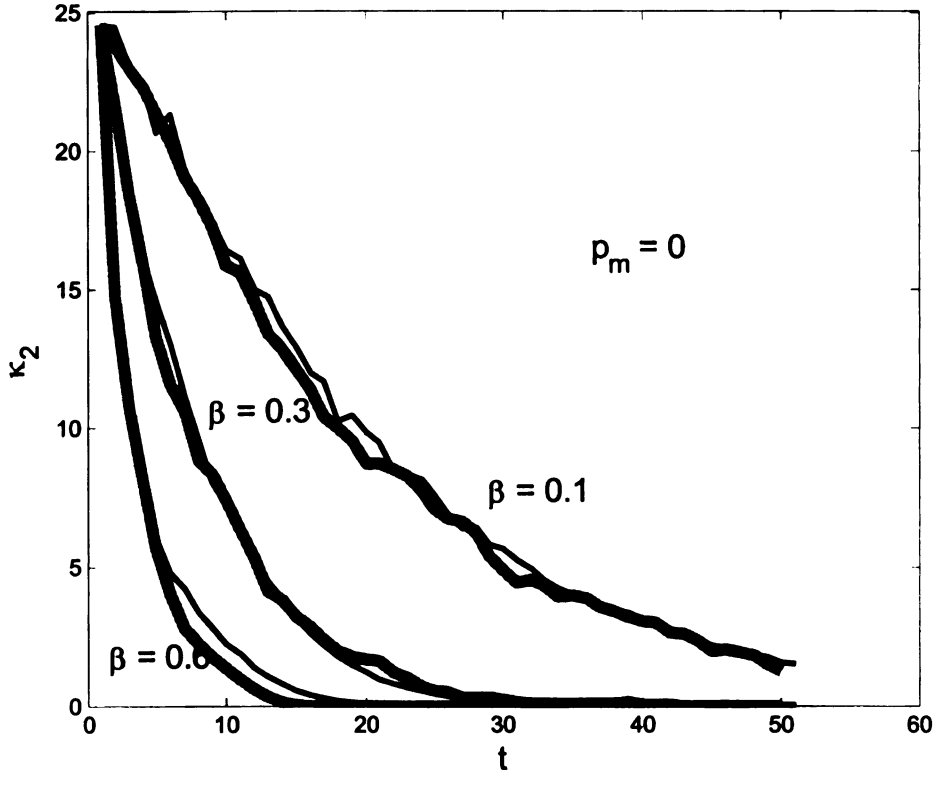


Figure 4.11 The fitness variance for fixed β Boltzmann selection for three different selection pressures $\beta = 0.1, 0.3, 0.6$ with $p_m = 0$

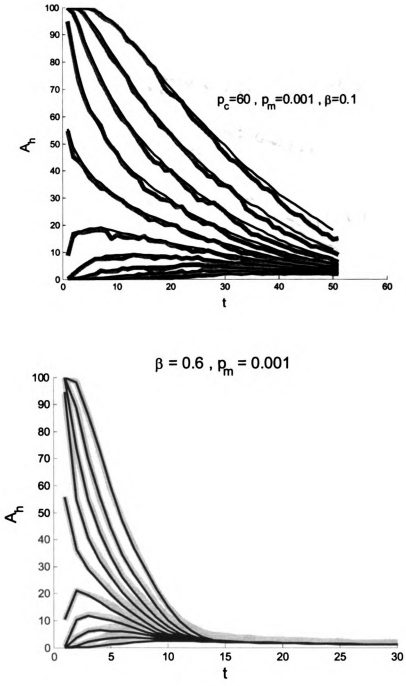


Figure 4.12 A_h curves for scaled-β Boltzmann selection. $p_m = 0.001, p_c = 60$. The first graph for $\beta = 0.1$, the second $\beta = 0.6$

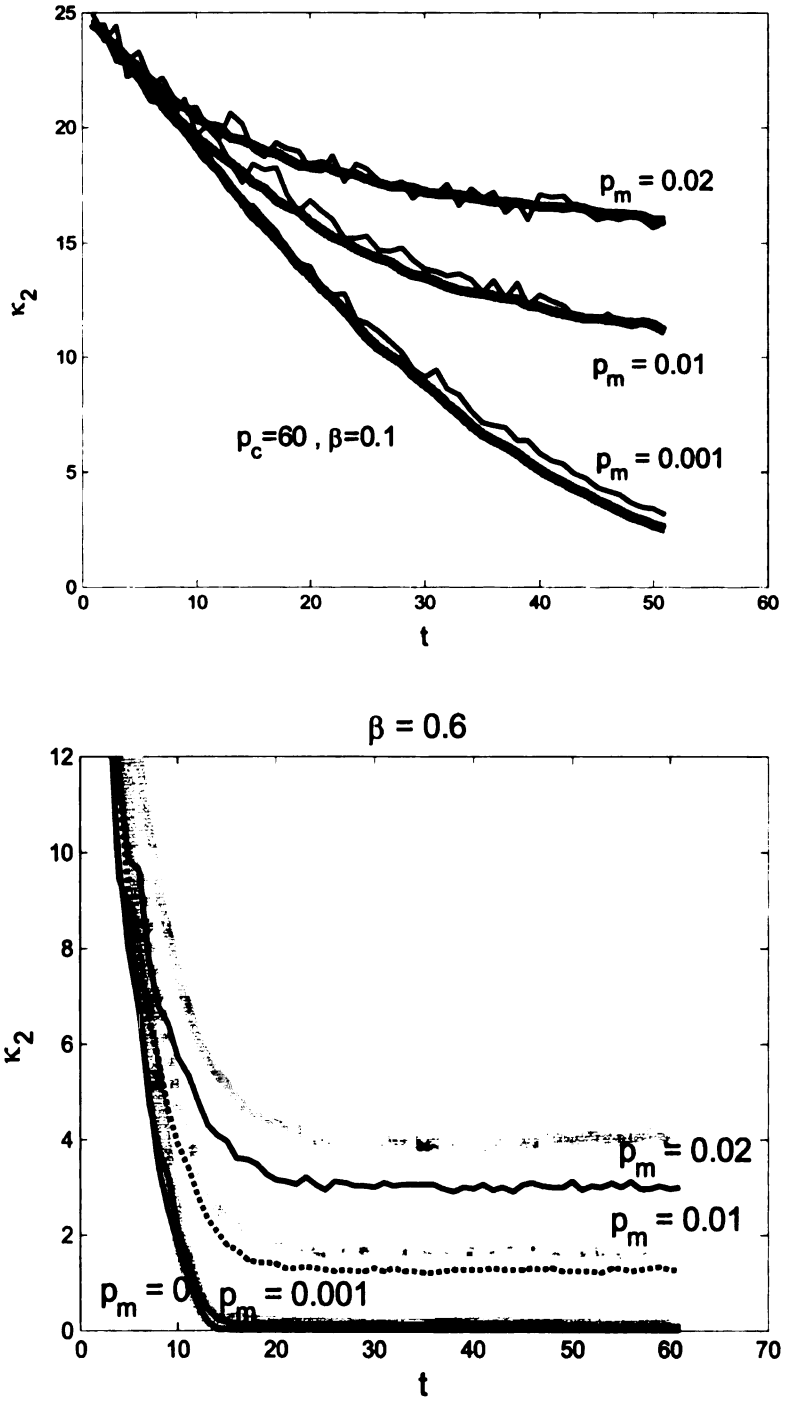


Figure 4.13 The fitness variance for scaled β Boltzmann selection for two different selection pressures $\beta = 0.1$ and $\beta = 0.6$ showing each selection pressure for different mutation rates

4.1.3 Weighted Fitness Function

In this section, we study the GA with two different weight profiles. We will call them Profile-A and Profile-B. The graphs of the weight profiles as a function of the gene locus are shown in Figure 4.14 and their equations are given by

$$\text{Profile-A} \qquad w_i = \begin{cases} -\frac{1.8}{49}(i-1)+1.9 & \text{if } 1 \le i \le 50 \\ \frac{1.8}{49}(i-51)+0.1 & \text{if } 51 \le i \le 100 \end{cases}$$

$$\text{Profile-B} \qquad w_i = \begin{cases} 1 & \text{if } 1 \le i \le 50 \\ 3-\frac{i-1}{25} & \text{if } 51 \le i \le 100 \end{cases} \tag{4.1}$$

In Profile-A, the weight increases from 0.1 to 1.9 linearly as we go from the middle of the chromosome towards the ends. Hence the gene values in the middle of the chromosome contribute less to the fitness compared to the ones near the ends of the chromosome. The maximum possible fitness value of this profile is 100, which is obtained when all the alleles are 1.

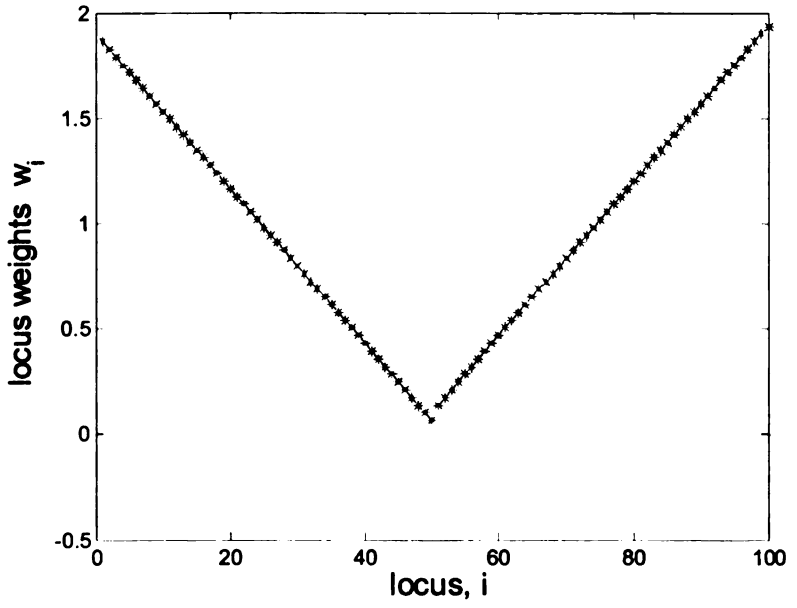
Profile-B is constant 1 for the first half of the chromosome. The weights of this profile decrease from 1 to -1 linearly in the second half of the chromosome. Note that Profile-B takes negative values in the last quarter of the chromosome. This means that having a 1 in this part of the chromosome causes the fitness to decrease. So, the GA would try to have 0's instead of 1's in the last quarter of the chromosome. The maximum possible fitness value of this profile is 63, which is

obtained when all the alleles from locus 1 to 75 are 1 and from locus 77 to 100 are 0. The value of locus 76 does not matter in this case since its weight is 0.

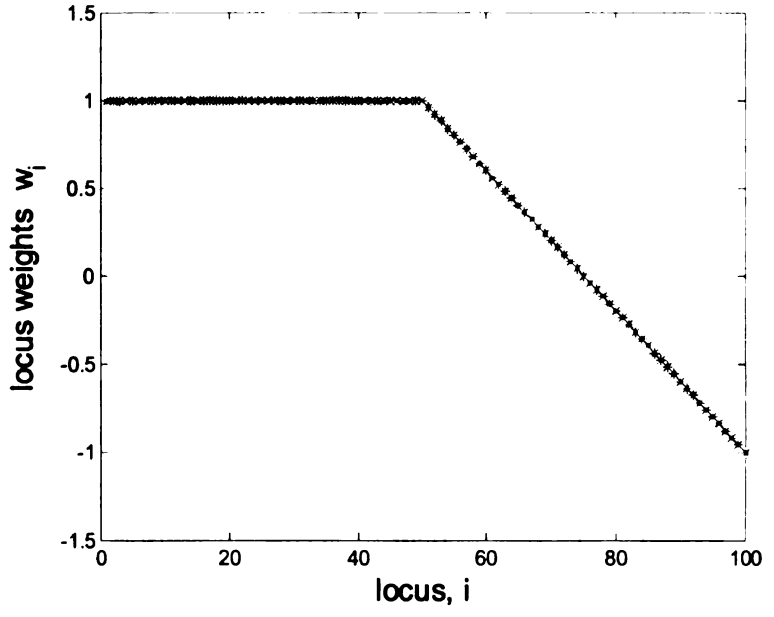
First we study Profile-A. The A_h curves for fitness proportional selection and Boltzmann selection with $\beta = 0.6$, without mutation are shown in Figure 4.15 and Figure 4.16. In Figure 4.15 (b) we also see the graphs for fitness proportional selection when the mutation rate is 2%. Figure 4.17 and Figure 4.18 have the mean fitness curves for several cases. In Figure 4.17, we see the graphs for different mutation rates of the fitness proportional selection. In Figure 4.18(a), the fitness proportional selection is compared with $\beta = 0.1$ and $\beta = 0.6$ of Boltzmann selection with scaled β. And, in Figure 4.18(b) $\beta = 0.6$ case of Boltzmann selection is shown for mutation rates of 0%, 0.1% and 1%. The corresponding fitness variance curves of all the cases of are shown in Figure 4.19 and Figure 4.20.

