8-1991

# Comparing finite and infinite population models of a genetic algorithm using the minimum deceptive problem

Allen Eugene Nix

To the Graduate Council:

I am submitting herewith a thesis written by Allen Eugene Nix entitled "Comparing finite and infinite population models of a genetic algorithm using the minimum deceptive problem." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Michael D. Vose, Major Professor

We have read this thesis and recommend its acceptance:

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Allen Eugene Nix entitled "Comparing Finite and Infinite Population Models of a Genetic Algorithm Using the Minimum Deceptive Problem". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Masters of Science, with a major in Computer Science.

Michael D. Vose

Michael D. Vose, Major Professor

We have read this thesis
and recommend its acceptance:

[illegible]

Jean R. S. Blair [illegible]

Accepted for the Council:

[illegible]

Associate Vice Chancellor
and Dean of the Graduate School

## STATEMENT OF PERMISSION TO USE

Signature [illegible]

Date 8 July 91

# COMPARING FINITE AND INFINITE POPULATION MODELS OF A GENETIC ALGORITHM USING THE MINIMUM DECEPTIVE PROBLEM

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Allen Eugene Nix

August 1991

# ACKNOWLEDGEMENTS

I would like to thank my advisor and senior committee member, Dr. Michael D Vose, without whose guidance and patience this thesis would not have been possible.

# ABSTRACT

Genetic algorithms (GAs) are general purpose algorithms designed to search irregular, poorly understood spaces. They are population based and use the ideas of evolution and survival of the fittest. For the finite population case, we model a genetic algorithm by representing the possible populations by the states of a Markov Chain. For the infinite population case, we use a model developed by Vose and Liepins [1]. We do not use previous models of GAs because they are incomplete in that they do not incorporate the effects of mutation which is a critical part of the evolutionary process. We consider the relationships between these models and an actual GA by investigating two minimal deceptive problems. The results of our computer simulations follow theoretical predictions and also reveal an unexpected effect of mutation on the deceptive problem.

# TABLE OF CONTENTS

# LIST OF FIGURES

Figure Page

# Chapter 1

# INTRODUCTION

Genetic algorithms (GAs) are general purpose algorithms designed to search irregular, poorly understood spaces. They are based on the idea of natural selection where the strongest survive to reproduce. Mating consists of two organisms contributing genes which make up chromosomes forming the genetic structure of the offspring. Occasionally, some of the genes will mutate producing genetic material in the offspring not contributed by either parent. Genes in a GA are represented by characters. For example, if the binary alphabet were used, a gene would be either 0 or 1, and a binary string would represent a chromosome.

In nature, the evolutionary process begins by elimination of weak organisms through competition, with the strongest surviving to pass genes on to their offspring. In a GA, the collection of organisms is represented by a collection of strings called the population. The elimination process is based on an objective function which gives fitness or strength to each string. The probability of each string being selected is given by normalizing the fitnesses to sum to 1. These probabilities are then used to select a string from the population for mating, hence strings with higher relative fitness are more likely to be selected.

In nature, the reproductive step consists of each parent donating genetic material to the genetic structure of the offspring. In a GA, this step is called crossover. It consists of breaking the two parent strings in the same random position and exchanging the portions of the strings to the left of the break point. Crossover is performed with some probability (the crossover rate), otherwise the children are taken to be the parents. One of the two offspring is kept and the other is discarded.

The next step is mutation which consists of some random change in the genes of the offspring. In a GA, this is implemented by changing each character in the string with some small probability (the mutation rate). The combination of crossover and mutation is referred to as recombination.

Each cycle of selection, crossover, and mutation produces one offspring for the next generation. Therefore, the cycle repeats using the same old population until the correct number of offspring have been attained to form the new population, and then the old population is discarded.

A GA is based on random choices and probabilities. Inherently, machines are not capable of making completely random choices, and they make round off errors in the calculations of probabilities. It is therefore possible that a GA when implemented on a computer would not behave as theoretically expected. We will investigate the accuracy of a GA implementation by comparing it with two mathematical models, one based on a finite population size, and one based on an infinite population. In the finite population case, we model a GA by representing the possible populations by the states of a Markov chain. We use the resultant steady state distribution to predict population distribution. For the infinite population case, we use a model developed by Vose and Liepins [1] which gives the probability of seeing each particular string for every generation.

We consider two types of objective functions; one that has been shown to make it hard for the GA to find the best string, and one that is known to make it easy [2]. We compare the results of a GA implementation with the results predicted by the mathematical models for both functions. Goldberg [2] originally investigated these functions but his model was not powerful enough to incorporate the effects of mutation. We are primarily interested in comparing a GA implementation, the Markov model, and the model of Vose and Liepins [1], and since these models can accommodate mutation, our investigation is based on a non-zero mutation rate.

Independently, T. Davis has also modeled a simple GA as a Markov Chain [8]. However, our work differs significantly from his. While he considers the asymptotics of steady state distributions as the mutation rate decreases, we investigate the asymptotics as population size increases. Moreover, our results concerning the matrix of transition probabilities are based on the model of Vose and Liepins, which simplifies representation and calculation.

In chapter 2, we use Markov chains to find a probability distribution which gives the expected proportions of populations that a GA should encounter if allowed to run an infinite amount of time. In chapter 3, we summarize a model developed by Vose and Liepins [1], called operator $\mathcal{G}$, which is used in the Markov model to find the probability of producing each string in the next generation given the current population. Chapter 4 describes the relationship between the Markov model and a GA and between the Markov model and operator $\mathcal{G}$. Chapter 5 outlines Goldberg's [2] two objective functions and the mathematical model he used to determine if finding the optimal string was easy or hard. Chapter 6 describes the results of simulations run for the Markov model, operator $\mathcal{G}$, and GAs and makes conclusions drawn from the results.

# Chapter 2

# MARKOV CHAIN DEVELOPMENT

We develop the Markov model by letting all possible populations represent the states of the Markov chain. We find a transition matrix that gives the probability of any given population being the next population based on the current population. We use the transition matrix to develop a vector that gives the probability of each population being encountered by a GA at generation $k$. We use the steady state distribution of the model to predict population behavior in a GA as $k \to \infty$.

Let $\Omega$ be the collection of length $\ell$ binary strings, and let $r = |\Omega| = 2^\ell$ be the number of possible strings. Let a population be a subset of $n$ strings of $\Omega$ where multiple instances of a string are allowed.

**Definition 1:** $N$ is the number of possible populations of strings where populations are numbered $0 \ldots N-1$. $Z$ is the $N \times r$ matrix where $z_{i,y}$ is the number of occurrences of string $y$ in the $i$ *th* population. (The numbers $y$ are identified with their binary representations.)

**Definition 2:** $\phi_i = < z_{i,0}, \ldots, z_{i,r-1} >$ is the $i$ *th* row of matrix $Z$ and represent the incidence vector for the *i th* population.

As an example, if $l = 2$, then the possible strings are $\{00, 01, 10, 11\}$. If $n = 2$, then

$$Z = \begin{bmatrix} 2000 \\ 1100 \\ 1010 \\ 1001 \\ 0200 \\ 0110 \\ 0101 \\ 0020 \\ 0011 \\ 0002 \end{bmatrix}, \ \phi_0 =< 2000 >, \text{ and} z_{0,0} = 2$$

.

**THEOREM 1**

There are

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

possible populations of strings.

Proof:

An incidence vector, $\phi_i$, can be represented graphically by using dots and slashes. Each dot represents one string, therefore a population of $n$ strings is represented by $n$ dots. To represent $z_{i,0}$ instances of string 0, a slash is put between the $z_{i,0}$ and the $z_{i,0}+1$ dot. To represent $z_{i,1}$ occurrences of string 1, a slash is put between the $z_{i,0}+z_{i,1}$ dot and the $z_{i,0}+z_{i,1}+1$ dot. Continuing in this way, a population of $n$ objects from $r$ types can be represented by $n$ dots and $r-1$ slashes. As an example, if $r = 4$ and $n = 5$ then the incidence vector $< 2, 0, 2, 1 >$ would be represented by

$$..//../.$$

where there are two of strings 0 and 2, one string 3, and none of string 1. If $r-1$ dots are added to the $n$ dots, then any population of size $n$ could be represented by appropriately choosing $r-1$ dots through which to put slashes. Since it is possible to represent all populations uniquely using this method, the number of possible populations, $N$, is just the number of ways of choosing $r-1$ dots from a total of $n+r-1$ dots. □

A Markov chain with the $N$ possible populations as states is used to model a genetic algorithm. The rows of matrix $Z$ describe the states of the model.

**Definition 3:** Random variable $Y_i$ is one member of the next population given that the current population is $\phi_i$, where $Y_i$ is an offspring resulting from selection and recombination of parents from population $i$.

**Definition 4:** $p_i(y)$ is the probability of producing the string $y$ in the next generation given that the current population is $\phi_i$. Thus $p_i$ is the probability density function for $Y_i$.

**Definition 5:** Random variable $\phi(k)$ for $k = 0, 1, \ldots$ is the incidence vector for the population at generation $k$.

**Definition 6:** $Q$ is the $N \times N$ transition matrix where $Q_{ij}$ is the probability that the $k$ *th* population will be $\phi_j$ given that the $k-1$ population is $\phi_i$.

**Definition 7:** $\pi(k) = < \pi_0(k), \ldots, \pi_{N-1}(k) >^T$ is a probability vector where $\pi_j(k)$ is the probability that the $k$ *th* generation is $\phi_j$.

**THEOREM 2**

$$\pi(k) = (Q^T)^k \, \pi(0) \tag{2.1}$$

where $\pi(0)$ is the probability vector for the initial population.

Proof by induction on $k$:

A) Base:

If $k = 0$ then

$$\pi(0) = (Q^T)^0 \, \pi(0)$$

Since $(Q^T)^0$ is the identity matrix, the base is established.

B) Induction:

The proof of the induction step relies on

$$\pi(k) = \left(Q^T\right) \, \pi(k-1) \tag{2.2}$$

That is

$$\pi_j(k) = \sum_{i=0}^{N-1} Q_{ij} \, \pi_i(k-1) \tag{2.3}$$

for $0 \leq j \leq N-1$.

Recall from definition 6 that $Q_{ij}$ was defined as the conditional probability

$$Q_{ij} = p\{\phi(k) = \phi_j \mid \phi(k-1) = \phi_i\}$$

where $\phi(k)$ represents the population at generation $k$. Recall from definition 7 that $\pi_i(k-1)$ was defined as

$$\pi_i(k-1) = p\{\phi(k-1) = \phi_i\}$$

By the definition of conditional probability

$$Q_{ij}\,\pi_i(k-1) = \frac{p\{\phi(k) = \phi_j \,\wedge\, \phi(k-1) = \phi_i\}}{p\{\phi(k-1) = \phi_i\}}\,p\{\phi(k-1) = \phi_i\}$$

$$= p\{\phi(k) = \phi_j \,\wedge\, \phi(k-1) = \phi_i\}$$

Substituting this into the right hand side of equation (2.3) gives

$$\sum_{i=0}^{N-1} p\{\phi(k) = \phi_j \,\wedge\, \phi(k-1) = \phi_i\}.$$

Now suppose a set of events $A_0, \ldots, A_{N-1}$ are such that $p\{A_0 \vee \cdots \vee A_{N-1}\} = 1$ and $p\{A_i \wedge A_j\} = 0$ if $i \neq j$. Then for any event $B$

$$p(B) = \sum_{i=0}^{N-1} p(B \,\wedge\, A_i)$$

Substituting $\phi(k) = \phi_j$ for $B$ and $\phi(k-1) = \phi_i$ for $A_i$, gives

$$p\{\phi(k) = \phi_j\} = \sum_{i=0}^{N-1} p\{\phi(k) = \phi_j \,\wedge\, \phi(k-1) = \phi_i\}$$

Since $p\{\phi(k) = \phi_j\} = \pi_j(k)$ by definition, equation (2.3) is established.

Substituting the inductive hypothesis

$$\pi(k-1) = (Q^T)^{k-1}\,\pi(0)$$

for $\pi(k-1)$ in equation (2.2) gives

$$\pi(k) = Q^T\,(Q^T)^{k-1}\,\pi(0)$$

$$= (Q^T)^k\,\pi(0) \tag{2.4}$$

Therefore equation (2.1) is established. □

Let

$$\pi = \lim_{k \to \infty} \pi(k)$$

if the limit exists.

Expression (2.4) can be substituted into the right hand side to give

$$\begin{aligned}
\pi &= \lim_{k \to \infty} (Q^T)^k \, \pi(0) \\
&= \lim_{k \to \infty} Q^T (Q^T)^{k-1} \, \pi(0) \\
&= Q^T \{ \lim_{k \to \infty} (Q^T)^{k-1} \, \pi(0) \} \\
&= Q^T \{ \lim_{k \to \infty} (Q^T)^{k} \, \pi(0) \} \\
&= Q^T \, \pi
\end{aligned}$$

Therefore if $\pi$ exists, it satisfies $Q^T \pi = \pi$ and $\sum_{j=0}^{N-1} \pi_j = 1$.

If some power of $Q$ has only positive entries, then $\lim_{k \to \infty} \pi(k)$ exists [4]. This corresponds to our situation since we will later observe that a nonzero mutation rate implies every entry of $Q$ is positive. To solve for $\pi$, routines from EISPACK [6], [7] are used.

There are $n$ members in a population, therefore $0 \leq z_{j,0} \leq n$. The number of ways of choosing $z_{j,0}$ occurrences of the string 0 for a population of size $n$ is given by the binomial coefficient

$$\binom{n}{z_{j,0}}$$

There are $n - z_{j,0}$ positions remaining to fill, so the possible combinations for string 1 are given by

$$\binom{n - z_{j,0}}{z_{j,1}}$$

Continuing in this way, the total possible combinations for all strings would be given by

$$\binom{n}{z_{j,0}}\binom{n-z_{j,0}}{z_{j,1}}\cdots\binom{n-z_{j,0}-z_{j,1}-\cdots-z_{j,r-2}}{z_{j,r-1}}$$

$$=\frac{n!}{(n-z_{j,0})!z_{j,0}!}\frac{(n-z_{j,0})!}{(n-z_{j,0}-z_{j,1})!z_{j,1}!}\cdots\frac{(n-z_{j,0}-z_{j,1}-\cdots-z_{j,r-2})!}{(n-z_{j,0}-z_{j,1}-\cdots-z_{j,r-1})!z_{j,r-1}!}$$

$$=\frac{n!}{z_{j,0}!\,z_{j,1}!\cdots z_{j,r-1}!}$$

If the next generation is $\phi_j$, then for each $y$ there must be $z_{j,y}$ occurrences of string $y$ produced. The probability, given events are independent, is almost given by

$$\prod_{y=0}^{r-1}\{p_i(y)\}^{z_{j,y}}$$

but it must be remembered that exactly which $z_{j,y}$ members of the next population are $y$ is not important, only that the correct number of strings occur.

Therefore, the probability that the next generation is $\phi_j$, given that the current generation is $\phi_i$, can be written as

$$Q_{i,j}=\frac{n!}{z_{j,0}!\,z_{j,1}!\cdots z_{j,r-1}!}\prod_{y=0}^{r-1}p_i(y)^{z_{j,y}}$$

$$=n!\prod_{y=0}^{r-1}\frac{p_i(y)^{z_{j,y}}}{z_{j,y}!} \qquad (2.5)$$

Therefore, if expression (2.5) is indeed the probability of the next generation being $\phi_j$ given that the current population is $\phi_i$ then, since probabilities sum to 1,

$$1=\sum_{\substack{\phi_j\\|\phi_j|=n}} n!\prod_{y=0}^{r-1}\frac{p_i(y)^{z_{j,y}}}{z_{j,y}!} \qquad (2.6)$$

where $|\phi_j|=\sum z_{j,y}$.

To prove that expression (2.5) gives a probability distribution, it is sufficient to show

**THEOREM 3**

For all probability distributions $p$ on the integers from 0 to $x$, and for all nonnegative integers $v \le n$ where $n$ is the population size

$$\frac{n!}{(n-v)!} = \sum_{\substack{\phi_j \\ |\phi_j| = n-v}} n! \prod_{y=0}^{x} \frac{p(y)^{z_{j,y}}}{z_{j,y}!} \tag{2.7}$$

Proof by induction on $x$:

A) Base:

To establish the base let $x = 0$, let $\phi_j = < z_{j,0} >$, and let $p(0) = 1$. Substituting these values in equation (2.7) gives

$$\sum_{z_{j,0}=n-v} n! \frac{1^{z_{j,0}}}{z_{j,0}!} = n! \frac{1^{n-v}}{(n-v)!} = \frac{n!}{(n-v)!}$$

so the base is true.

B) Induction:

Let $\phi_j = < z_{j,0}, \ldots, z_{j,x-1}, z_{j,x} >$ and $\phi'_j = < z_{j,0}, \ldots, z_{j,x-1} >$.

Note that since $p(0) + \cdots + p(x-1) + p(x) = 1$ then

$$\frac{p(0)}{1-p(x)} + \cdots + \frac{p(x-1)}{1-p(x)} = 1$$

Letting $p'(y) = \frac{p(y)}{1-p(x)}$ gives $p'(0) + \cdots + p'(x-1) = 1$ and $p(y) = p'(y)\,(1 - p(x))$. Using this relationship we have

$$\sum_{\substack{\phi_j \\ |\phi_j| = n-v}} n! \prod_{y=0}^{x} \frac{p(y)^{z_{j,y}}}{z_{j,y}!} = \sum_{\substack{\phi_j \\ |\phi_j| = n-v}} n! \prod_{y=0}^{x} \frac{p'(y)^{z_{j,y}}}{z_{j,y}!} (1 - p(x))^{z_{j,y}}$$

$$= \sum_{\substack{\phi_j \\ |\phi_j| = n-v}} n! \prod_{y=0}^{x} \frac{p'(y)^{z_{j,y}}}{z_{j,y}!} \prod_{y=0}^{x} (1 - p(x))^{z_{j,y}} \qquad (2.8)$$

The last product of this expression can be rewritten as

$$\prod_{y=0}^{x} (1 - p(x))^{z_{j,y}} = (1 - p(x))^{\sum z_{j,y}} = (1 - p(x))^{|\phi_j|} = (1 - p(x))^{n-v}$$

Substituting this into expression (2.8) gives

$$\sum_{\substack{\phi_j \\ |\phi_j| = n-v}} n! \left\{ \prod_{y=0}^{x} \frac{p'(y)^{z_{j,y}}}{z_{j,y}!} \right\} (1 - p(x))^{n-v}$$

By breaking the sum across all $\phi$ according to the value $z_{j,x}$, this can be written as

$$= \sum_{z_{j,x}=0}^{n-v} (1 - p(x))^{n-v} \sum_{\substack{\phi'_j \\ |\phi'_j| = n-v-z_{j,x}}} n! \prod_{y=0}^{x} \frac{p'(y)^{z_{j,y}}}{z_{j,y}!}$$

$$= \sum_{z_{j,x}=0}^{n-v} (1 - p(x))^{n-v} \sum_{\substack{\phi'_j \\ |\phi'_j| = n-v-z_{j,x}}} n! \left( \prod_{y=0}^{x-1} \frac{p'(y)^{z_{j,y}}}{z_{j,y}!} \right) \frac{p'(x)^{z_{j,x}}}{z_{j,x}!}$$

$$= \sum_{z_{j,x}=0}^{n-v} \frac{(1-p(x))^{n-v}}{z_{j,x}!} \left(\frac{p(x)}{1-p(x)}\right)^{z_{j,x}} \sum_{\substack{\phi'_j \\ |\phi'_j| = n-v-z_{j,x}}} n! \prod_{y=0}^{x-1} \frac{p'(y)^{z_{j,y}}}{z_{j,y}!}$$

$$= \sum_{z_{j,x}=0}^{n-v} (1-p(x))^{n-v-z_{j,x}}\, p(x)^{z_{j,x}} \frac{1}{z_{j,x}!} \sum_{\substack{\phi'_j \\ |\phi'_j| = n-v-z_{j,x}}} n! \prod_{y=0}^{x-1} \frac{p'(y)^{z_{j,y}}}{z_{j,y}!} \qquad (2.9)$$

Substituting $v = v + z_{j,x}$ into the inductive hypothesis (equation (2.7)) gives

$$\frac{n!}{(n-v-z_{j,x})!} = \sum_{\substack{\phi'_j \\ |\phi'_j| = n-v-z_{j,x}}} n! \prod_{y=0}^{x-1} \frac{p'(y)^{z_{j,y}}}{z_{j,y}!}$$

Substituting the left hand side of this equation for the right hand sum of (2.9) gives

$$\sum_{z_{j,x}=0}^{n-v} (1-p(x))^{n-v-z_{j,x}}\, p(x)^{z_{j,x}} \frac{n!}{z_{j,x}!\,(n-v-z_{j,x})!} \qquad (2.10)$$

Note that

$$\frac{n!}{z_{j,x}!\,(n-v-z_{j,x})!} = \frac{n!\,(n-v)!}{(n-v)!\,z_{j,x}!(n-v-z_{j,x})!} = \binom{n-v}{z_{j,x}} \frac{n!}{(n-v)!}$$

Substituting this result into (2.10) gives

$$\frac{n!}{(n-v)!} \sum_{z_{j,x}=0}^{n-v} \binom{n-v}{z_{j,x}} (1-p(x))^{n-v-z_{j,x}}\, p(x)^{z_{j,x}} \qquad (2.11)$$

which by the binomial theorem is

$$\frac{n!}{(n-v)!}(1 - p(x) + p(x))^{n-v}$$

$$= \frac{n!}{(n-v)!}$$

□

If $v = 0$ and $x = r - 1$, then as a special case of this theorem we have

$$1 = \sum_{\substack{\phi_j \\ |\phi_j| = n}} n! \prod_{y=0}^{r-1} \frac{p_i(y)^{z_{j,y}}}{z_{j,y}!}$$

To complete the model, the conditional probability function $p_i(y)$ which gives the probability of producing string $y$ from population $\phi_i$ must be calculated. This function was developed by Vose and Liepins [1]. A summary of their model is presented in the following section on operator $\mathcal{G}$.

Chapter 3

# OPERATOR $\mathcal{G}$

The conditional probability function $p_i(y)$ is related to the operator $\mathcal{G}$ in the model developed by Vose and Liepins [1]. Theirs is a mathematical model of a genetic algorithm based on an infinite population of fixed length strings which are selected with probability proportional to relative fitness and recombined using crossover and mutation. The methods used are reproduced here without proofs.

Given a vector $x$, let $|x|$ denote the sum of its coordinates. Let the operator $\oplus$ be exclusive-or on integers and let the operator $\otimes$ be logical-and.

**DEFINITION 1**: $F$ is the $r \times r$ nonnegative diagonal matrix with $i, i\ th$ entry $f(i)$, where $f$ is the objective function that assigns fitnesses $f(i)$ to string $i$.

**DEFINITION 2**: $s^{k-1} = < s_0, \ldots, s_{r-1} >$ is the vector representing the probabilities of the strings in the $k-1$ generation being selected as parents where $s_a$ is the probability that string $a$ is selected.

The probability of string $y$ being in the next generation is

$$\sum_{a,b} p\{a \text{ is a parent}\}\, p\{b \text{ is a parent}\}\, p\{y \text{ is a child of } a \text{ and } b\} \tag{3.1}$$

Since $\phi_i \mid \phi_i \mid^{-1}$ is a vector with the $y$ *th* component equal to the proportion of $y$ in population $i$, $F\phi_i$ is a vector pointing in the same direction as the vector having $y$ *th* component equal to the probability that $y$ will be selected for recombination. Therefore

$$p\{a \,\text{is a parent}\} = s_a^{k-1} = \left(\frac{F\phi_i}{\mid F\phi_i \mid}\right)_a \tag{3.2}$$

To develop $p\{y \,\text{is a child of}\, a \,\text{and}\, b\}$, let $r_{a,b}(y)$ be the probability that $y$ results from the recombination of parents $a$ and $b$. Note that $r_{a,b}(y) = r_{a\oplus y, b\oplus y}(0)$. Using this relation and equation (3.2) we can rewrite the sum in (3.1) as

$$\sum_{a,b} s_a^{k-1}\, s_b^{k-1}\, r_{a,b}(y)$$

$$= \sum_{a\oplus y, b\oplus y} s_{a\oplus y}^{k-1}\, s_{b\oplus y}^{k-1} r_{a,b}(0) \tag{3.3}$$

Define permutations $\sigma_j$ by

$$\sigma_j < s_0, \ldots, s_{N-1} >^T = < s_{0\oplus j}, \ldots, s_{(N-1)\oplus j} >^T$$

where vectors are regarded as columns, and $T$ denotes transpose. Using this definition, the sum (3.3) can be rewritten to yield

$$p\{y \,\text{is in the next generation}\} = \sum_{a,b} (\sigma_y\, s^{k-1})_a\, (\sigma_y\, s^{k-1})_b\, r_{a,b}(0) \tag{3.4}$$

If we let $M$ be the matrix with $a$, $b$ *th* entry $r_{a,b}(0)$, then $M_{a,b}$ will be the probability that 0 results from the recombination process based on parents $a$ and $b$. By letting the crossover rate be $\chi$ and the mutation rate be $\mu$, we can derive an explicit formula for $M_{a,b}$ from the following considerations:

1. The probability that 0 results from parents $a$ and $b$ depends on the probability that mutation changes the 1's occurring in the results of crossover to 0 and leaves the other bits alone.

2. The number of 1's occurring in the results produced by crossing $a$ and $b$ at position $y$ are given by $\mid a \mid - \Delta_{a,b,y}$ and $\mid b \mid + \Delta_{a,b,y}$ where

$$\Delta_{a,b,y} = |(2^y - 1) \otimes a| - |(2^y - 1) \otimes b|$$

3. The probability of changing a specified collection of $b$ bits (in a length $\ell$ binary string) via mutation is $(1-\mu)^{\ell-b}\mu^b$.