When we consider Profile-B, A_h curves look as in Figure 4.22 and Figure 4.22, where we see the curves for $\beta = 0.1$ and $\beta = 0.6$ with no mutation and $\beta = 0.6$ with 1% mutation. Comparison of mean fitness curves for fitness proportional selection and Boltzmann selection with $\beta = 0.1$ and $\beta = 0.6$ is in Figure 4.23. In Figure 4.24, the mean fitnesses with Boltzmann selection ($\beta = 0.6$) for different mutation rates are shown for time > 9, to be able to show the details of the graphs. Lastly, the fitness variance curves comparing fitness

proportional selection and Boltzmann selection with $\beta = 0.1$ and $\beta = 0.6$ are in Figure 4.25.



Profile-A



Profile-B

Figure 4.14 Profile A and Profile B for the weighted fitness function;

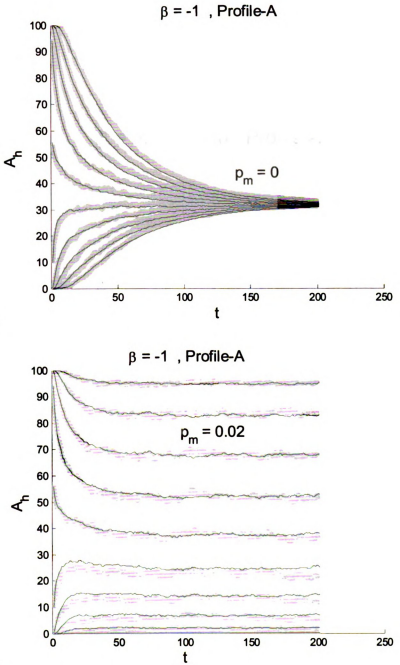


Figure 4.15 A_h curves for Profile-A with fitness proportional selection, $p_m = 0$ and $p_m = 0.2$

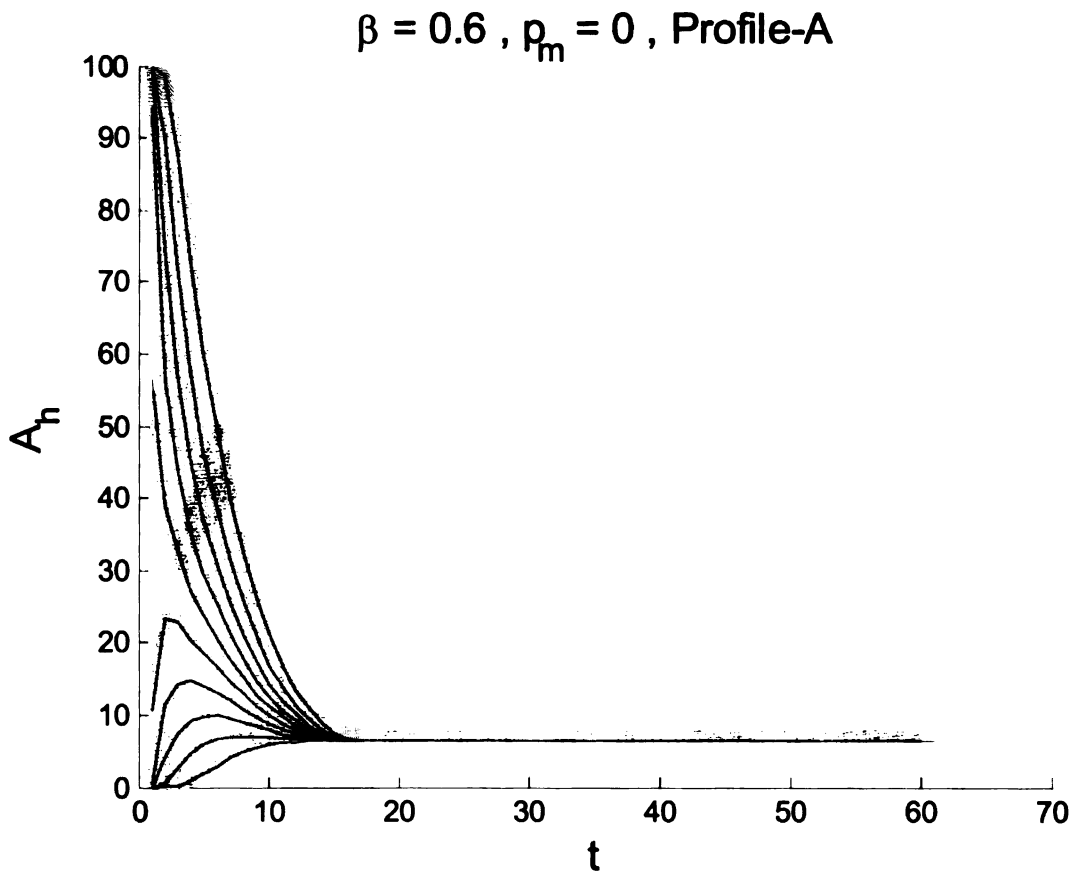


Figure 4.16 A_h curves for Profile-A for scaled-β Boltzmann selection. with $\beta = 0.6$ and $p_m = 0, p_c = 60$.

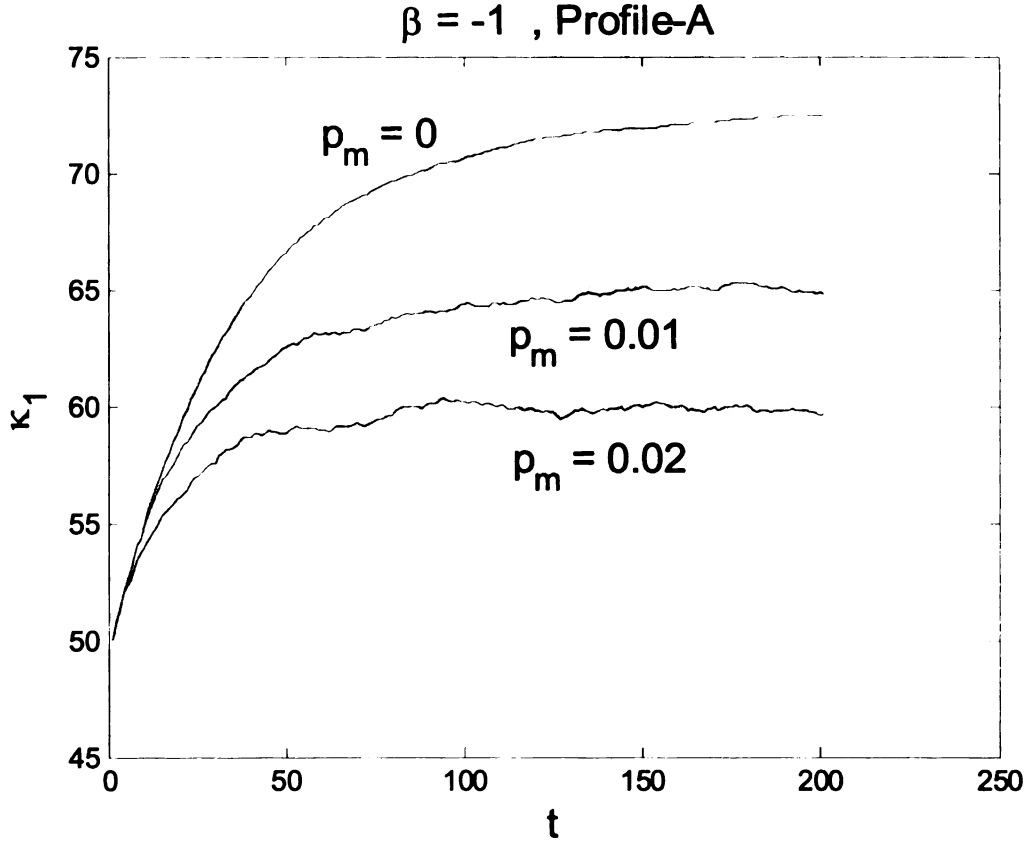
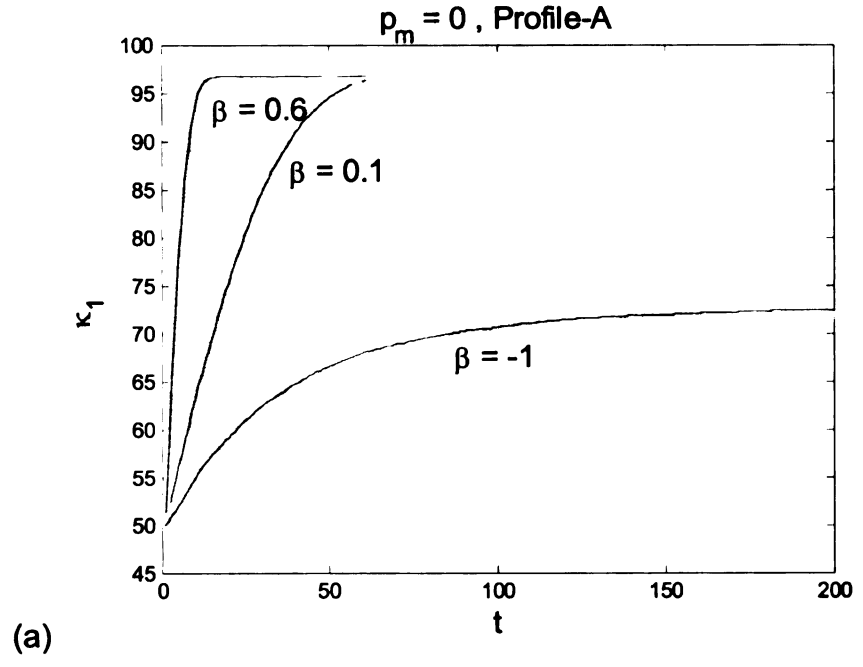
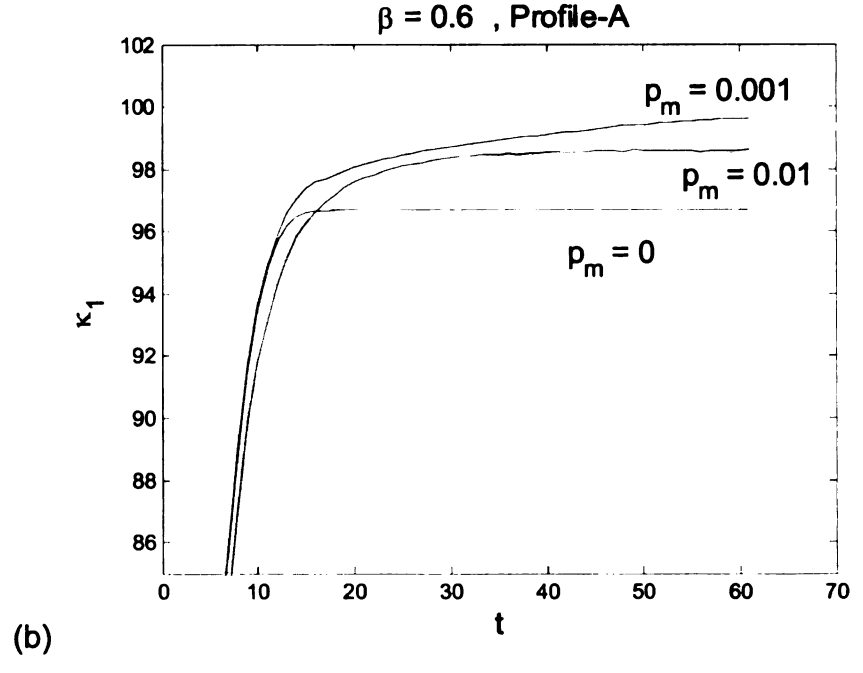