These three observations lead to

$$M_{a,b} = \frac{\chi}{\ell-1}\sum_{y=1}^{\ell-1}\left\{\frac{\mu^{|a|-\Delta_{a,b,y}}\,(1-\mu)^{\ell-|a|+\Delta_{a,b,y}}}{2} + \frac{\mu^{|b|+\Delta_{a,b,y}}\,(1-\mu)^{\ell-|b|-\Delta_{a,b,y}}}{2}\right\}$$
$$+ (1-\chi)\left(\frac{\mu^{|a|}\,(1-\mu)^{\ell-|a|}}{2} + \frac{\mu^{|b|}\,(1-\mu)^{\ell-|b|}}{2}\right)$$

$$= \frac{(1-\mu)^\ell}{2}\left\{\eta^{|a|}\left(1-\chi+\frac{\chi}{\ell-1}\sum_{y=1}^{\ell-1}\eta^{-\Delta_{a,b,y}}\right) + \eta^{|b|}\left(1-\chi+\frac{\chi}{\ell-1}\sum_{y=1}^{\ell-1}\eta^{\Delta_{a,b,y}}\right)\right\}$$

where $\eta = \mu/(1-\mu)$ and division by zero at $\mu = 0$ and $\mu = 1$ is to be removed by continuity.

Define the operator $\mathcal{M}$ by

$$\mathcal{M}(x) = <(\sigma_0\, x)^T M \sigma_0\, x, \ldots, (\sigma_{r-1}\, x)^T M \sigma_{r-1}\, x>^T$$

Then equation (3.4) can be rewritten as

$$p\{y \text{ is in the next generation}\} = \left(\mathcal{M}\frac{F\phi_i}{\mid F\phi_i \mid}\right)_y$$

Now we can define operator $\mathcal{G}$ by

$$\mathcal{G}(\phi_i) = \left(\mathcal{M}\frac{F\phi_i}{\mid F\phi_i \mid}\right)$$

Therefore if the current population is $\phi_i$, then the expected next population can be described as:

$$\mathcal{G}(\phi_i)$$

The $y\ th$ component of $\mathcal{G}(\phi_i)$ is the expected proportion of string $y$ in the next generation given that the current population is $\phi_i$. Since the expected proportion is equal to the probability of occurrence, we have

$$p_i(y) = (\mathcal{G}(\phi_i))_y$$

Note that if the mutation rate is nonzero, there is a positive probability of any string mutating into any other string. Hence the matrix $M$ is positive in this case. It follows that the system of quadratic forms represented by operator $\mathcal{M}$ is positive definite and hence $\mathcal{G}(\phi_i)$ has positive coordinates since $F$ has nonzero diagonal entries. Therefore the transition matrix $Q$ of the Markov model has positive entries and its steady state distribution vector $\pi$ exists.

Let the populations encountered be regarded as points on the simplex

$$\Lambda \ = \ \{x \in \Re^r \ : \ x_j \ \geq \ 0 \text{ and } \mid x \mid = 1\}$$

through the correspondence $\phi_i \leftrightarrow \phi_i \mid \phi_i \mid^{-1} \in \Lambda$. The sequence of populations $\phi_i$, $\mathcal{G}(\phi_i)$, $\mathcal{G}(\mathcal{G}(\phi_i))$, ... quickly converges along a simple trajectory in $\Lambda$ to a fixed point of $\mathcal{G}$. Presently, it has not been proven that convergence always occurs, but this is conjectured by Vose and Liepins and is supported by several simulations. It has been shown that operator $\mathcal{G}$ can have one or more fixed points. If only one fixed point exists, then the initial population is of no consequence. If more than one fixed point exists, then the initial population determines which convergence path is taken.

Chapter 4

# RELATING THE MARKOV MODEL TO A GA AND THE OPERATOR $\mathcal{G}$

We relate the Markov model to a GA by developing summary vectors. This vector for the GA represents the average number of occurrences of each string through generation $k$. The vector for the Markov model represents the expected average as $k \to \infty$. Operator $\mathcal{G}$ also gives expected string averages, but for an infinite population. We let the population size $n$ of the Markov model become large to relate the Markov model to operator $\mathcal{G}$.

The vector

$$s(k) = \frac{1}{nk} \sum_{i=0}^{N-1} \phi_i \; c(k)_i$$

has $y$ *th* component equal to the proportion of the time string $y$ was encountered in all populations through generation $k$, where $n$ is the population size, $c(k)$ is a vector having $i$ *th* component equal to the number of occurrences of population $i$ through generation $k$, and $\phi_i$ is the incidence vector for population $i$.

The Markov model behaves exactly like a genetic algorithm, but its steady state distribution models population behavior as $k \to \infty$. That is, the steady state distribution is given by

$$\lim_{k\to\infty} \pi(k) \;=\; \lim_{k\to\infty} \pi(0)\, Q^k \;=\; \text{the solution to the equation } \; \pi = \pi\, Q$$

where $\pi(0)$ is the vector describing the probability of each population at generation 0, and $Q$ is the transition matrix. The $j\ th$ component of $\pi$ is the relative proportion of time that the $j\ th$ population occurs.

Therefore since

$$\lim_{k\to\infty} \frac{c(k)_i}{k} \;=\; \pi_i$$

we have

$$\begin{aligned} \lim_{k\to\infty}\ s(k) \;&=\; \frac{1}{n} \sum_{i=0}^{N-1} \phi_i \lim_{k\to\infty} \frac{c(k)_i}{k} \\ &=\; \frac{1}{n} \sum_{i=0}^{N-1} \phi_i\, \pi_i \;=\; s \end{aligned}$$

which is the expected population with respect to the steady state distribution of the Markov model.

To relate the Markov model to $\mathcal{G}$, we visualize populations as points in the space $\Lambda$. The successive populations move around in $\Lambda$ under the influence of selection, crossover, and mutation. Since the expected population, $s$, of the Markov model represents an average over an infinite number of generations and since a fixed point of $\mathcal{G}$ also represents a similar average, it might be expected that the Markov model would give high probability to populations near a fixed point of $\mathcal{G}$.

Nix and Vose [3] have shown that as the population size $n$ approaches infinity, the steady state distributions only have limits which give positive probability only to the fixed points of $\mathcal{G}$. Therefore as the population size increases, the proportions of strings encountered in either a GA or the Markov model converge to the fixed point of $\mathcal{G}$ when only one fixed point exists. If more than one fixed point exists, then the populations seen are less predictable because of unknown variables such as the number of fixed points and how strong an attractor each fixed point is.

# Chapter 5

# THE MINIMUM DECEPTIVE PROBLEM

Minimum deceptive problems have been classified by Goldberg into two types according to whether they make it easy or hard for a GA to find the optimal string [2]. We use two minimum deceptive problems, one of each type, to investigate the Markov model. In this chapter, we summarize the problems and the mathematical model that Goldberg used to make his classification of minimum deceptive problems.

**DEFINITION 1:** A schema is a sequence of 0s, 1s, and *s representing the set of all strings which match the sequence, where * matches both 0 and 1. For example, schema $1*$ represents the strings 10 and 11.

**DEFINITION 2:** Let $f_i$ represent the fitness of string $i$ where strings are identified with their binary representation. Define the utility of a schema as the average fitness of all strings represented by the schema. Define different schema to be competing when they have the same fixed positions. As an example, $f_{1*}$ is the utility of schema $1*$ and equals the average fitness of strings 10 and 11. Since we are using the binary alphabet, the only competing schema for $1*$ is $0*$.

**DEFINITION 3:** Optimal strings are defined as strings having fitness equal to the global optimum, suboptimal strings have lesser fitness.

**DEFINITION 4:** A problem is deceptive when the optimal string $x \in S_1$ and the utility of $S_2$ > the utility of $S_1$ for the same two competing schema $S_1$ and $S_2$.

The smallest string length where deception is possible is two. Assuming the string length is two and that string 11 is optimal, there are four possible schemata containing 1 fixed position; two contain suboptimal strings (0$*$, $*$0) and two contain the optimal string (1$*$, $*$1). By the definition of deception, either

$$f_{0*} = \frac{f_{00} + f_{01}}{2} > \frac{f_{10} + f_{11}}{2} = f_{1*} \tag{5.1}$$

or

$$f_{*0} = \frac{f_{00} + f_{10}}{2} > \frac{f_{01} + f_{11}}{2} = f_{*1} \tag{5.2}$$

must be true. Without loss of generality, we assume that equation (5.1) is true so that $f_{0*} > f_{1*}$ is the deceptive condition.

The fitness can be normalized with respect to the complement of the global optimum to give

$$r = \frac{f_{11}}{f_{00}} \quad c = \frac{f_{01}}{f_{00}} \quad c' = \frac{f_{10}}{f_{00}}$$

therefore the globality condition can be written

$$r > c; \quad r > 1; \quad r > c'$$

and the deceptive condition can be written

$$r < 1 + c - c'$$

It follows that

$$c' < 1; \quad c' < c$$

From these conditions, it is apparent that $c$ may have a range of values which are described as two types of deceptive problems:

$$\text{Type I: } f_{01} > f_{00} \, (c > 1)$$

$$\text{Type II: } f_{00} \geq f_{01} \, (c \leq 1)$$

In developing his model of the two-bit problem, Goldberg made some initial observations. When two-bit strings mate and cross, the offspring are always copies of the parents if the parents are noncomplementary and different from the parent if the parents are complementary. That is, noncomplementary parents 00 and 01 produce offspring 00 and 01. Complementary parents 00 and 11 produce offspring 01 and 10. Using these observations, Goldberg modeled the expected proportions for strings 11 and 01 in the next generation by:

$$p_{11}^{t+1} = p_{11}^{t} \frac{f_{11}}{\overline{f}} \left[1 - \chi \frac{f_{00}}{\overline{f}} p_{00}^{t}\right] + \chi \frac{f_{01} f_{10}}{\overline{f}^2} p_{01}^{t} p_{10}^{t} \tag{5.3}$$

$$p_{01}^{t+1} = p_{01}^{t} \frac{f_{01}}{\overline{f}} \left[1 - \chi \frac{f_{10}}{\overline{f}} p_{10}^{t}\right] + \chi \frac{f_{00} f_{11}}{\overline{f}^2} p_{00}^{t} p_{11}^{t} \tag{5.4}$$

where $\overline{f}$ is the average fitness of the population, $f_{11}$ is the fitness of string 11, $p_{11}^{t}$ is the expected proportion of string 11 at generation $t$, and $\chi$ is the crossover rate.

The proportions of the remaining strings in the next generation can be modeled by exchanging all strings in (5.3) and (5.4) with their complements.

Goldberg found [2] that as long as there is some initial representation of the string 11 in the population, his model predicted a GA would find the global optimum for Type 1 problems. He also found that when solving the Type 2 problem, his model would converge to the suboptimal if the string 00 was too great a proportion of the initial population. He concluded that the Type 2 problem will find the suboptimal string, (GA-HARD), and that the Type 1 will find the optimal string, (GA-EASY).

Goldberg did not consider the effects of mutation on the expected proportions of strings in the next generation when developing his model. Therefore, we use operator $\mathcal{G}$ because it does not require a zero mutation rate and is therefore more characteristic of a true GA. If we let the mutation rate $\mu = 0$, string length $\ell = 2$, and crossover $\chi = 1$, the operator $\mathcal{G}$ can be shown equivalent to Goldberg's model when the string length is two.

# Chapter 6

# RESULTS OF SIMULATIONS AND CONCLUSIONS

In this chapter, we define the Type 1 and Type 2 objective functions used for running simulations. We discuss the methods used for comparison. We discuss the results of simulations of the Markov model, operator $\mathcal{G}$, and GAs. Finally, we make conclusions based on the results.

Simulations of the Markov model, operator $\mathcal{G}$, and GAs were run using Type 1 and Type 2 objective functions with crossover and mutation rates of .8 and .01 respectively. The Type 1 and Type 2 fitness functions used were

| Type 1 | | Type 2 | |
|---|---|---|---|
| string | fitness | string | fitness |
| 00 | 4.00 | 00 | 4.00 |
| 01 | 4.10 | 01 | 3.00 |
| 10 | 0.10 | 10 | 1.00 |
| 11 | 4.11 | 11 | 5.00 |

These functions qualify since $f_{11}$ is the global optimum, $f_{0*} > f_{1*}$, and for Type 1 $c = \frac{4.10}{4.0} > 1$ and for Type 2 $c = \frac{3.0}{4.0} \leq 1$. Type 2 problems were broken into Type 2A and Type 2B. Type 2A has an initial population that is heavily biased towards string 00 and Type 2B has approximately equal initial proportions of strings. For our investigation, we ran simulations for operator $\mathcal{G}$, GAs of population size 10, 100, 1000, and 10000, and both GA and the Markov Model for population sizes 2, 6, 10, 14, 18, and 22. Large memory requirements for the Markov model prevent population sizes larger than 22 from being run since each simulation required two $N \times N$ matrices and six $N \times 1$ matrices of double precision numbers ($N$ is 2300 for population size 22).

Since operator $\mathcal{G}$ models an infinite population, we wanted to see if as the population size of the GA increased, its behavior converged to that of operator $\mathcal{G}$. We use the summary vector $s(k)$ for the GA and the fixed point(s) of $\mathcal{G}$ for making comparisons. Since convergence normally occurs rapidly at first and slows as the number of generations increases, all vectors are recorded at generations which are powers of two to give more detail to lower generations. Graphs representing the results of our simulations are on a logarithmic scale.

To compare the Markov model to a GA we calculate the expected population, $s$, with respect to the steady state distribution and define the function

$$b(k) = \max\{\mid s_0 - s(k)_0 \mid, \mid s_1 - s(k)_1 \mid, \mid s_2 - s(k)_2 \mid, \mid s_3 - s(k)_3 \mid\}$$

to measure how far apart the average populations are. We also calculate the covariance matrices for the Markov model and GA respectively as:

$$D_{ij} = \sum_{m=0}^{N-1} \pi_m \left(\frac{1}{n}(\phi_m)_i - s_i\right)\left(\frac{1}{n}(\phi_m)_j - s_j\right)$$

$$D(k)_{ij} = \sum_{m=0}^{N-1} \frac{c(k)_m}{k} \left(\frac{1}{n}(\phi_m)_i - s(k)_i\right)\left(\frac{1}{n}(\phi_m)_j - s(k)_j\right)$$

To compare these matrices we define the function

$$h(k) = \max_{ij} \mid D_{ij} - D(k)_{ij} \mid$$

Functions $b(k)$ and $h(k)$ are used to determine if convergence is occurring, the rate at which it occurs, and the effect of population size.

In addition to graphing these convergence functions, we also rotate $\Lambda$, which is embedded in four dimensions, into a three dimensional space so that we can visualize it. The result is a solid tetrahedron. Probability distributions over $\Lambda$ are visualized by selecting a population coordinate in the tetrahedron and summing the probabilities of all populations within radius $r$ of the coordinate. The sums are used to scale color intensity of the coordinate where darker color represents larger sums. Therefore, dark areas of the tetrahedron represent populations that occur with high probability. For example, fig 51 shows that for population size 22, Type 2, populations with the largest proportions of string 11 have a higher probability of occurring.

## 6.1 Type I Results

The Markov Model/GA, Type 1 simulations, figs 1-6 in the appendix, show that functions $b$ and $h$ approach zero, therefore string averages of the GA and the model are getting closer as the number of generations increases. They also show that as population size increases there is less variation in the functions, therefore string averages for large populations are closer than for small populations. Convergence seems to occur by 100,000 generations for all population sizes. The tetrahedrons for Markov Model, Type 1 simulations, figs 39-44, show that as the population size increases, populations are more probable near the fixed point of $\mathcal{G}$, ¡0.044, 0.764, 0.008, 0.182¿ (fig 19). The fixed point is found by using the components of the vector as coordinates in the tetrahedron. However, they do not conclusively show that as population size increases positive probability is given only to the fixed point of $\mathcal{G}$. We cannot run simulations of the Markov model large enough to show this result, but since as the number of generations become large the results of GAs seem to match those of the Markov model, we used a GA to simulate the Markov chain. The tetrahedron in figure 45 clearly shows that the populations encountered are grouped near the fixed point of $\mathcal{G}$ as given in fig 19.

If we compare results of simulations of a GA, figs 20-23, and operator $\mathcal{G}$, fig 19, we see that as population size increases, convergence to the expected proportions becomes more rapid. Note that these graphs represent averages of each string. This is not surprising since operator $\mathcal{G}$ models an infinite population and shows fast convergence. All graphs show that the suboptimal string 01 is the clear winner. Hence the Type 1 problem is GA-HARD when a mutation rate of .01 is used. This was unexpected since Goldberg [2] had shown the Type 1 problem without mutation to be easy. We verified Goldbergs result by using operator $\mathcal{G}$ with mutation off, fig 38, and found the optimal string as Goldberg predicted.

To confirm that for operator $\mathcal{G}$ the Type 1 problem with mutation is difficult for every initial population, we ran simulations using a lattice of coordinates in the tetrahedron as initial populations. Each coordinate was colored according to its corresponding fixed point. The results, fig 52, show only one pixel intensity, therefore only one fixed point was found. A more extensive simulation used four million initial random populations, and again all converged under operator $\mathcal{G}$ to the same fixed point. Random populations correspond to random points in $\Lambda$ and were chosen uniformly in $\Lambda$ according to

$$< u_1^{\frac{1}{3}}\, u_2^{\frac{1}{2}}\, u_3,\ u_1^{\frac{1}{3}}\,(1 - u_2^{\frac{1}{2}}),\ u_1^{\frac{1}{3}}\, u_2^{\frac{1}{2}}\,(1 - u_3),\ 1 - u_1^{\frac{1}{3}} > \in \Lambda$$

where $u_1$, $u_2$, and $u_3$ are independent uniformly distributed random variables in [0,1]. We conclude that mutation makes the Type 1 problem GA-HARD according to Goldbergs criteria (remember that on two bit strings, operator $\mathcal{G}$ is equivalent to extending Goldbergs model to accommodate mutation).

Since we were obtaining unexpected results, we wondered how well Goldberg's results would model actual GAs. After all, his model assumes an infinite population in addition to no mutation. We ran simulations for population sizes 10000 and 100000, fig 24-

25. Population size 100000 finds the optimal string 11 as predicted, but population size 10000 finds the suboptimal string 01 at 10,000 generations. Evidently 10000 is not a large enough population size for an infinite population model of the Type 1 problem without mutation to be correct. This should caution the GA community against applying these models to predict GA behavior (as they currently do) since real GAs are run with small populations.

## 6.2 Type II Results

The objective function used in the Type 2 problem has two fixed points, figs 26 (Type 2A), 33 (Type 2B). To determine the proportion of populations that converged to each fixed point, we ran simulations as before using random initial populations. The results show 75% of the populations converge to the Type 2B fixed point. A simulation using a lattice of coordinates in the tetrahedron as initial populations, fig 53, shows the basins of attraction for the two fixed points of $\mathcal{G}$.

Since the Markov model is independent of initial population, we first look at the tetrahedrons for the Type 2 problem, figs 46-51. We can see that as the population size gets larger, populations are more probable near the Type 2B fixed point. It is apparent that for the Type 2 objective function, this fixed point is a stronger attractor than the other.

If we look at the Markov Model/GA, Type 2A simulations, figs 7-12, and the Type 2B simulations, figs 13-18, we see that the 2A and 2B simulations converge although at different rates. The graphs show that the 2A function $b$ values are initially larger. This is to be expected since a string other than the one found by the Markov model initially dominates. However, since convergence does occur, the GA must eventually agree with Markov model.

The GA, Type 2A simulations, figs 27, 29-31, showed that population sizes 10 and 100 found the optimal string as predicted by the Markov model while the larger populations were dominated by the suboptimal. To investigate this apparent anomaly, we ran GAs for population sizes 10, 20, 30, 40, 50, and 60, fig 32. As population size increased, the time required for the GA to find the optimal string increased. This explains the results for the larger populations, fig 30 and 31; the GA simply was not run large enough. The Markov Model is based on an infinite number of generations and finds the optimal string. Operator $\mathcal{G}$ is based on an infinite population and and finds the suboptimal. As population size increases, we expect a GA to more closely follow operator $\mathcal{G}$ for an increasing number of generations by having populations dominated with the suboptimal. However, this behavior must eventually reverse since the GA converges to the Markov model, fig 32, as the number of generations goes to infinity.

The graphs show that the Type 2 problem with mutation is not GA-HARD. Again, these results were unexpected since Goldberg [2] had shown the Type 2 problem without mutation to be hard.

We conclude that GAs do closely follow the Markov Model and conjecture that the Markov model agrees with the fixed point of $\mathcal{G}$ which has the largest basin of attraction. Convergence of GAs to the Markov model occurs quickly for the Type 1 problem, but we saw in the Type 2 problems that when operator $\mathcal{G}$ has more than one fixed point, convergence may require large numbers of generations. When there is only one fixed point of operator $\mathcal{G}$, then increasing population size speeds convergence of a GA to the Markov model. Conversely, if there is more than one fixed point and the initial population is within the basin of attraction of a suboptimal, then increasing population size slows convergence. Perhaps the most surprising result is that mutation can make GA-HARD problems easy and GA-EASY problems hard.

Since we have shown that GAs using a small population size may not yield the same results as an infinite population model, and that mutation can reverse results, caution should be exercised by the GA community when using these model to predict GA behavior.

# BIBLIOGRAPHY

# BIBLIOGRAPHY

[1] Vose, M.D. and Liepins, G.E., *Punctuated Equilibria in Genetic Search*, Accepted, Complex Systems.

[2] Goldberg, David E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Co., Reading, MA., 1989.

[3] Nix, A.E. and Vose M.D., *Modeling Genetic Algorithms with Markov Chains*, Accepted, Annals of Mathematics and Artificial Intelligence.

[4] Minc, H., *Nonnegative Matrices*, John Wiley and Sons Publishing, New York, 1988.

[5] Rosen, Kenneth, *Discrete Mathematics and Its Applications*, Random House, New York, 1988.

[6] Smith, B.T. and others, *Matrix Eigensystem Routines - EISPACK Guide*, Springer-Verlag, New York, 1976.

[7] Garbow, B.S. and Boyle, J.M. and Dongarra, J.J. and Moler, C.B., *Matrix Eigensystem Routines - EISPACK Guide Extension*, Springer-Verlag, New York, 1977.

[8] T. Davis, *Toward An Extrapolation Of The Simulated Annealing Convergence Theory Onto The Simple Genetic Algorithm*, Dissertation presented to the University of Florada, 1991.

## APPENDIX

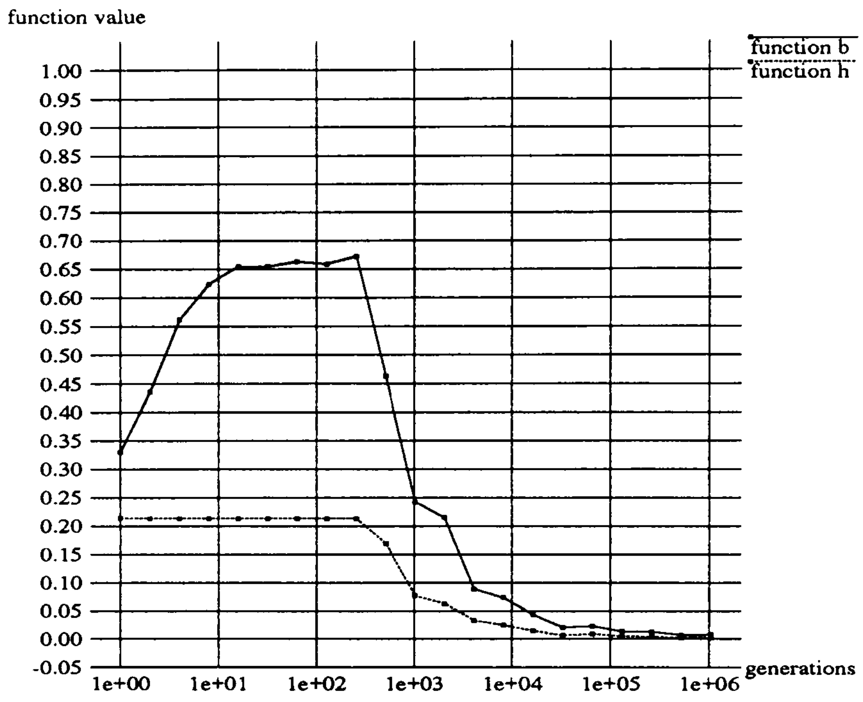


Figure 1. Convergence of a Genetic Algorithm to the Markov Model for Population 2, Type 1

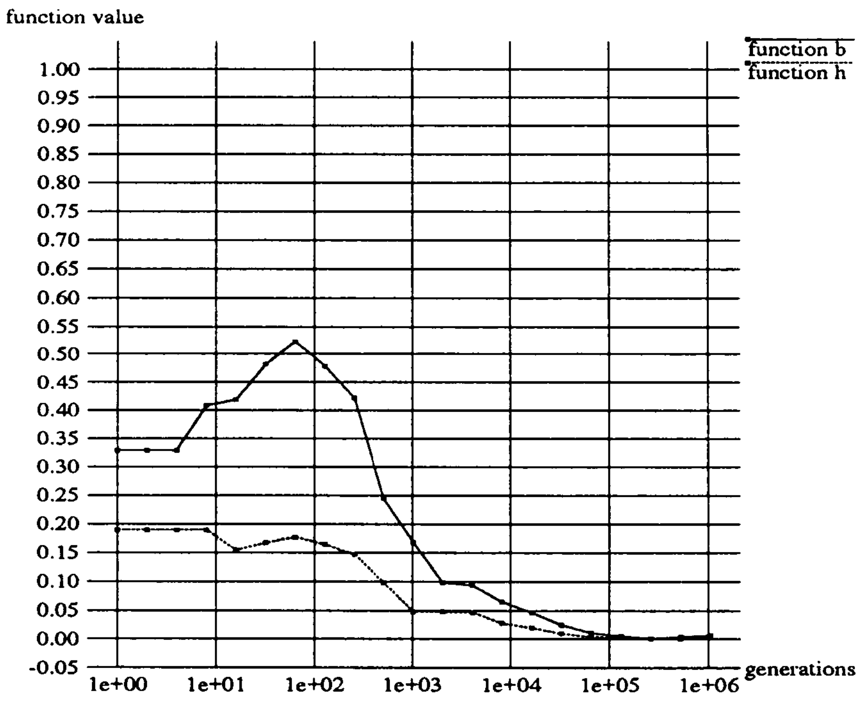


Figure 2. Convergence of a Genetic Algorithm to the Markov Model for Population 6, Type 1