Figure 4.17 Mean fitness curves for Profile-A for fitness proportional selection with different p_m's

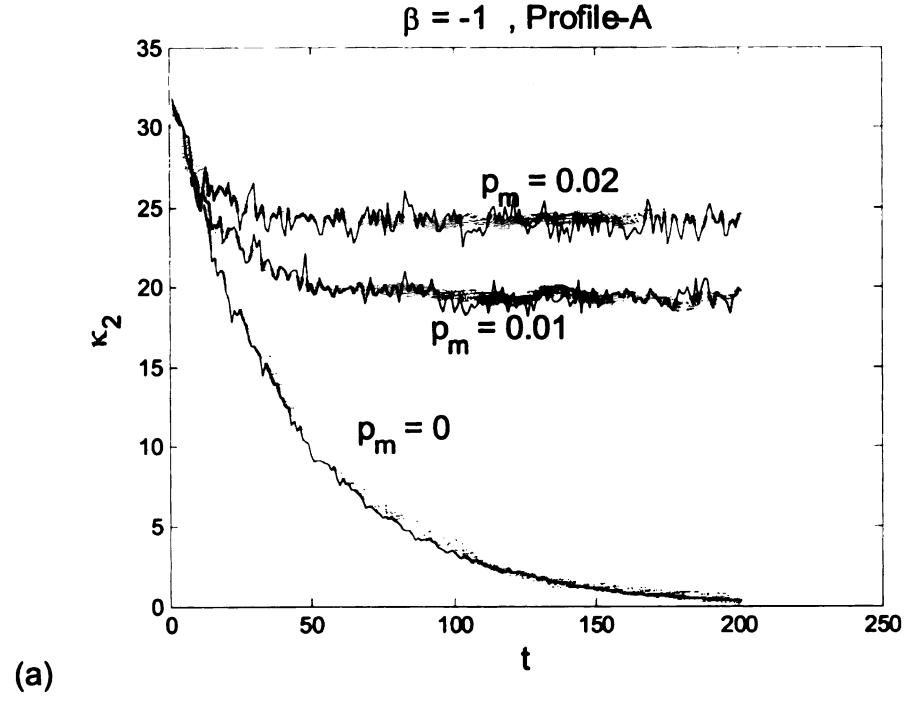


(a)

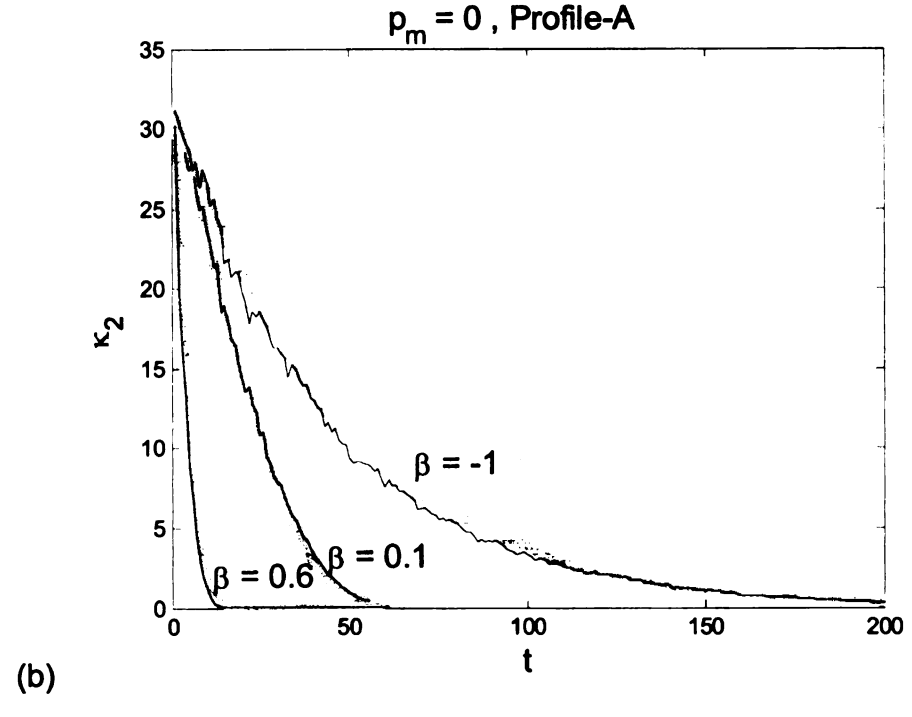


(b)

Figure 4.18 Mean fitness curves for Profile-A with Boltzmann selection, scaled β : (a) comparing different β cases; (b) for $\beta = 0.6$ with different p_m's



(a)



(b)

Figure 4.19 Fitness variance curves for Profile-A: (a) for fitness proportional selection with different p_m's; (b) comparing different β cases

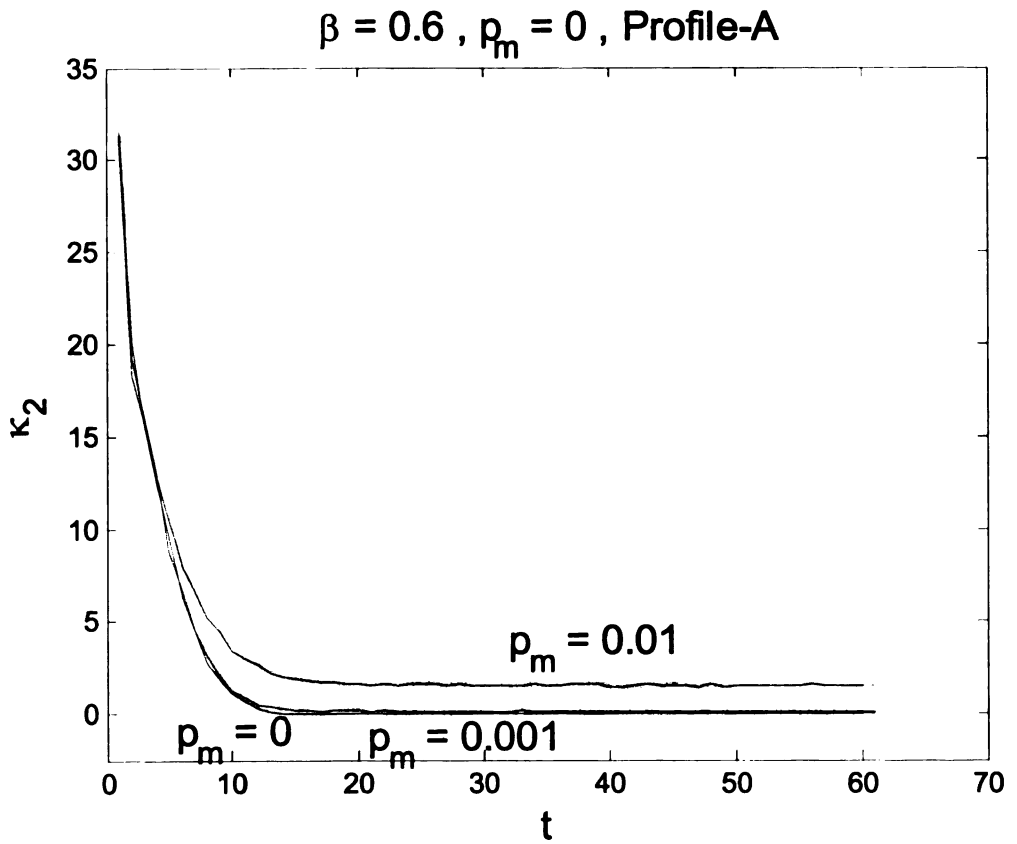
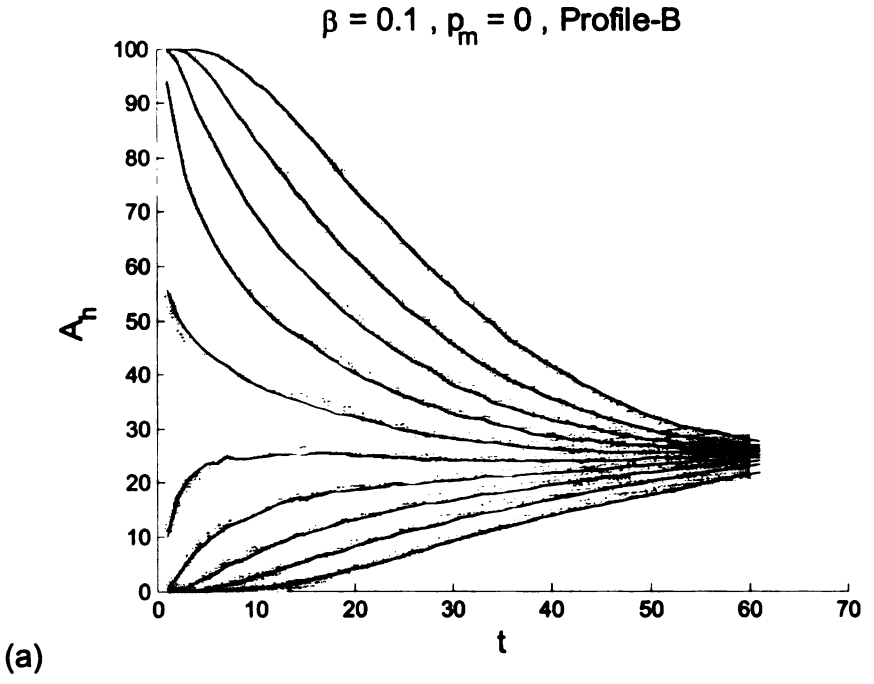
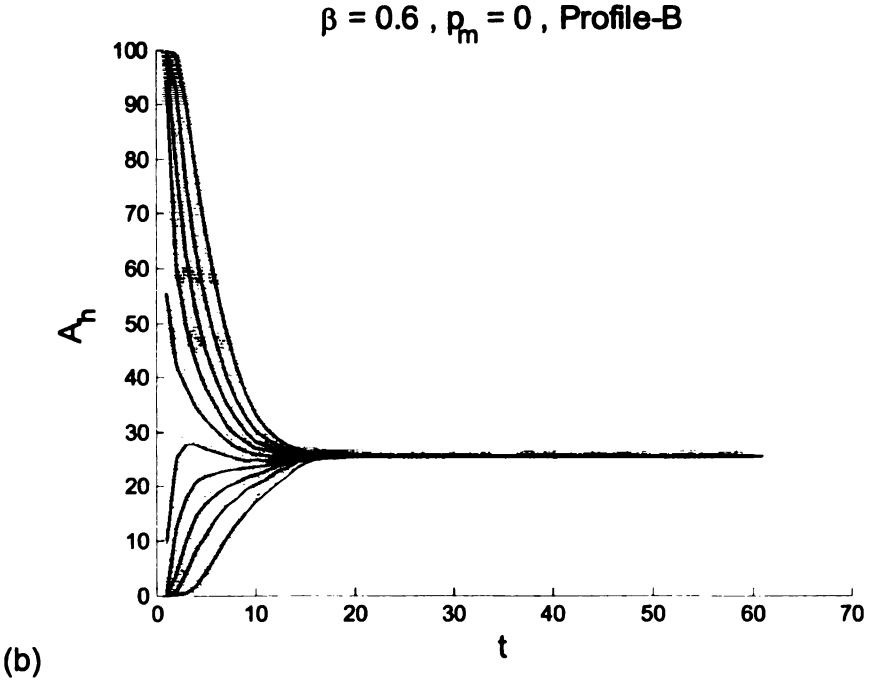


Figure 4.20 Fitness variance curves for Profile-A for $\beta = 0.6$ with different p_m's. Boltzmann selection is with scaled β



(a)



(b)

Figure 4.21 A_h curves for Profile-B with scaled-β Boltzmann selection.
(a) $\beta = 0.1$, no mutation; (b) $\beta = 0.6$ with no mutation

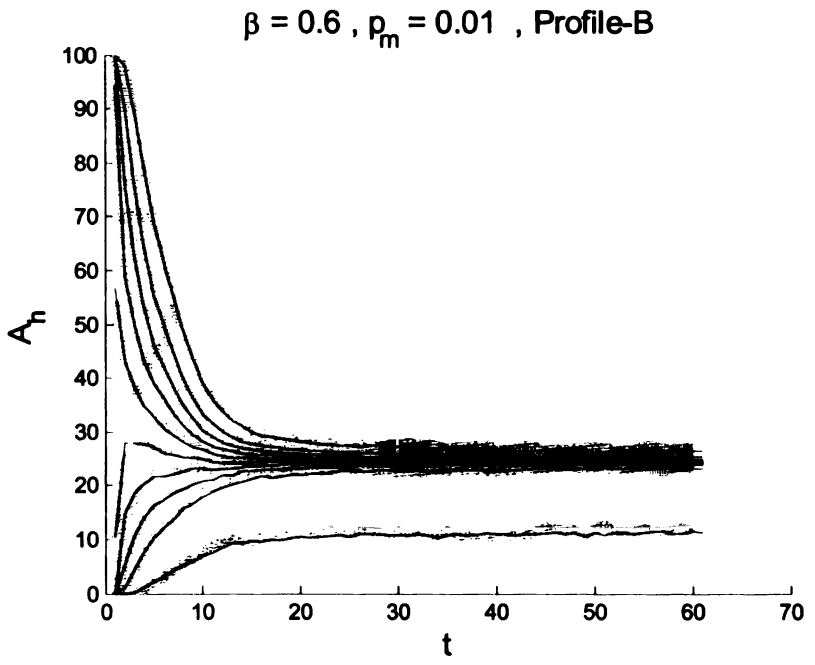


Figure 4.22 A_h curves for Profile-B with scaled-β Boltzmann selection. $\beta = 0.6$ with 1% mutation. In all cases $p_c = 60$

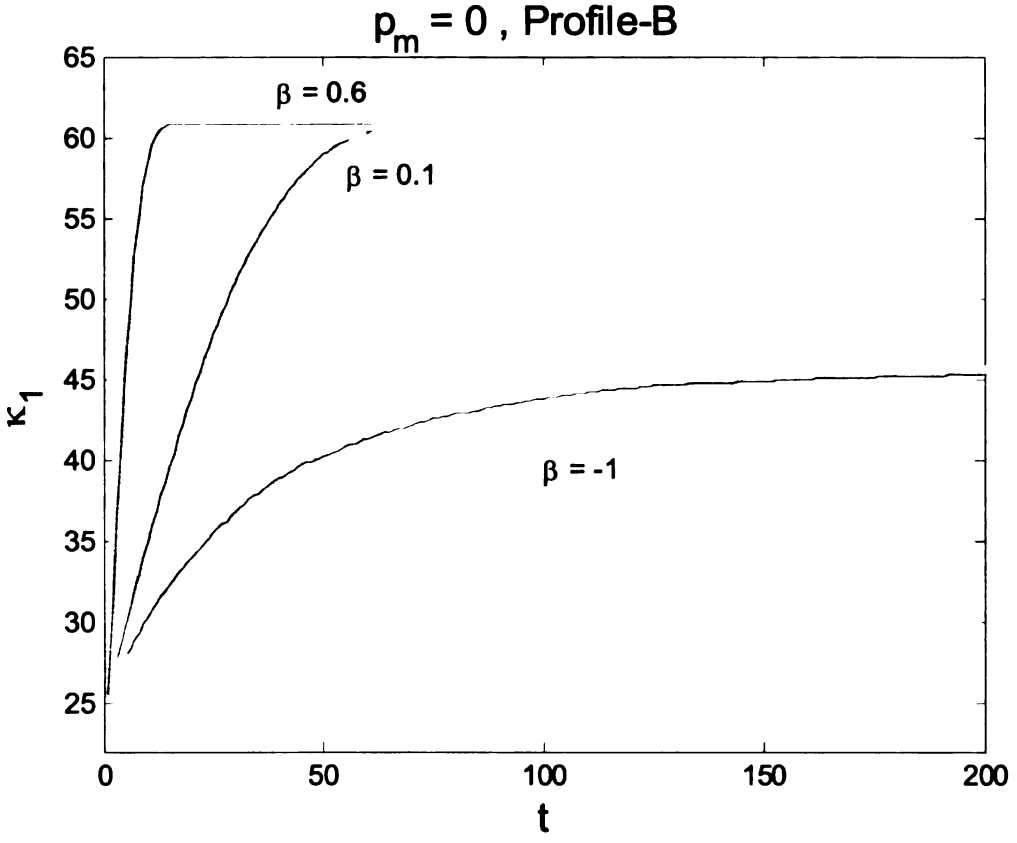


Figure 4.23 Mean fitness curves for Profile-B, showing fitness proportional selection and Boltzmann selection with $\beta = 0.1$ and $\beta = 0.6$

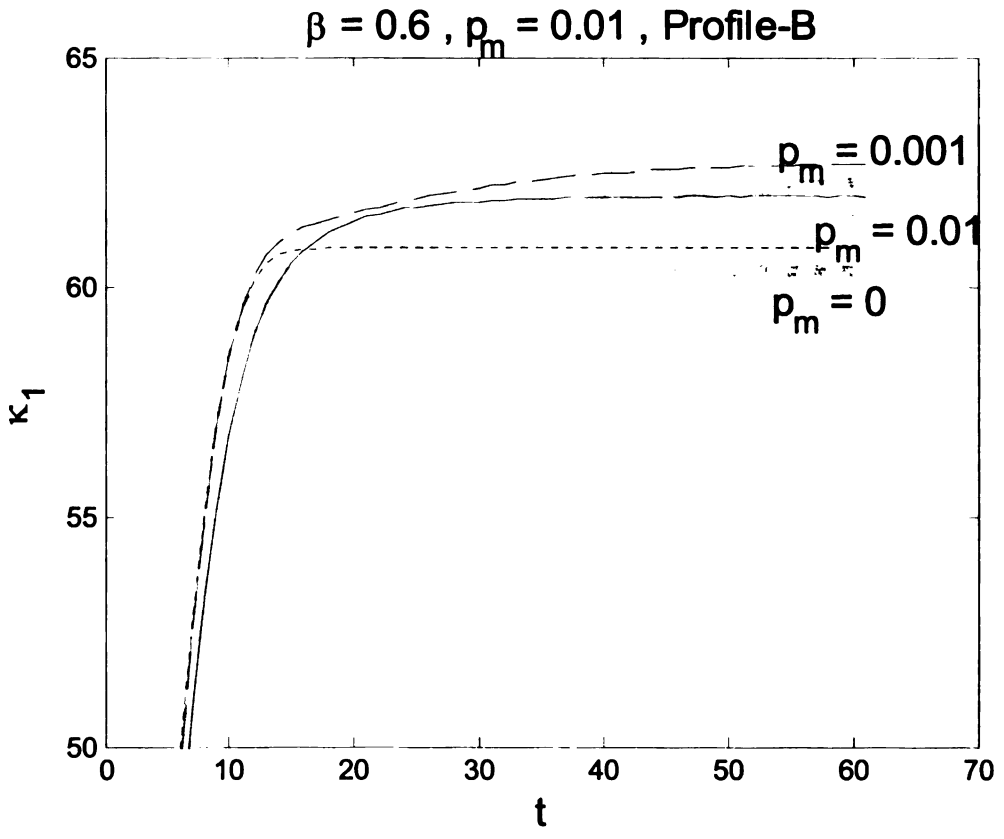


Figure 4.24 Mean fitness curves for Profile-B. Boltzmann selection with $\beta = 0.6$, and different mutation rates

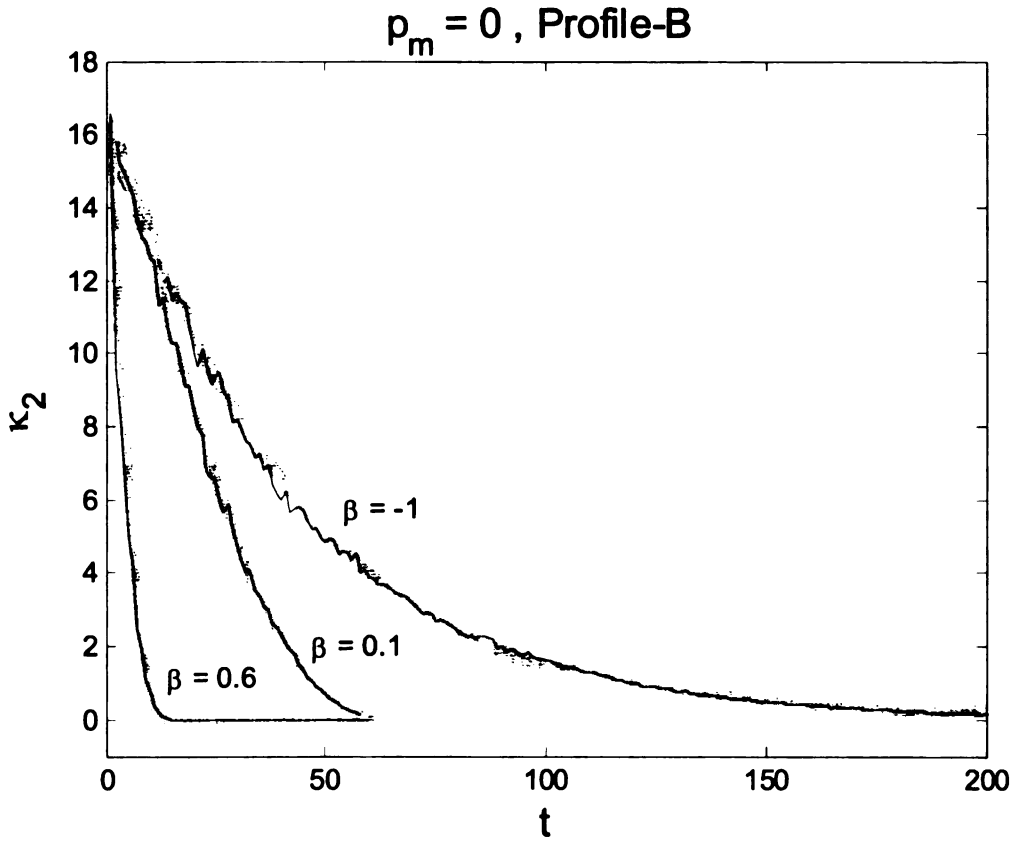


Figure 4.25 Fitness variance curves for Profile-B, showing fitness proportional selection and Boltzmann selection with $\beta = 0.1$ and $\beta = 0.6$

4.2 Deceptive Function Problem

In this Section, we apply the GA to a 3-bit deceptive function. We study only the case where the crossover rate is high enough. The chromosome length is chosen as *L* = 99, and the population size is *P* = 50. So, we have 33 deceptive blocks, each of which is 3 bits long. In all the cases studied in this section, the values of *A* and *B* are chosen as *A=4* and *B=1*. Note that the maximum possible fitness for a chromosome, when the fitness is not weighted, is $A \times 33 = 4 \times 33 = 132$, which is achieved when all the alleles are 1.

4.2.1 Fitness Proportional Selection

We apply the model developed in Sections 3.2.1 through 3.2.4. In Figure 4.26, we have the curves for both the evolution of alleles with zero value, A_h^{α}, and the evolution of the deceptive blocks, A_h^{γ}, $h = 0, 0.1, \ldots, 0.9$, when the fitness proportional selection is applied. The mutation rate is zero and the crossover rate p_c = "high enough". In Figure 4.27, A_h^{α} and A_h^{γ} curves are shown when the mutation rate is 0.1%. The graphs of mean fitness and fitness variance are shown in Figure 4.28 comparing cases with different mutation rates.