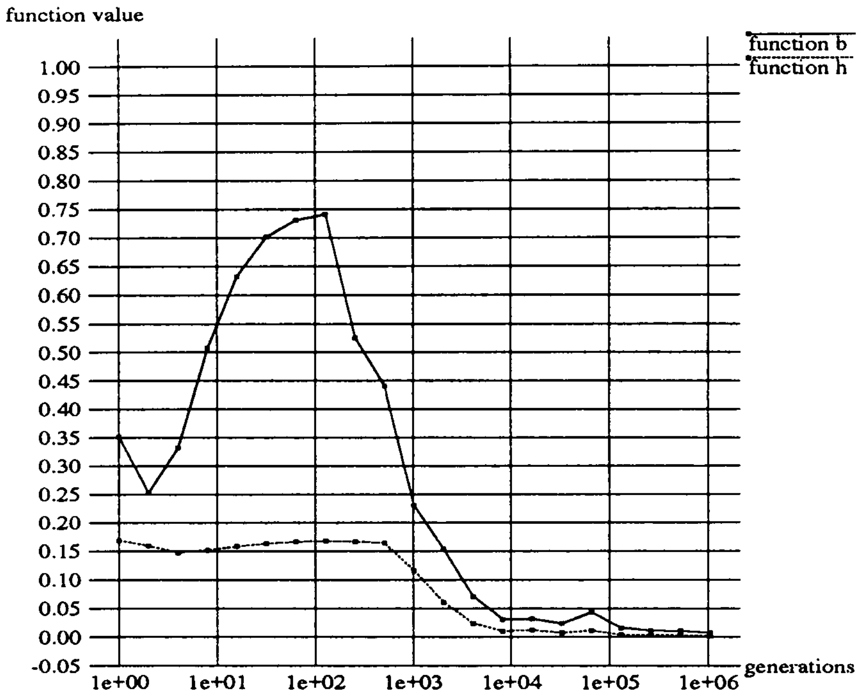


Figure 3. Convergence of a Genetic Algorithm to the Markov Model for Population 10, Type 1

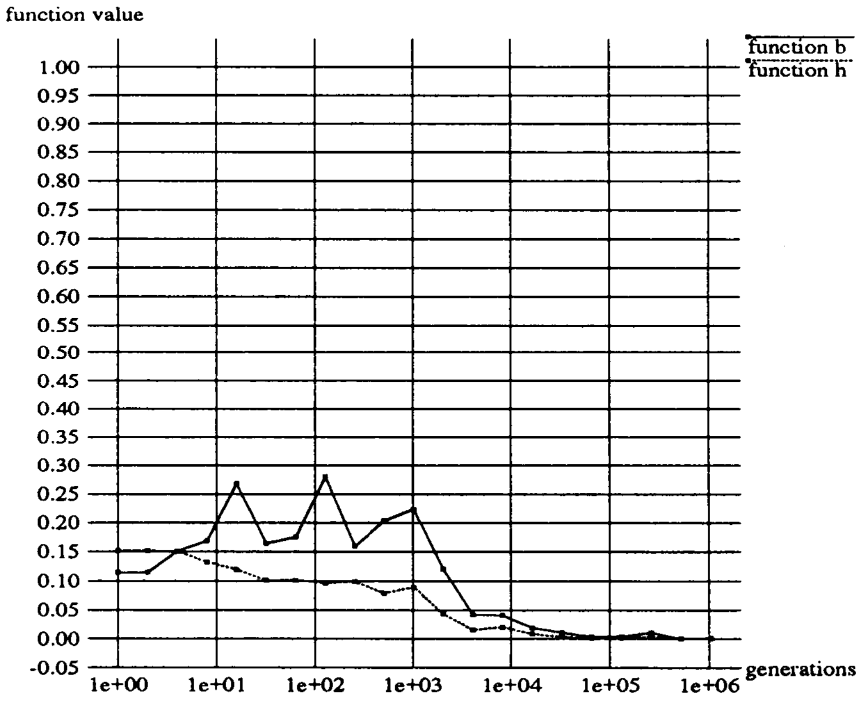


Figure 4. Convergence of a Genetic Algorithm to the Markov Model for Population 14, Type 1

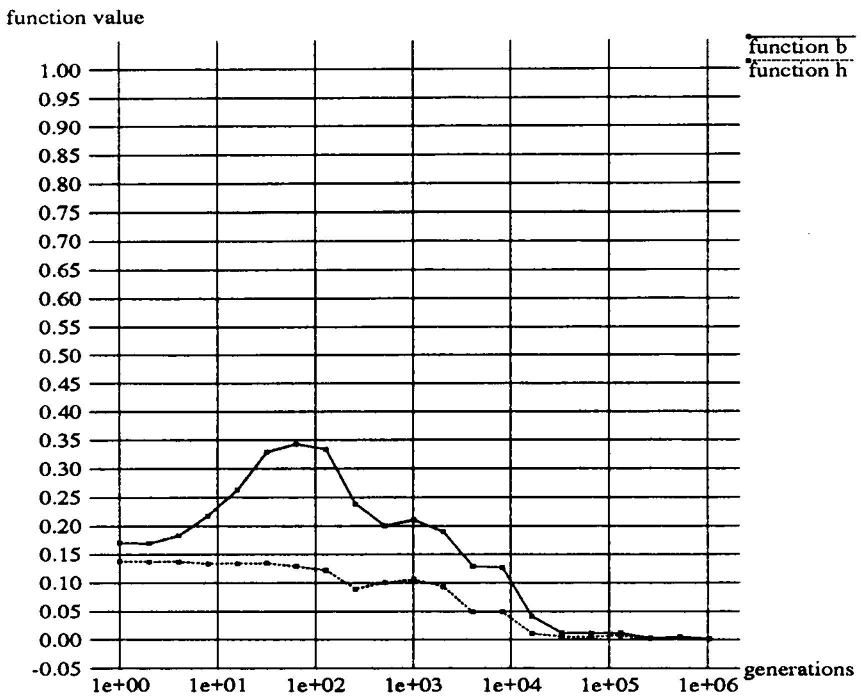


Figure 5. Convergence of a Genetic Algorithm to the Markov Model for Population 18, Type 1

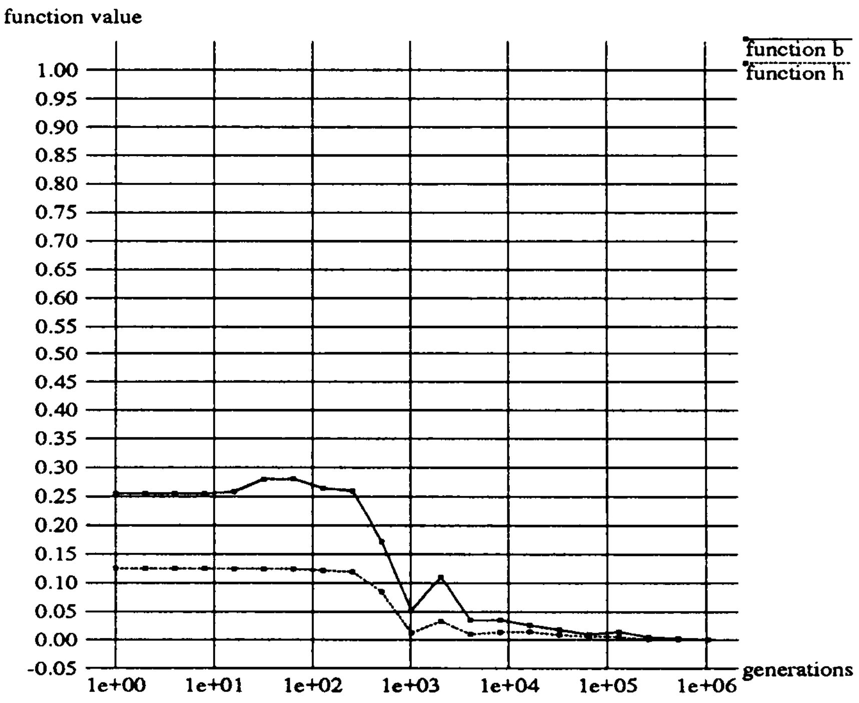


Figure 6. Convergence of a Genetic Algorithm to the Markov Model for Population 22, Type 1

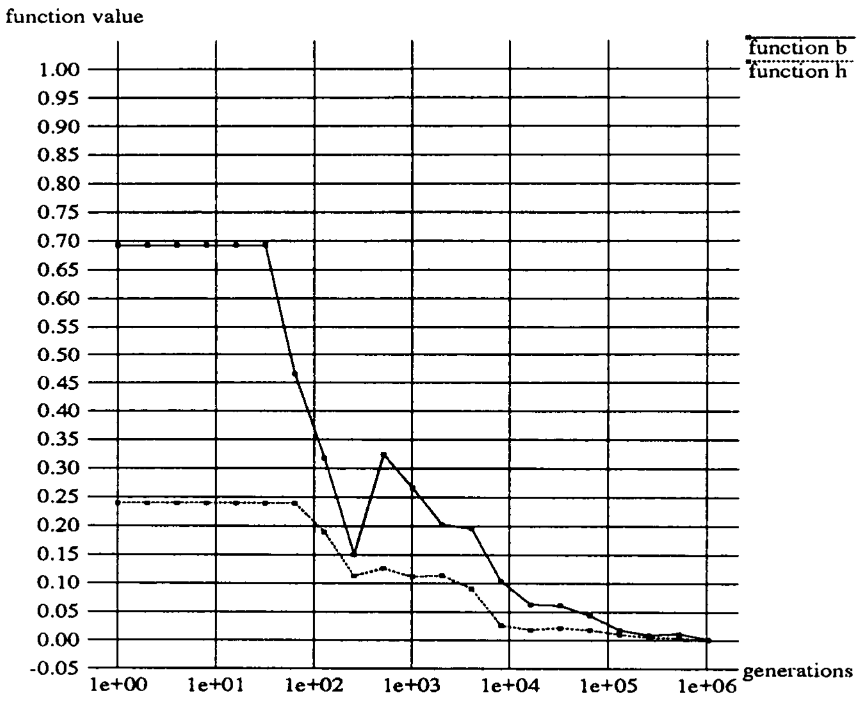


Figure 7. Convergence of a Genetic Algorithm to the Markov Model for Population 2, Type 2A

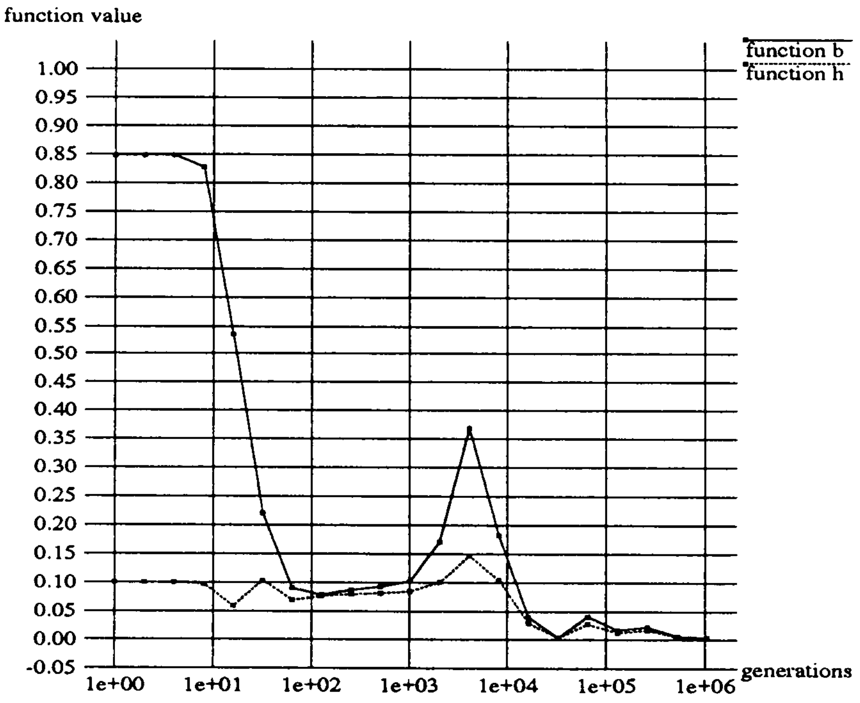


Figure 8. Convergence of a Genetic Algorithm to the Markov Model for Population 6, Type 2A