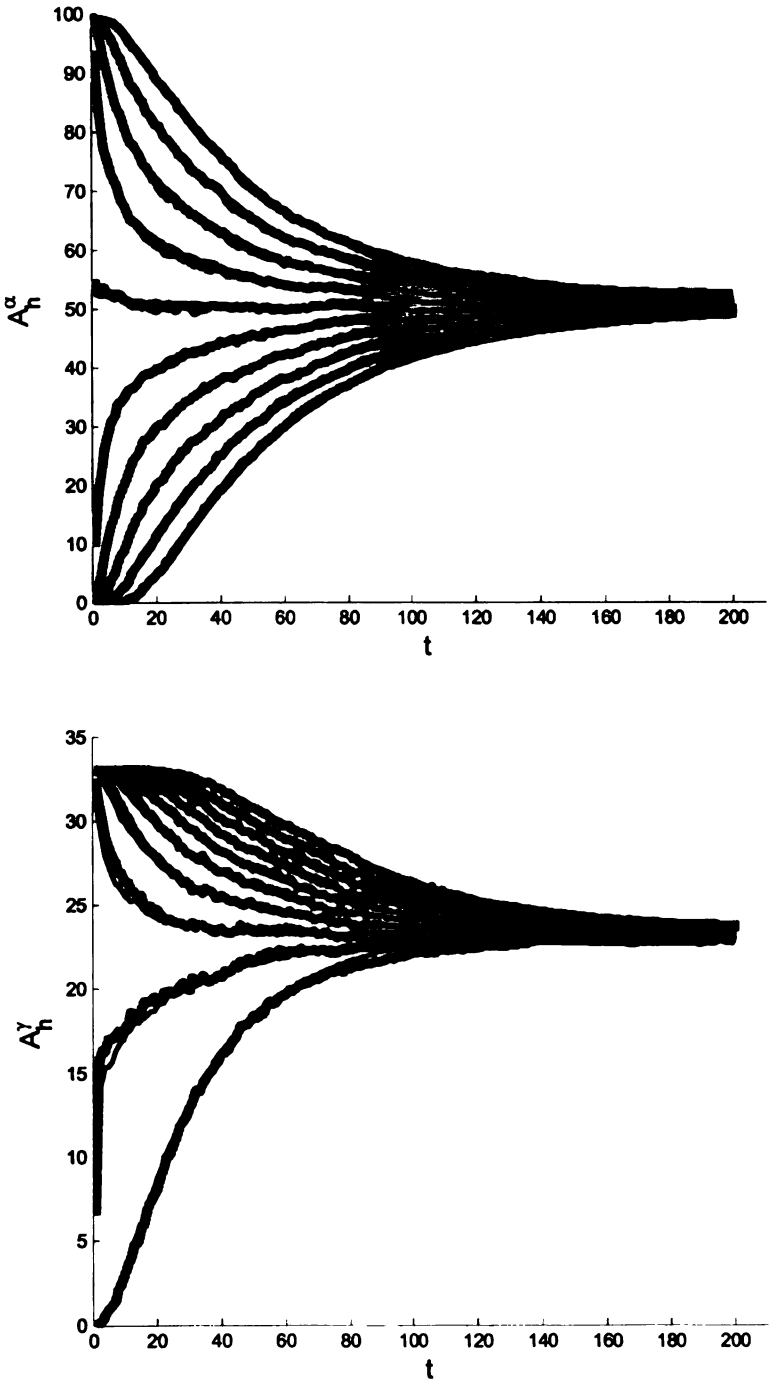


Figure 4.26 A_h^α and A_h^γ curves for fitness proportional selection. p_c = "high enough", p_m = 0. Black lines are from GA run, gray lines are from model simulation

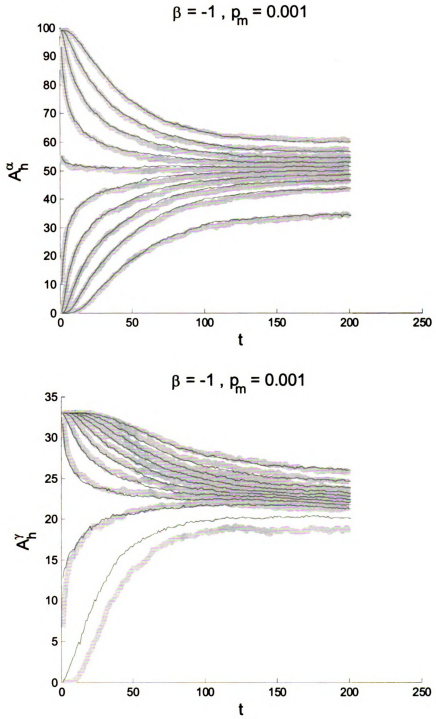


Figure 4.27 A_h^{α} and A_h^{γ} curves for fitness proportional selection with mutation rate p_m = 0.001, and p_c = "high enough"

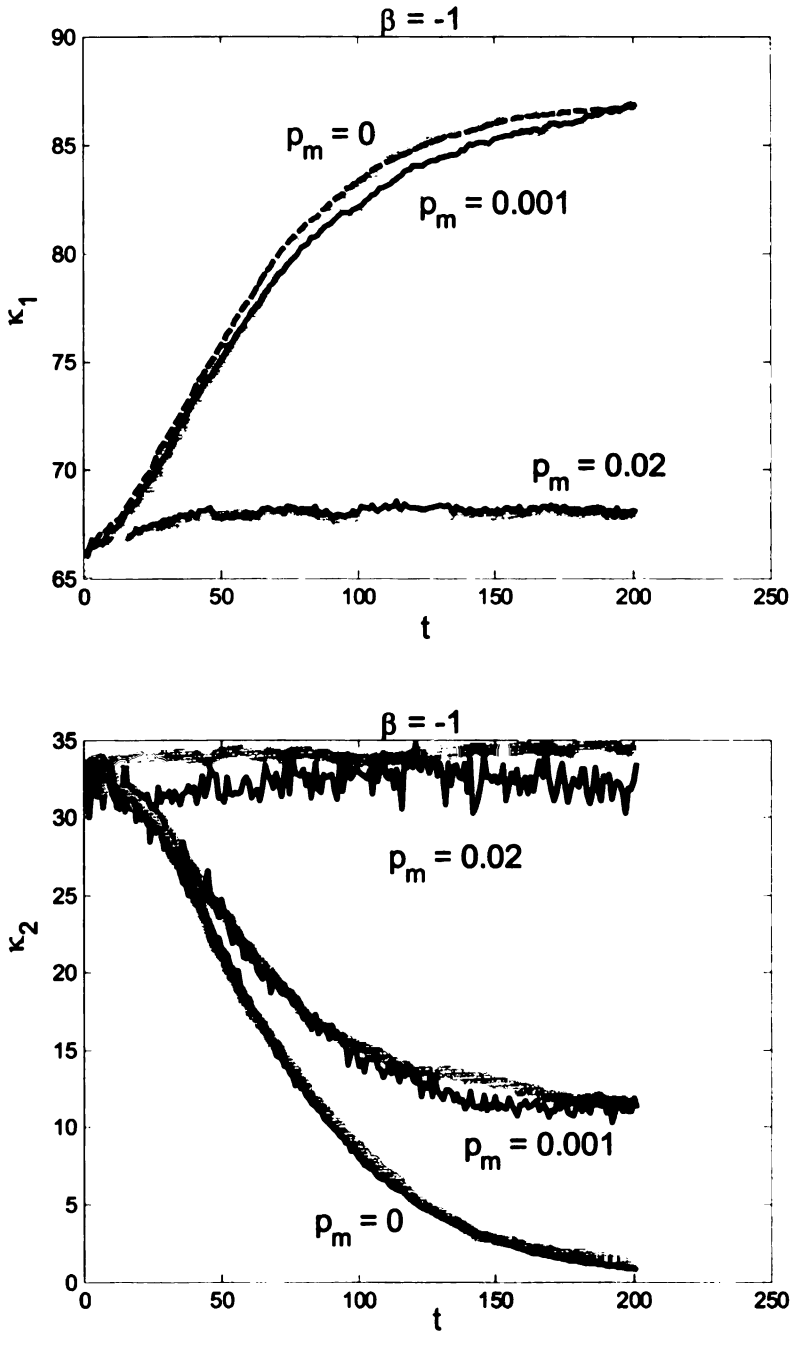


Figure 4.28 Mean fitness and fitness variance curves for fitness proportional selection. p_c = "high enough", p_m = 0 , 0.001 ,and 0.2. Black lines are from GA run, gray lines are from model simulation

4.2.2 Boltzmann Selection

Application of scaled-β Boltzmann selection with 0.1 and 0.6 selection pressures to our deceptive function yield A_h^α and A_h^γ curves in Figure 4.29 and Figure 4.30, in which cases the mutation rate is taken as zero. Comparison of mean fitness and fitness variance for selection pressures of 0.1, 0.3 and 0.6 are in Figure 4.31. Lastly, in Figure 4.32, we have the mean fitness and fitness variance curves for $\beta = 0.1$ Boltzmann selection with four different mutation rates, p_m =0, 0.001, 0.01 and 0.02.

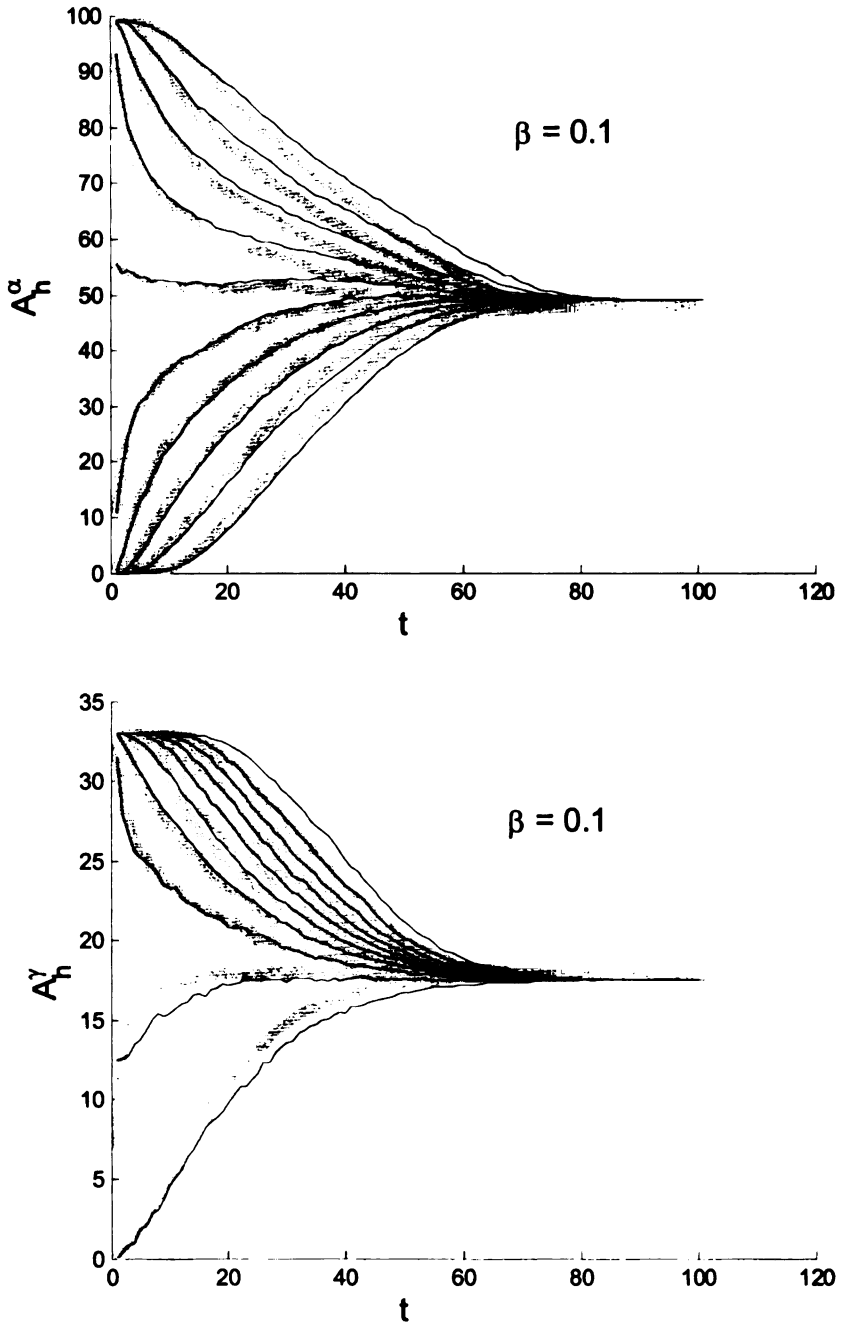


Figure 4.29 A_h^α and A_h^γ curves for selection with Boltzmann scaling. $\beta = 0.1$ p_c = "high enough", p_m = 0. Black lines are from GA run, gray lines are from model simulation