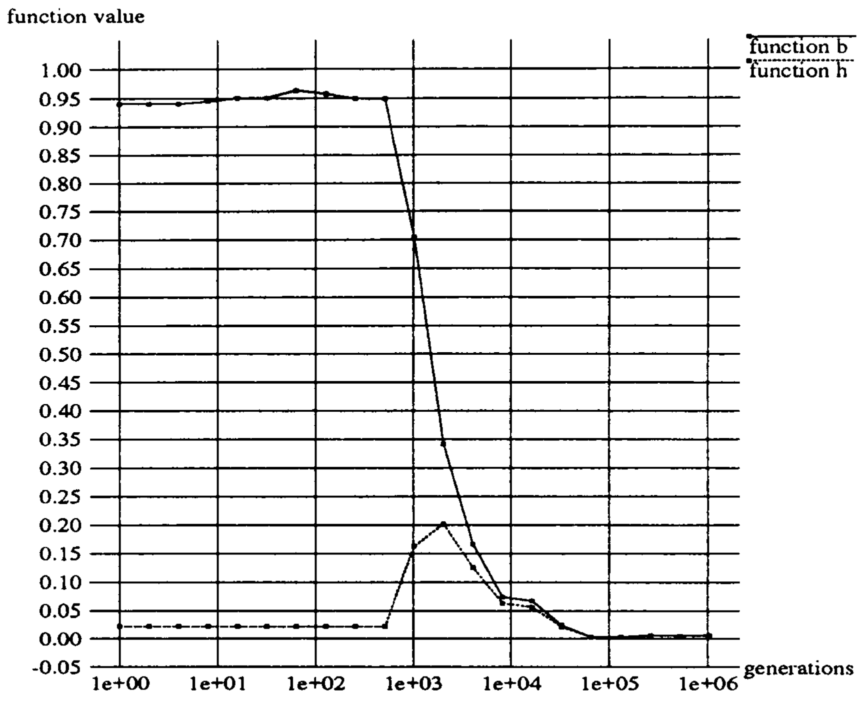


Figure 9. Convergence of a Genetic Algorithm to the Markov Model for Population 10, Type 2A

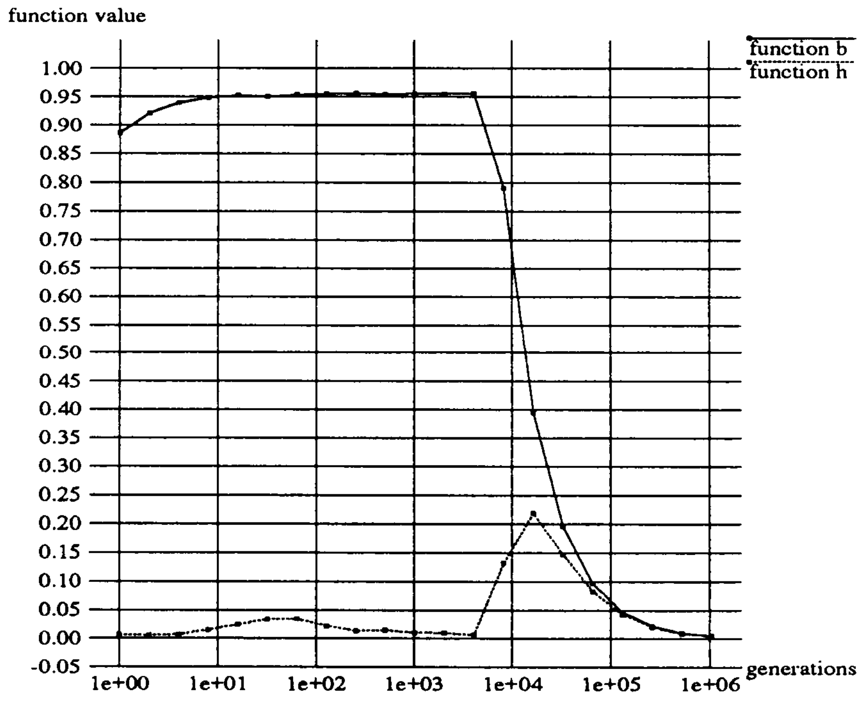


Figure 10. Convergence of a Genetic Algorithm to the Markov Model for Population 14, Type 2A

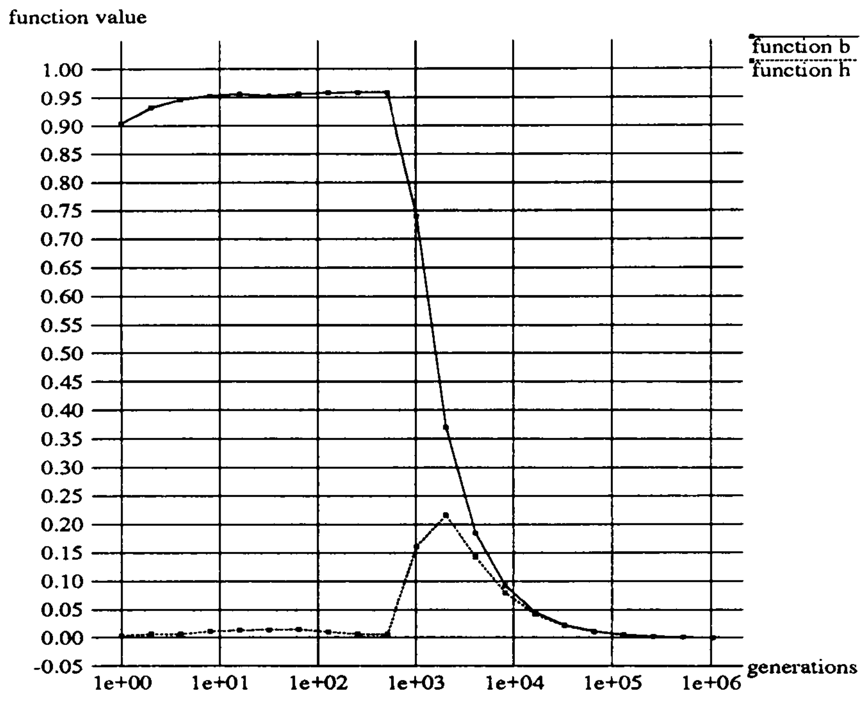


Figure 11. Convergence of a Genetic Algorithm to the Markov Model for Population 18, Type 2A

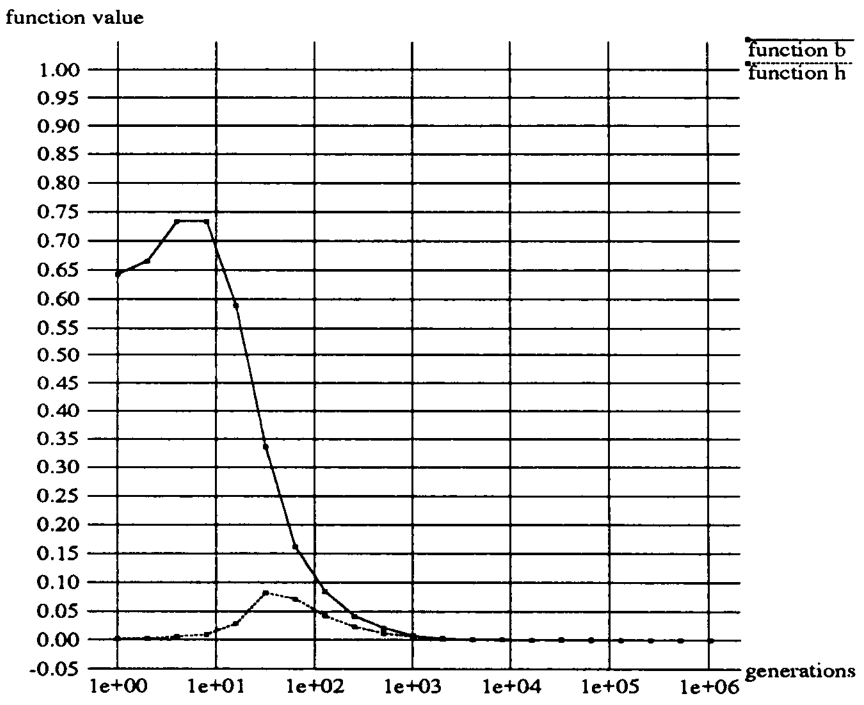


Figure 12. Convergence of a Genetic Algorithm to the Markov Model for Population 22, Type 2A

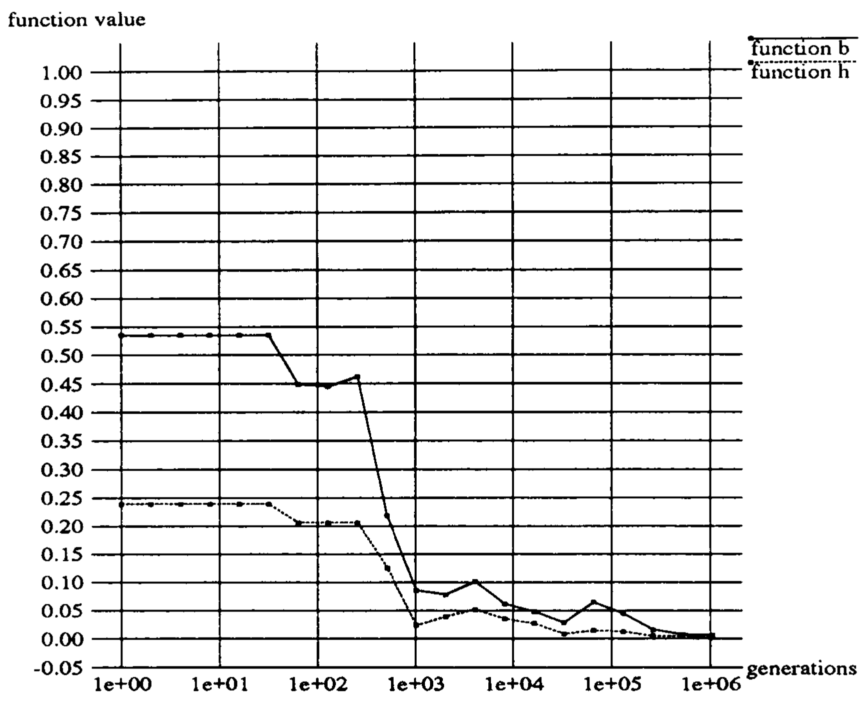


Figure 13. Convergence of a Genetic Algorithm to the Markov Model for Population 2, Type 2B

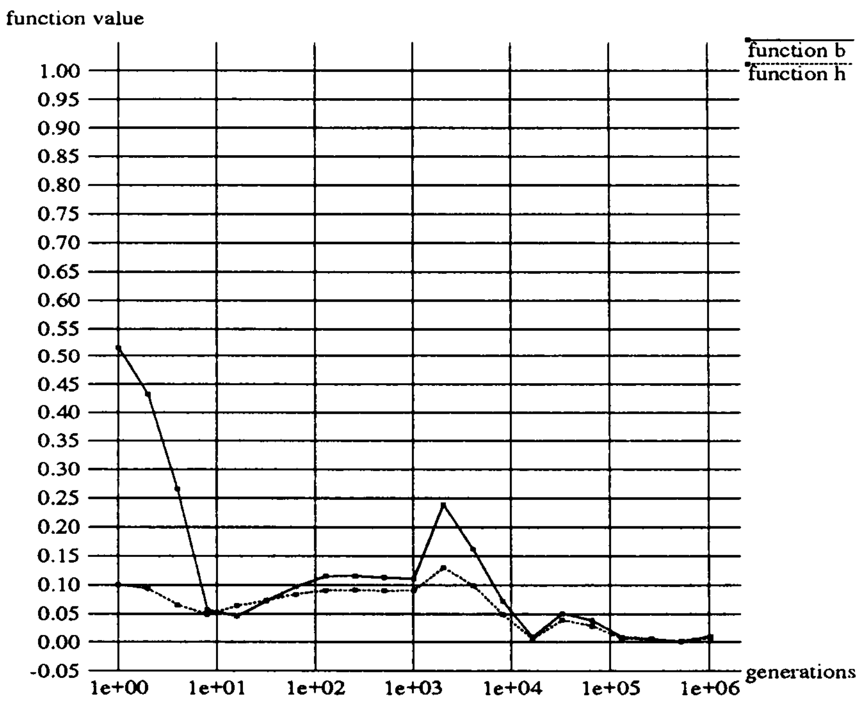


Figure 14. Convergence of a Genetic Algorithm to the Markov Model for Population 6, Type 2B