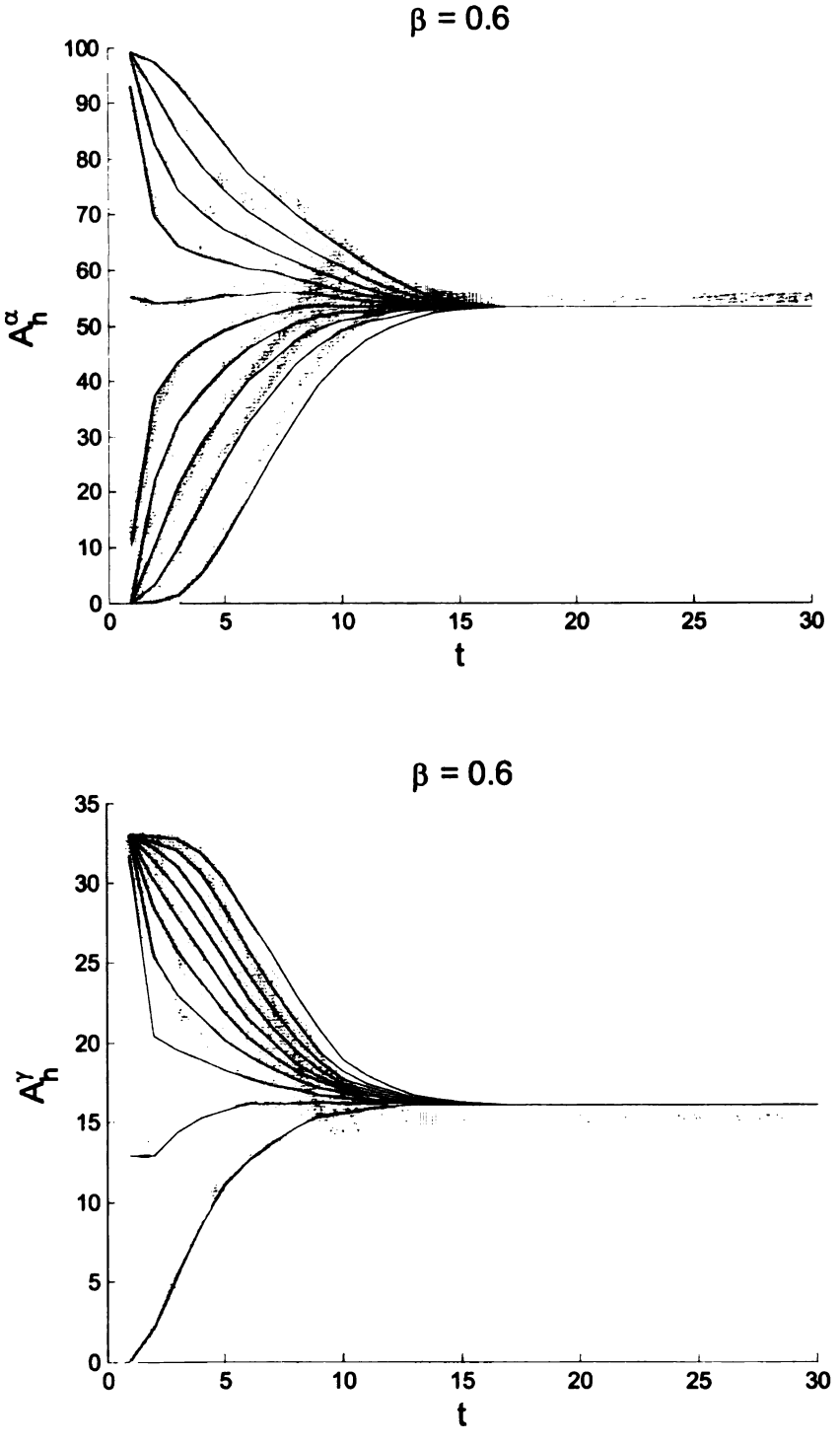


Figure 4.30 A_h^α and A_h^γ curves for selection with Boltzmann scaling. $\beta = 0.6$ p_c = "high enough", p_m = 0. Black lines are from GA run, gray lines are from model simulation

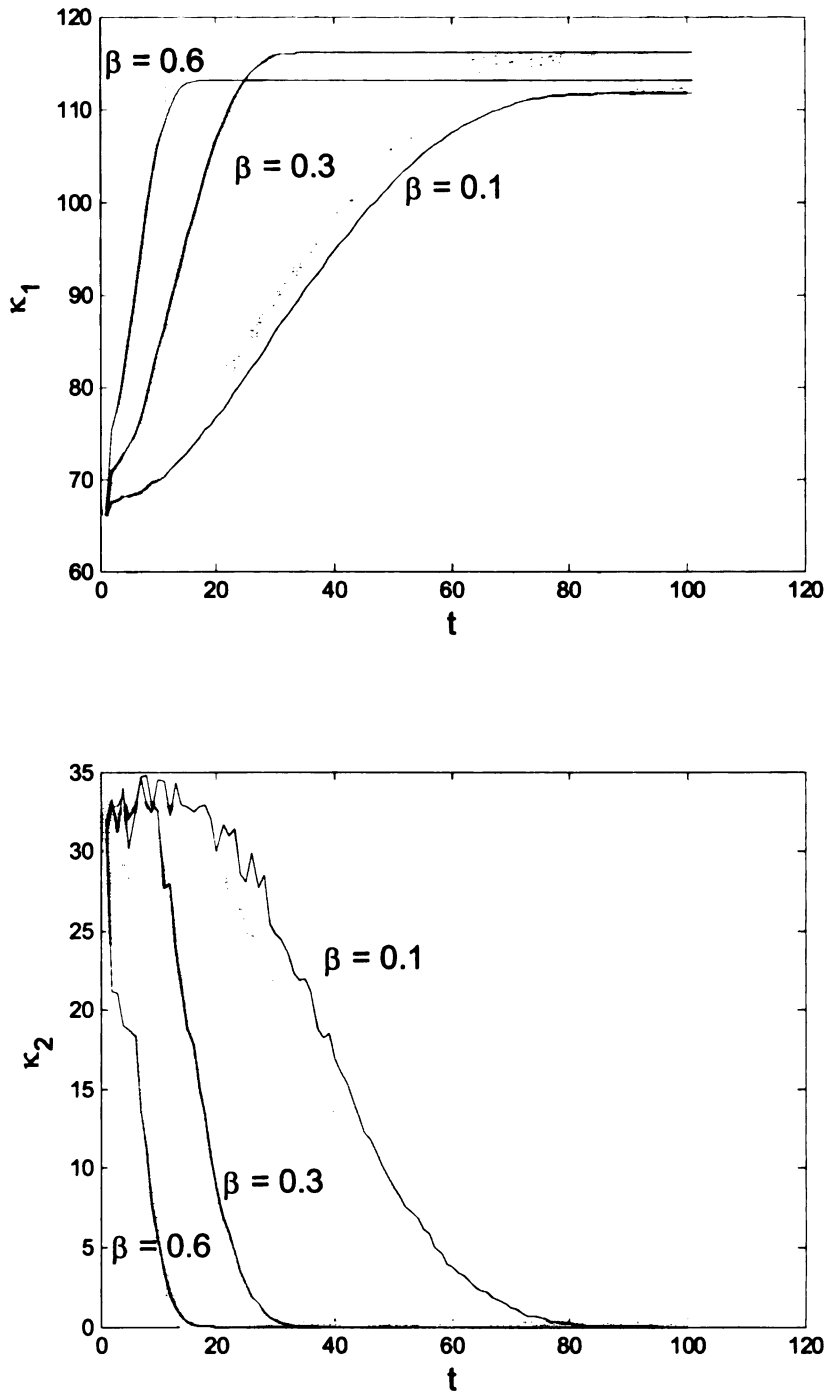


Figure 4.31 Comparison of mean fitness and fitness variance curves for selection with Boltzmann scaling for $\beta = 0.1$, 0.3 and 0.6. p_c = "high enough", p_m = 0.

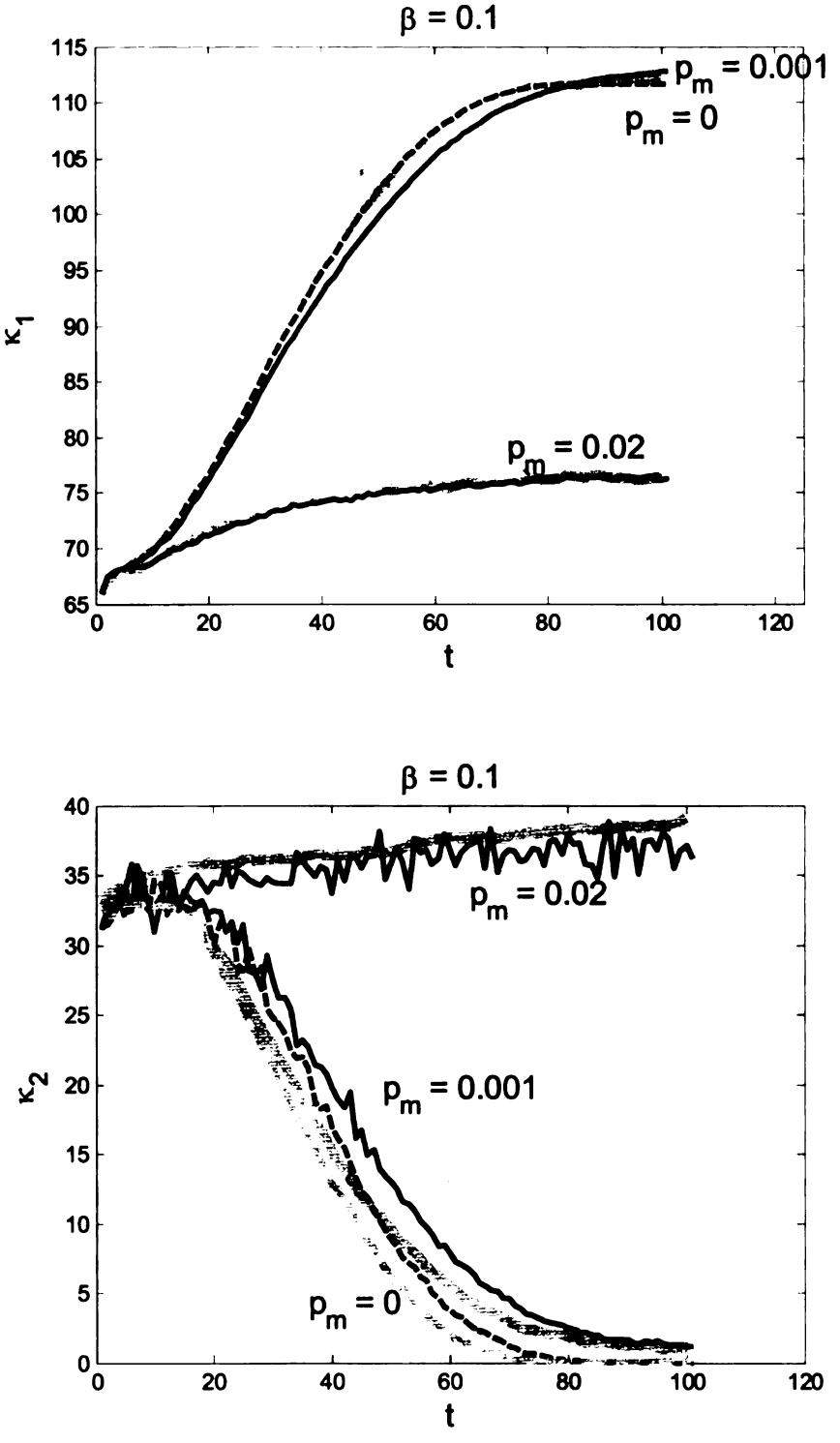


Figure 4.32 Comparison of mean fitness and fitness variance curves for selection with Boltzmann scaling for $\beta = 0.1$, p_c = "high enough", for different mutation rates

Chapter 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In this thesis, we have developed a new and very practical model for the GA dynamics of the OneMax problem and for a form of deceptive function problem that is described in Section 3.1. The model can be used to find the mean allele and, in the case of deceptive function, mean all-1 deceptive blocks values. Then, it predicts mean fitness and fitness variance of the population. Although the problems for which our model proved successful were rather simple or idealized, they were often sufficiently involved to capture interesting nontrivial features of the GA dynamics.

An attempt to develop a stochastic differential equation model to predict evolution of fitness distribution for the OneMax problem is presented in Appendix A. Although our attempt was not successful, it shows a strong connection between fitness evolution and certain diffusion equations and points toward the possibility of the existence of a diffusion-type equation that could describe the OneMax dynamics.