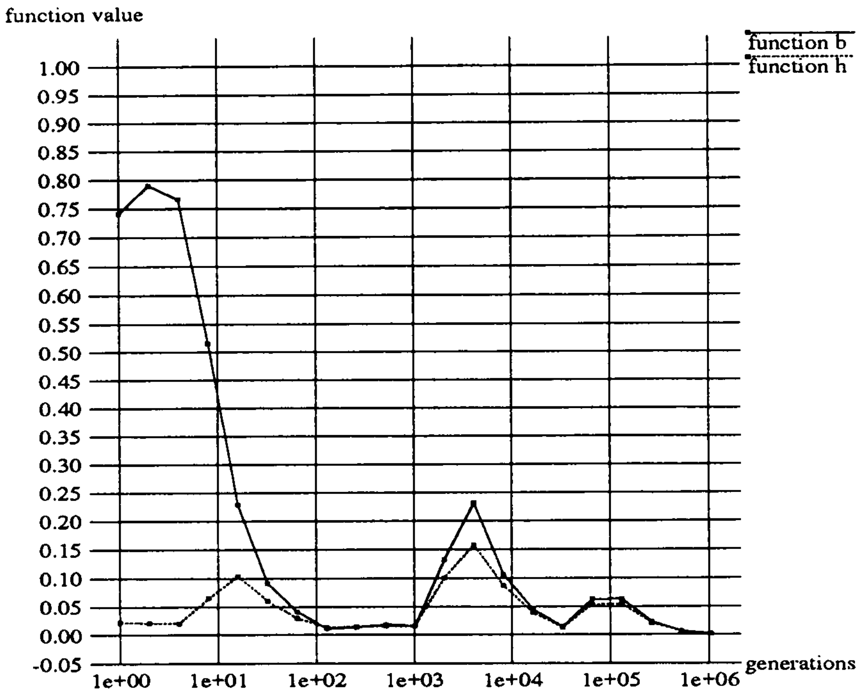


Figure 15. Convergence of a Genetic Algorithm to the Markov Model for Population 10, Type 2B

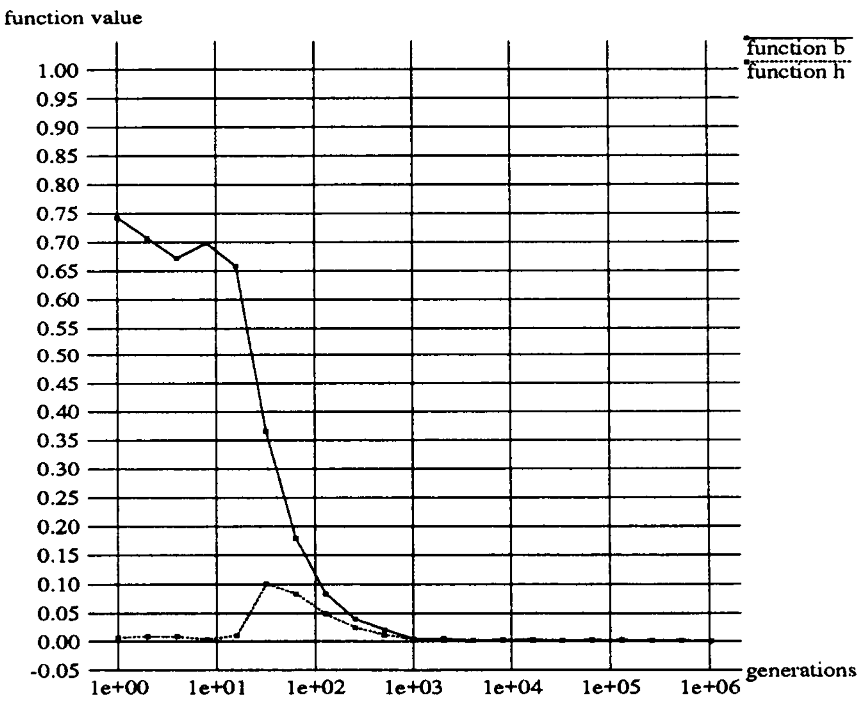


Figure 16. Convergence of a Genetic Algorithm to the Markov Model for Population 14, Type 2B

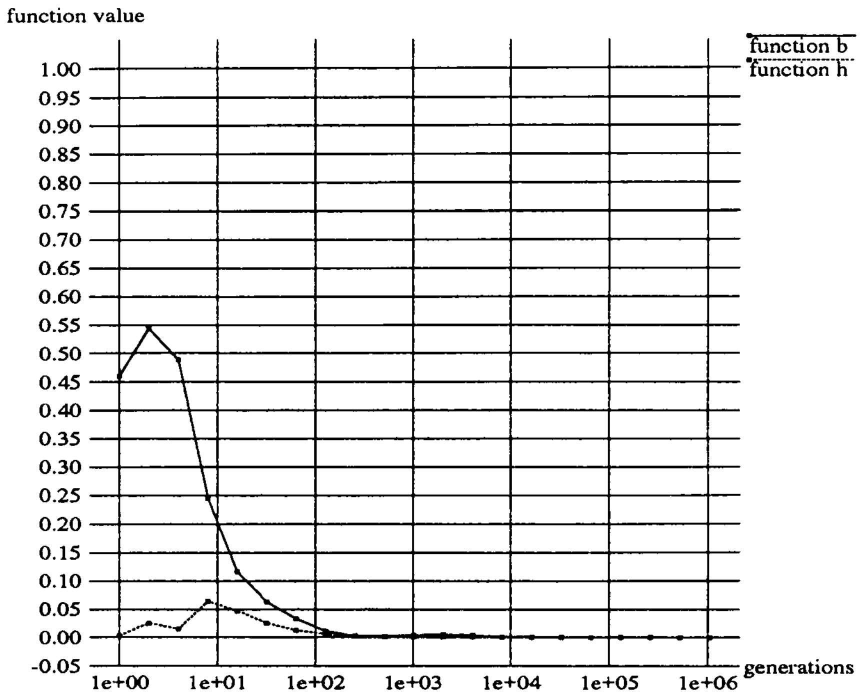


Figure 17. Convergence of a Genetic Algorithm to the Markov Model for Population 18, Type 2B

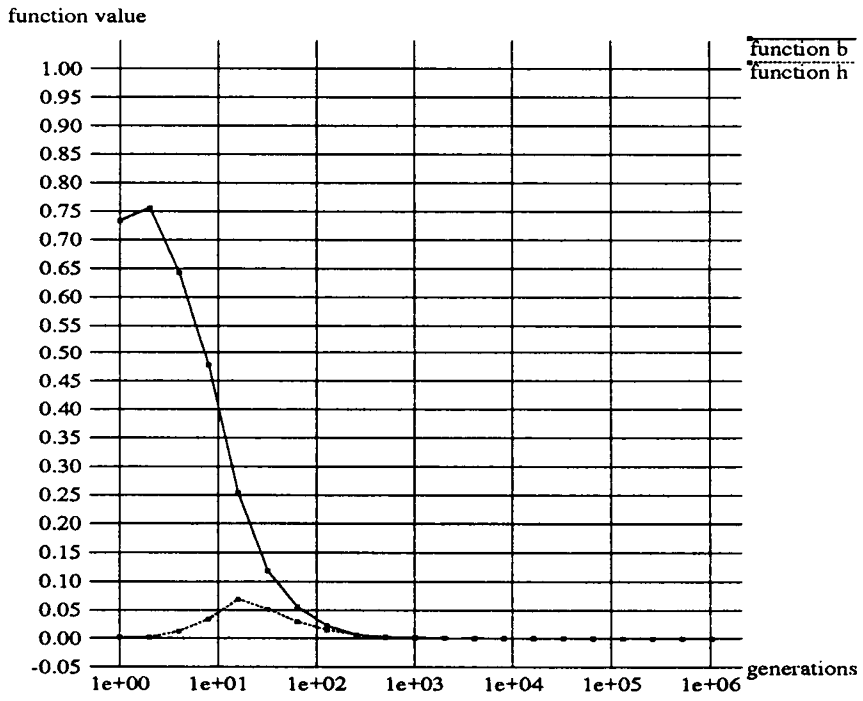


Figure 18. Convergence of a Genetic Algorithm to the Markov Model for Population 22, Type 2B

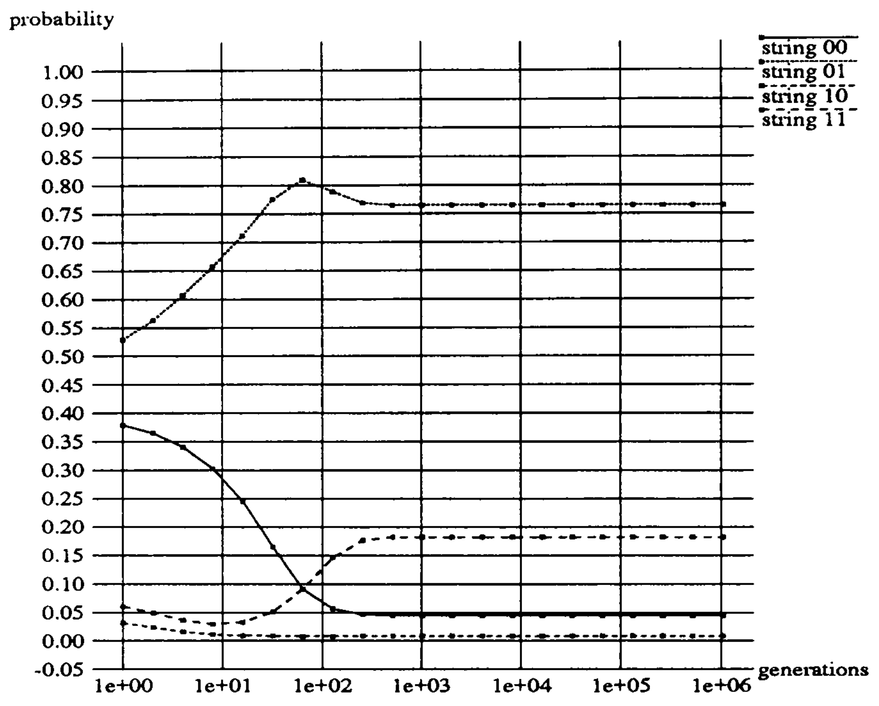


Figure 19. Probability of Encountering String i in k Generations for Infinite Population, Type 1

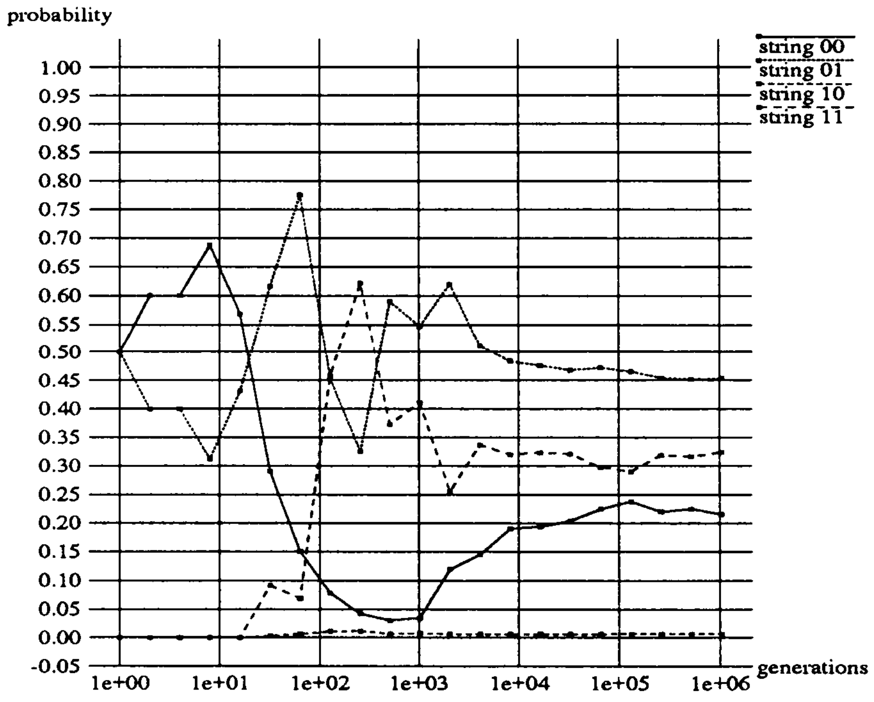


Figure 20. Probability of Encountering String i in k Generations for Population 10, Type 1

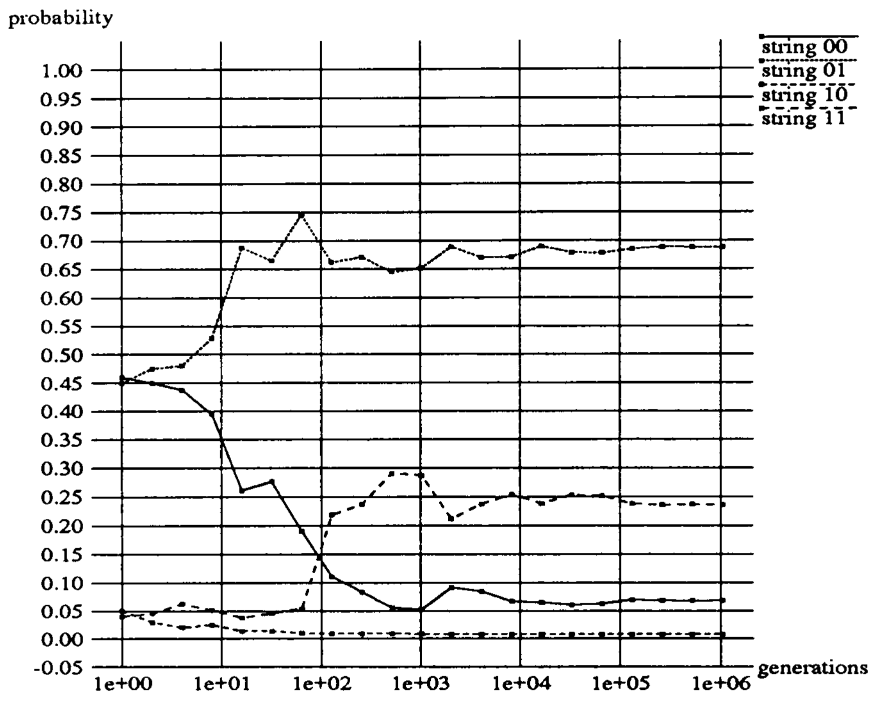


Figure 21. Probability of Encountering String i in k Generations for Population 100, Type 1