The model can be applied to these benchmark problems even when the crossover, mutation and the selection rates (in the case of Boltzmann scaling) are changed at predetermined generations during a GA run. Because of this capability of the model, it is unique, to the best of the author's knowledge, among the current models of the GA.

The method of building blocks for modeling parallel genetic algorithms is applied by Cantú-Paz [2001] in the case where the migration occurs only when all the populations are converged. Since our model estimates the mean value at each locus at any generation, it can be used to determine a suitable migration time as well as the migration rate for parallel genetic algorithms (in the island model case) when migrations are allowed at any generation for our benchmark problems.

The OneMax model, Diagram 1 (page 65), for modeling the case of typical crossover rates, uses some statistics of early generations of the GA in order to predict the rest of the evolution. The simulation results in Section 4.1 show that the model describes the GA dynamics for the OneMax problem very well for different crossover and mutation rates with fitness proportional selection. The correction weights introduced by Equation (2.20) are a new way of analyzing the crossover operator, and they work very well for two-point crossover, in our GA problem.

Note that our OneMax model for "high enough" crossover rates is covering a different case than the statistical mechanics model of Prügel-Bennett and

Shapiro [1994]. The maximum entropy assumption of Prügel-Bennett and Shapiro essentially models a situation in which the crossover operator is assumed to be effective enough to allow a relocation of the alleles which is probabilistically most likely to occur, under the constraints of the given mean fitness and fitness variance, when the alleles move freely. On the other hand, our model of "high enough" crossover rates does not assume any constraint in relation to how much the alleles can be mixed. The lower crossover rates are modeled relative to this extreme case using correction measures.

Extension of the OneMax model to the weighted fitness function is described in Section 2.3. The graphs of the model simulations, Section 4.1.3, show that the model is tested for weight Profiles A and B and the results are in excellent agreement with the GA results. The author believes that the model is effective for any weight profile.

Application of the OneMax model, modified as described in Section 2.4, to the Boltzmann selection gives very good estimations of this case for both the fixed and the scaled selection pressures, in the case of "high enough" crossover rates.

The model for the deceptive function, Diagram 2 (page 89), is quite powerful in predicting the dynamics for both fitness proportional and Boltzmann selections. This model is unique, to the best of the author's knowledge, in predicting the dynamics of a deceptive function to such an extent. It estimates the simultaneous

evolution of both the mean allele values and the mean all-1 deceptive blocks, as well as the mean fitness and fitness variance of the population.

5.2 Future Work

One immediate improvement needed in our OneMax model is to modify it to include the lower crossover rates with Boltzmann selection. It is observed by the author that the correction weights, as defined in Section 2.2.4, are not by themselves sufficient to predict the Boltzmann selection behavior since the μ-graphs, Figure 2.6, of this case change significantly as a function of time, unlike the fitness proportional selection case in which the statistics from generation 5, for example, would be sufficient to predict future behaviors.

Our model of the deceptive function also needs to be developed further to include lower crossover rates. In this case, one would need two sets of correction weights, one for modifying Equation (3.8) and the other for Equation (3.9). Then, the correction weights are going to be a function of two variables, α and γ.

The future work to improve the model would also include the estimation of the fitness variance for lower crossover rates, and an investigation of the predictive power of the model in the presence of external noise.

The next step in developing our model towards more complex fitness functions is to consider a fitness function which is made up several OneMax and deceptive parts as shown in Figure 5.1. Here, the chromosome has many

subdivisions, possibly overlapping, and the fitness of the chromosome is found by adding up the contribution of each part, which is found by applying either a (weighted) OneMax or a (weighted) deceptive function. It is speculated that a model for such a GA problem might be established by "pasting together" models developed in this thesis.

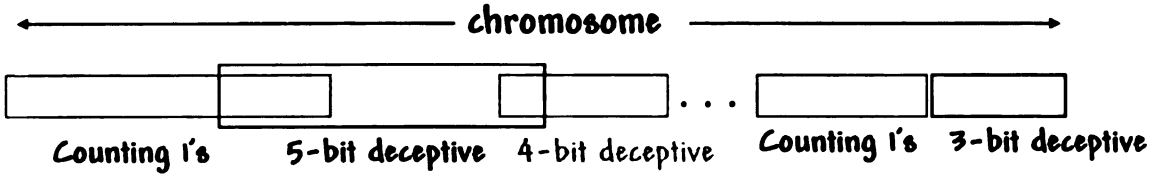


Figure 5.1 A complicated fitness function a composition of several OneMax and deceptive parts

Once a model is developed for a GA problem as described in Figure 5.1, the next step might be to consider a real-life problem and try to represent it with a fitness function that looks like Figure 5.1.

A different direction to follow in developing a model for complex real-life problems is as follows. Note that, when a "snapshot" of a GA is examined at some instant, from the point of view of the crossover + selection operators, in relation to determining the next mean allele (or mean all-1 deceptive block) values, any GA problem looks as if it is a weighted OneMax problem at that instant. In other words, any GA problem, in theory, can be seen as a series of weighted OneMax problems, the weights of which change dynamically from generation to generation. So, if one can estimate such a weighted OneMax decomposition of a real-life problem, it should be sufficient to predict the

dynamics the real-life problem by applying the OneMax model developed in this dissertation with fitness weights changing as a function of time.

APPENDIX

A COMPARISON OF THE ONEMAX PROBLEM WITH A DIFFUSION MODEL

In this section we will study the diffusion equation

$$\frac{\partial f(x,t)}{\partial t} = -\frac{\partial}{\partial x}[A_1(x)f(x,t)] + \frac{1}{2}\frac{\partial^2}{\partial x^2}[A_2(x)f(x,t)] \qquad \text{(A.1)}$$

where

$$A_1(x) = x\left(1 - \frac{1}{K}\ln x\right)\left[\alpha + \frac{\sigma^2}{2}\left(1 - \frac{1}{K}\ln x\right) - \frac{\sigma^2}{2K}\right] \qquad \text{(A.2)}$$

and

$$A_2(x) = \sigma^2 x^2\left(1 - \frac{1}{K}\ln x\right)^2 \qquad \text{(A.3)}$$

with initial condition

$$\lim_{t\to 0} f(x,t) = \delta(x - x_0) \qquad \text{(A.4)}$$

and explore its relationship to the evolution process of the one-max problem. The constants are chosen such that $\sigma, K > 0$ and $0 < \alpha < 1$.

Equation (A.1) appear in the study of population growth processes in random environments [Capocelli,Ricciardi,1975], [Ricciardi,1977, 1985]. If we consider the equation

$$\frac{dx}{dt} = \alpha x\left(1 - \frac{1}{K}\ln x\right) \tag{A.5}$$

as modeling a growth process with fertility rate α, then e^K becomes the asymptotic population size. Changing α into $\alpha + \Lambda(t)$, where $\Lambda(t)$ is a white noise with intensity $\frac{\sigma^2}{2}$, the resulting stochastic equation becomes

$$\frac{dx}{dt} = \alpha x\left(1 - \frac{1}{K}\ln x\right) + x\left(1 - \frac{1}{K}\ln x\right)\Lambda(t) \ . \tag{A.6}$$

Equation (A.1) is the Fokker-Plank equation, in other words the *forward equation*, for the transition probability density function, $f(x,t)$, of the stochastic process described by Equation (A.6). Here, $A_1(x)$ and $A_2(x)$ give the infinitesimal drift and variance of this diffusion process, respectively. From Equations (A.2) and (A.3) we see that the diffusion interval is $(0, e^K)$.

We can also write the so-called Kolmogorov or the *backward equation* for (A.6) as

$$\frac{\partial f(x,t|x_o)}{\partial t} = A_1(x_0)\frac{\partial f}{\partial x_0} + \frac{1}{2}A_2(x_0)\frac{\partial^2 f}{\partial x_0^2} \qquad \text{(A.7)}$$

In this equation, the initial condition, x_0, of the forward equation is considered as another variable of the function f. The solution of (A.7), or equivalently of (A.1), can be obtained by applying the transformation

$$\psi(t,x) = \int^{x} \frac{du}{u\left(1-\frac{1}{K}\ln u\right)} - \alpha t \ ,$$

$$\tilde{x} = \psi(t,x) \ ,$$

$$\tilde{x}_0 = \psi(t,x_0) \ ,$$

to Equation (A.7). Defining $\tilde{f}$ as

$$\tilde{f}(\tilde{x},t|\tilde{x}_0) = \left[\frac{\partial\psi(t,x)}{\partial x}\right]^{-1} f(x,t|x_0) \ , \qquad \text{(A.8)}$$

Equation (A.7) is transformed into the well known Wiener process described by the equation

$$\frac{\partial\tilde{f}(\tilde{x},t|\tilde{x}_0)}{\partial t} = \frac{\partial^2\tilde{f}}{\partial\tilde{x}_0^2}$$

with the solution in the form of a Gaussian function

$$\tilde{f}(\tilde{x},t|\tilde{x}_0)=(4\pi t)^{-1/2}\exp\left[-\frac{(\tilde{x}-\tilde{x}_0)^2}{4t}\right]. \qquad \text{(A.9)}$$

By (A.8) we obtain

$$f(x,t)=\frac{K}{2(K-\ln x)x\sqrt{\pi\gamma t}}\cdot\exp\left\{-\frac{\left[K\ln\left(\frac{K-\ln x_0}{K-\ln x}\right)-\alpha t\right]^2}{4\gamma t}\right\}. \qquad \text{(A.10)}$$

where x takes values between 0 and e^K.

Figure (A.1) shows the graph of this solution for different time values. The constants in f are taken as $\alpha=1.5, \sigma^2=0.4$, and $K=4.6052$. For any fixed value of t, say $t=t_1$, the function $f(x,t_1)$ gives the probability density function for the value of population after t_1 time units from the beginning. In particular

$$\int_{-\infty}^{+\infty} f(x,t_1)dx=\int_0^{e^K} f(x,t_1)dx=1.$$

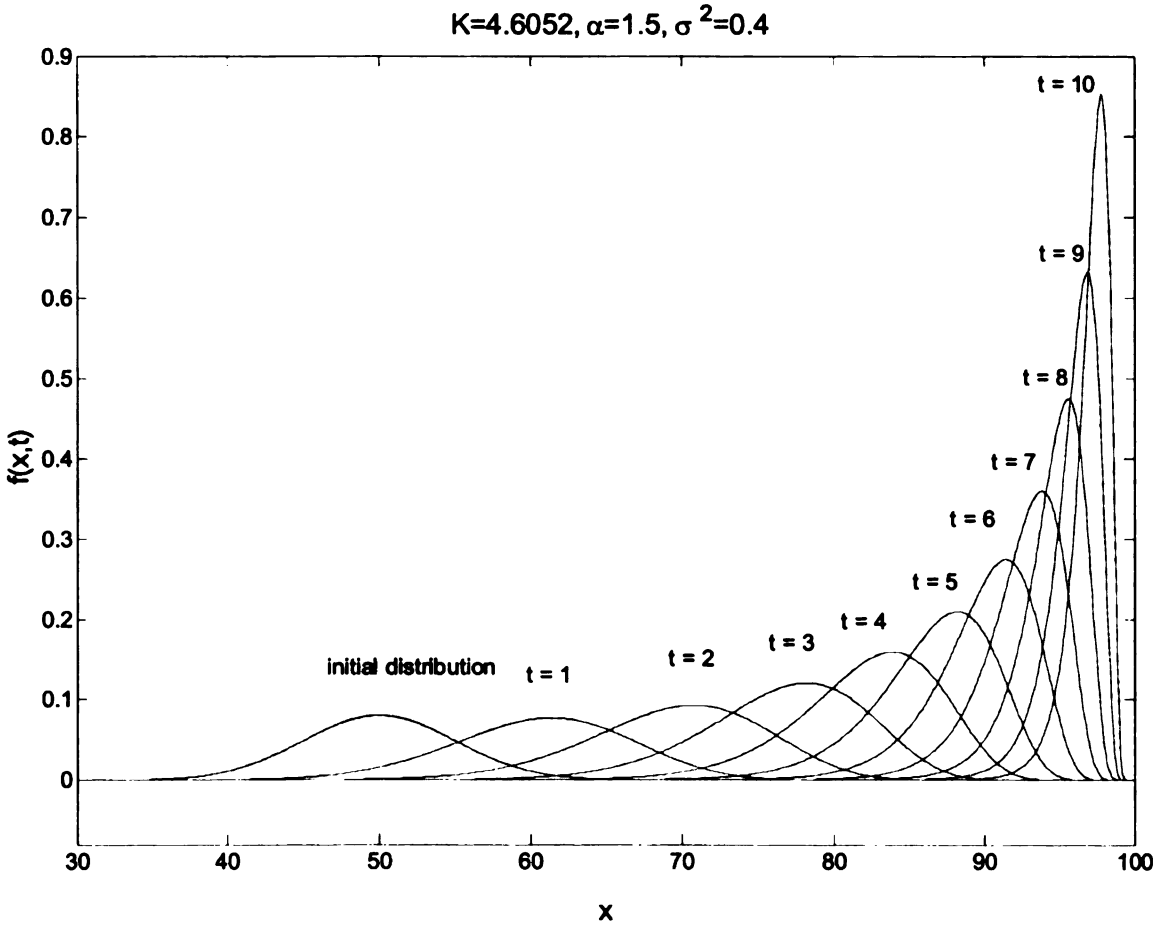


Figure A.1 Evolution of transition probability distribution, $f(x,t)$

On the other hand, in Figure (A.2) we see the evolution of fitness distribution for the OneMax problem with a population size of 50, (mutation rate 0.001 and Boltzmann selection with $\beta = 0.3$, in which two-point crossover is applied). The figure shows the fitness graphs for every second generation from 0, 180. For each generation the area under the curve remains equal to 1 as in the case of Figure (A.1). When we compare Figure (A.1) with Figure (A.2) we see a remarkable similarity between two figures.

In Figure (A.3), the initial distribution is chosen somewhat similar to the artificial migration case CASE2 of in Section 1.2.3. The solution of the differential equation for this initial condition is shown together with the solution with regular initial condition. The comparison shows a similar relationship as seen in Section 1.2.3 between CASE1 and CASE2.

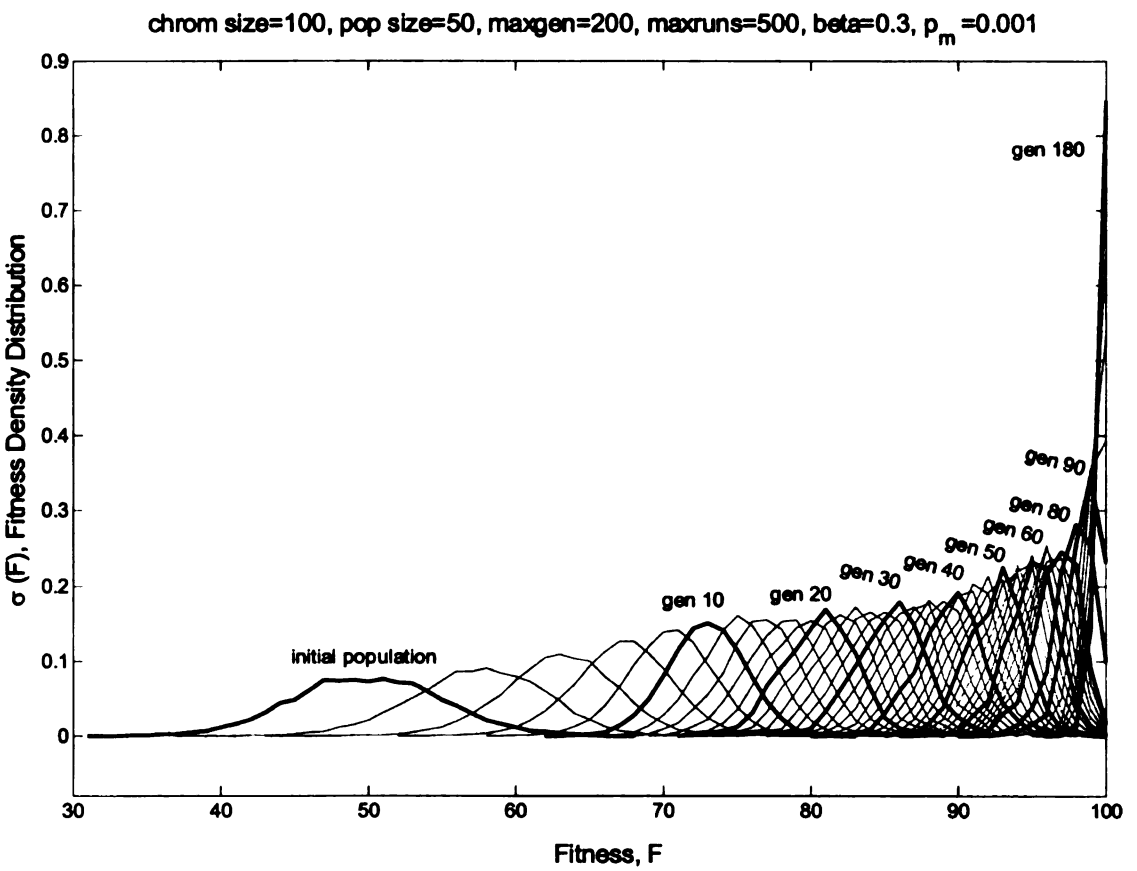


Figure A.2 Evolution of fitness distribution for 180 generations of OneMax problem

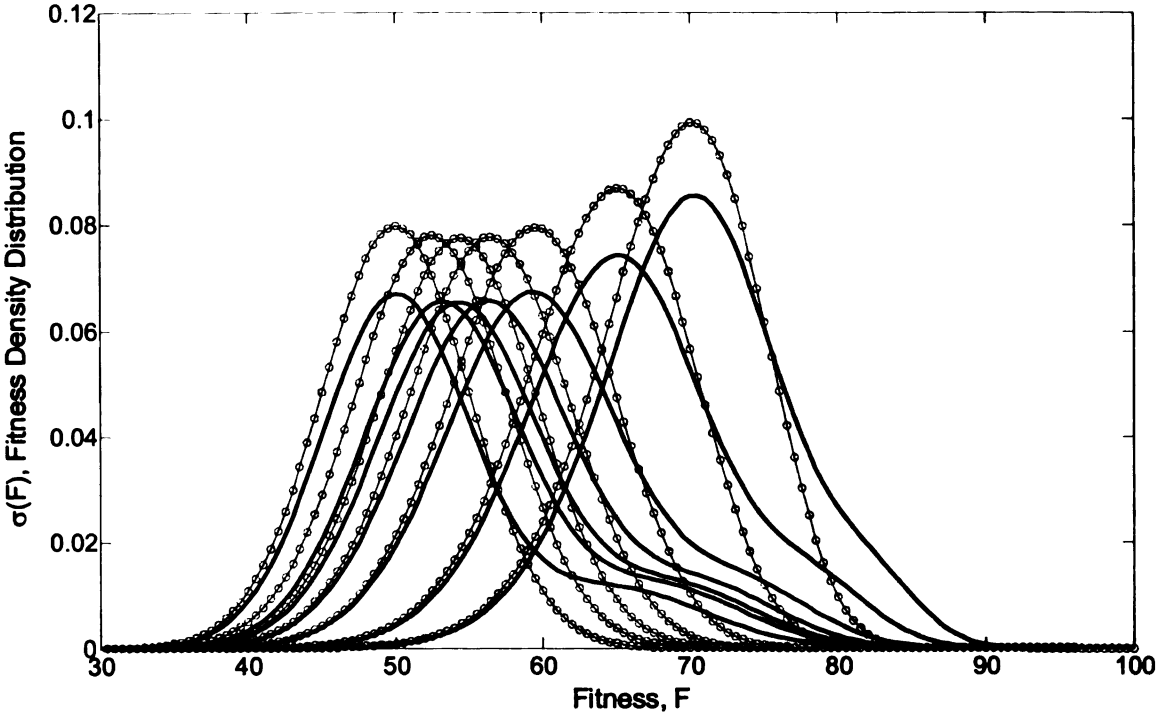


Figure A.3 Evolution of two transition probability distributions, $f(x,t)$. For the dark curves, initial condition is given to resemble a migration case

Although our attempts to determine suitable A_1 and A_2 and find an evolution equation in the form of Equation (A.1) for the fitness distribution failed, the similarity between two evolution graphs suggests that with a suitable modification to Equation (A.1), one might be able to obtain a differential equation which models the evolution of the fitness distribution for the one-max problem of genetic algorithms.

BIBLIOGRAPHY

Buyukbozkirli, B., Goodman, E.D., A statistical model of GA dynamics for OneMax and deceptive functions, *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO), Seattle (2004)

Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. *Kluwer Academic Publishers*, (2001)

Capocelli, R.M., Ricciardi L.M., A note on growth processes in random environments. *Biol. Cybernetics* 18 (1975), 105-109.

Furutani, H., Schema Analysis of OneMax Problem: Evolution equation for first order schemata. *Foundations of Genetic Algorithms* (FOGA) 7, (2002)

Goldberg D. E., Genetic Algorithms in Search. *Optimization & Machine Learning*, Addison Wesley (1989)

Goldberg, D.E., Deb, K., Thierens, D.: Toward a Better Understanding of Mixing in Genetic Algorithms. *Journal of the Society of Instrument and Control Engineers,* (1993), 32(1), 10-16

Goldberg, D.E.: The Design of Innovation. *Kluwer Academic Publishers, Boston, Dordrecht, London,* (2002)

Harik, G., Cantu-Paz, E., Goldberg, D., & Miller, B. L. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation,* 7(3), 231-253, (1999)

Holland J.H., Hierarchical descriptions of universal spaces and adaptive systems, *Technical Report ORA Projects 01252 and 08226*, University of Michigan, Department of Computer and Communication Sciences (1968)

Holland J.H., Adaptation in natural and artificial systems. *Ann Arbor: University of Michigan Press* (1975)

Nix, A.E., Vose, M.D.: Modeling Genetic Algorithms with Markov Chains. *Ann. Math. Art. Intell.,* (1991), 5:79-88

Ozpineci, B., Tolbert, L.M., Chiasson, J. N., Harmonic optimization of multilevel converters using genetic algorithms. *35th Annual IEEE Power Electronics Specialists Conference*, Aachen, Germany (2004)

Poli, R., Exact schema theory for genetic programming and variable length genetic algorithms with one point crossover. *Genetic Programming and Evolvable Machines,* 2(2): 123-163, (2001)

Prügel-Bennett, A., Shapiro, J.L.: An Analysis of Genetic Algorithms Using Statistical Mechanics. *Phys. Rev. Lett.,* (1994), 72(9):1305-1309

Prügel-Bennett, A., Shapiro, J.L.: The Dynamics of a Genetic Algorithm for Simple Random Ising Systems. *Physica D,* (1997), 104:75-114

Prügel-Bennett, A., Modeling finite populations. *Foundations of Genetic Algorithms* (FOGA) 7, (2002)

Ricciardi L.M., Diffusion Processes and Related Topics in Biology. *Lecture Notes in Biomathematics,* 14, Springer-Verlag (1977)

Ricciardi L.M., Stochastic Population Theory: Diffusion Processes. (1985)

Rattray, L.M.: Modeling the Dynamics of Genetic Algorithms Using Statistical Mechanics. *PhD thesis, University of Manchester*, Manchester, U.K., (1996)

Stephens C.R., Waelbroeck H., Schemata evolution and building blocks. *Evolutionary Computation*, 7(2):109-124, (1999)

Suzuki, J., A Markov chain analysis on simple genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 25 No.4, April (1995)

Vose, M.D., Liepins, G.E.: Punctuated Equilibria in Genetic Search. *Complex Systems*, (1991), 5:31-44

Vose, M. D., The Simple Genetic Algorithm, Foundations and Theory. *The MIT Press,* (1999)

Whitley D., A Genetic Algorithm Tutorial, Statistics and Computing, vol. 4, pages 65-85 (1994)

Wright, A. H., Rowe, J.E., Stephens C. R., Poli, R., Bistability in a Gene Pool GA with Mutation. *Foundations of Genetic Algorithms* 7, FOGA, (2002)

Wright, A., Poli, R., Stephens, C. , Langdon, W.B., and Pulavarty, S. An Estimation of Distribution Algorithm Based on Maximum Entropy. *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO), Seattle (2004)