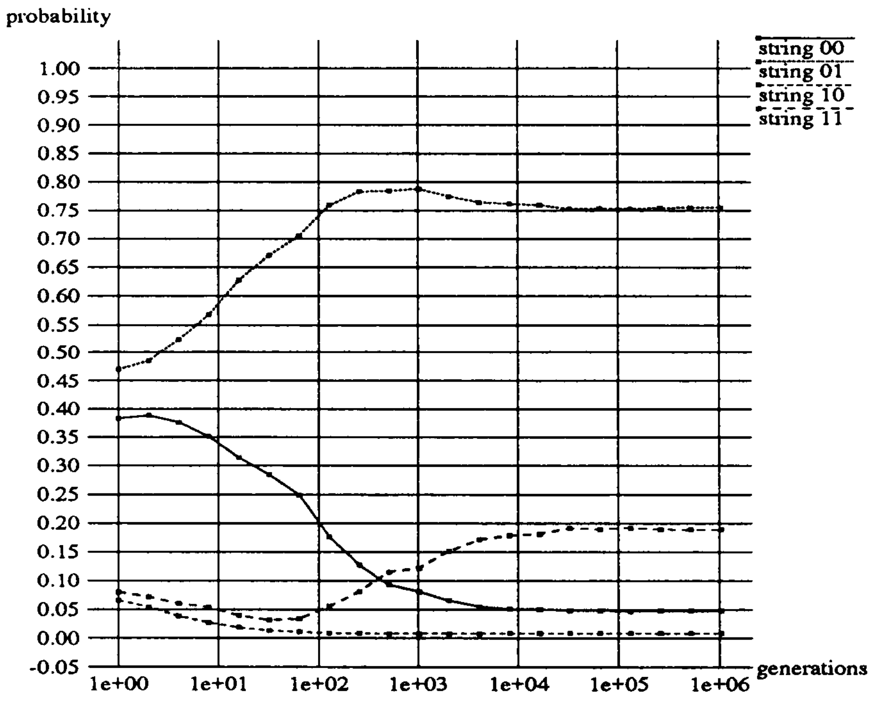


Figure 22. Probability of Encountering String i in k Generations for Population 1000, Type 1

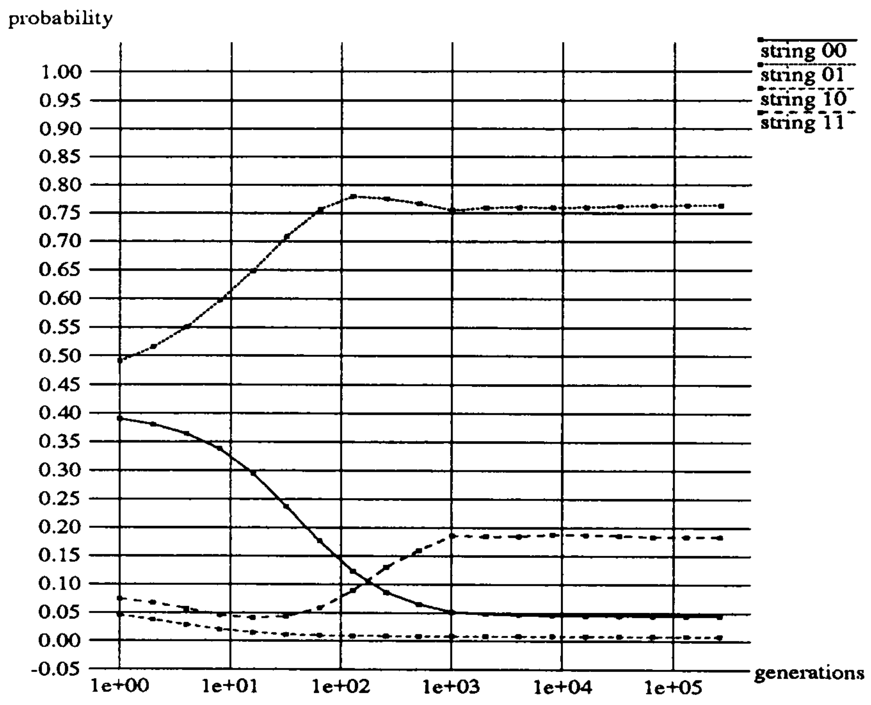


Figure 23. Probability of Encountering String i in k Generations for Population 10000, Type 1

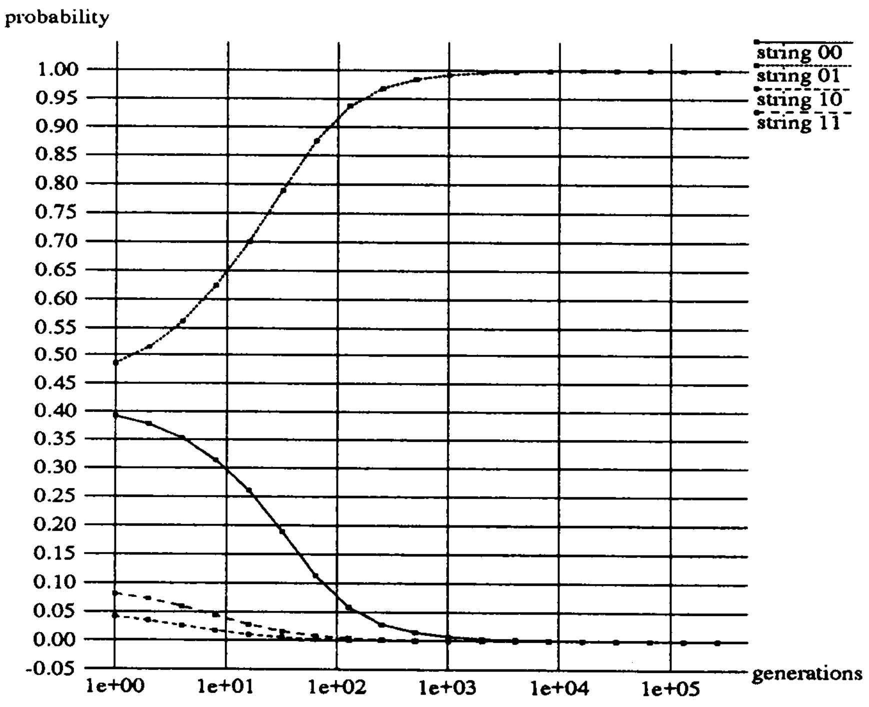


Figure 24. Probability of Encountering String i in k Generations for Population 10000, Type 1, Mutation 0

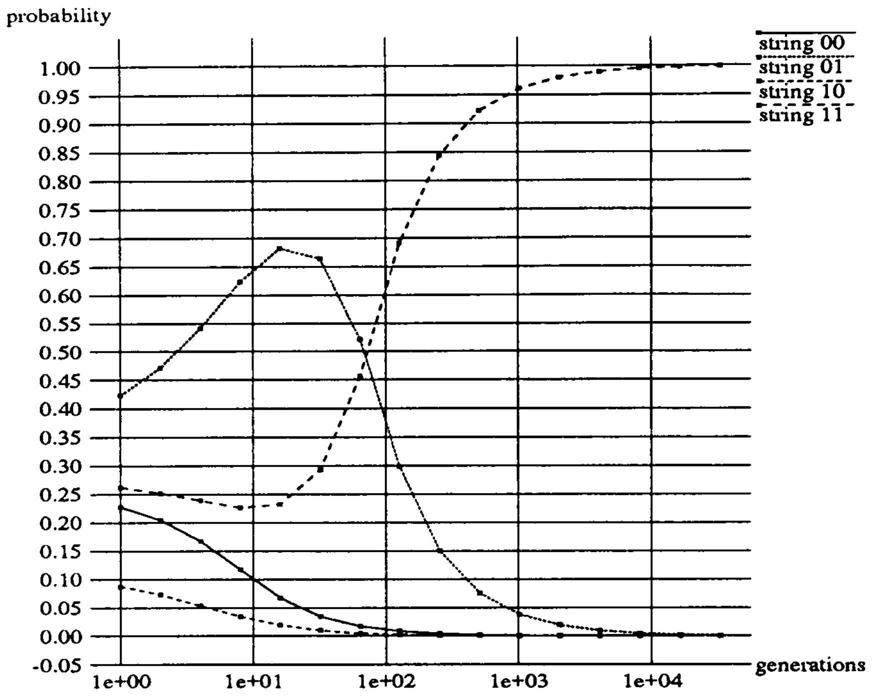


Figure 25. Probability of Encountering String i in k Generations for Population 100000, Type 1, Mutation 0

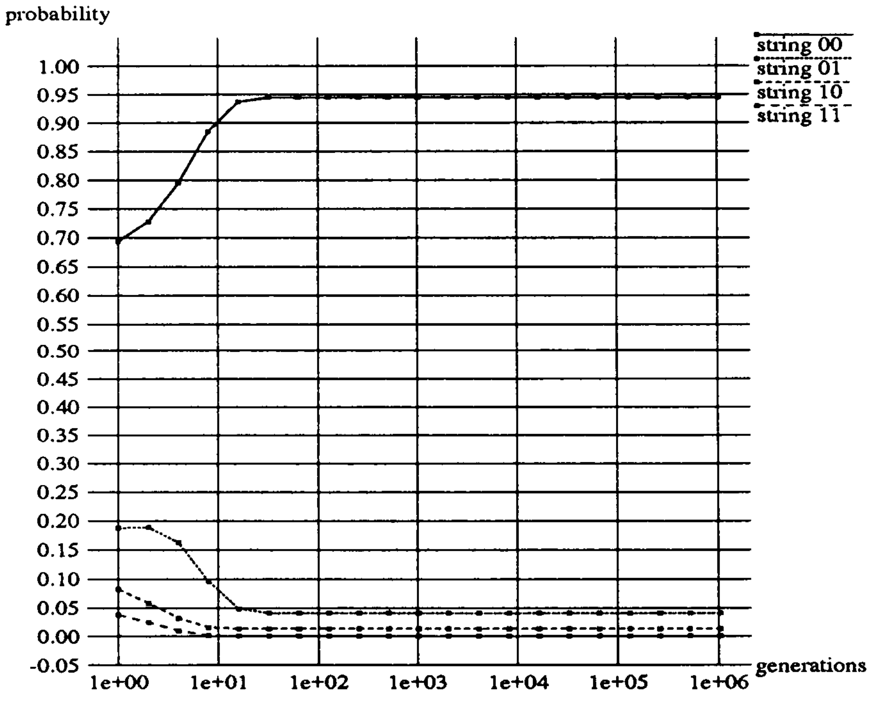


Figure 26. Probability of Encountering String i in k Generations for Infinite Population, Type 2A

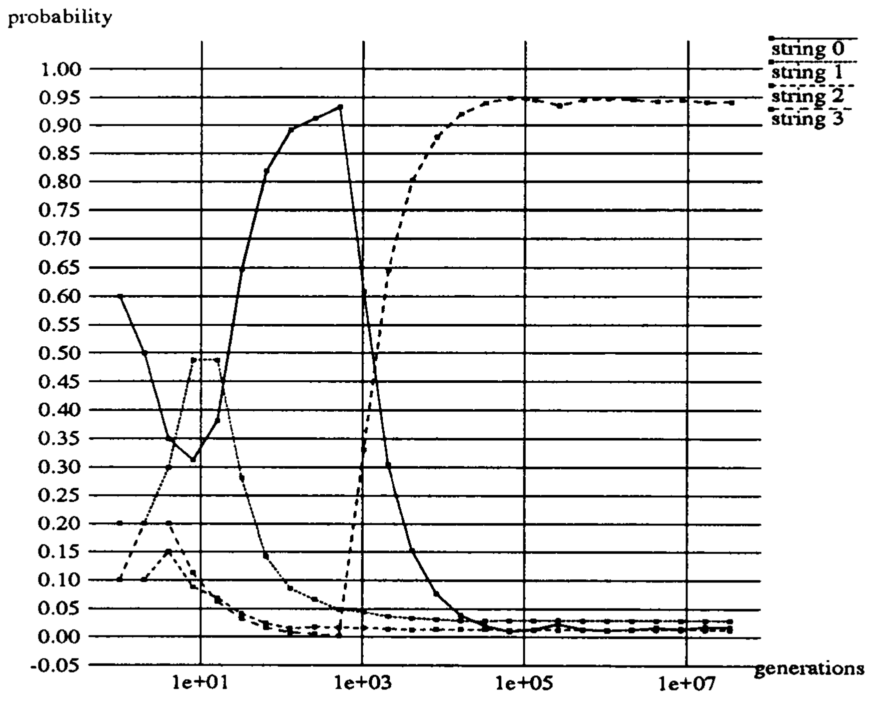


Figure 27. Probability of Encountering String i in k Generations for Population 10, Type 2A

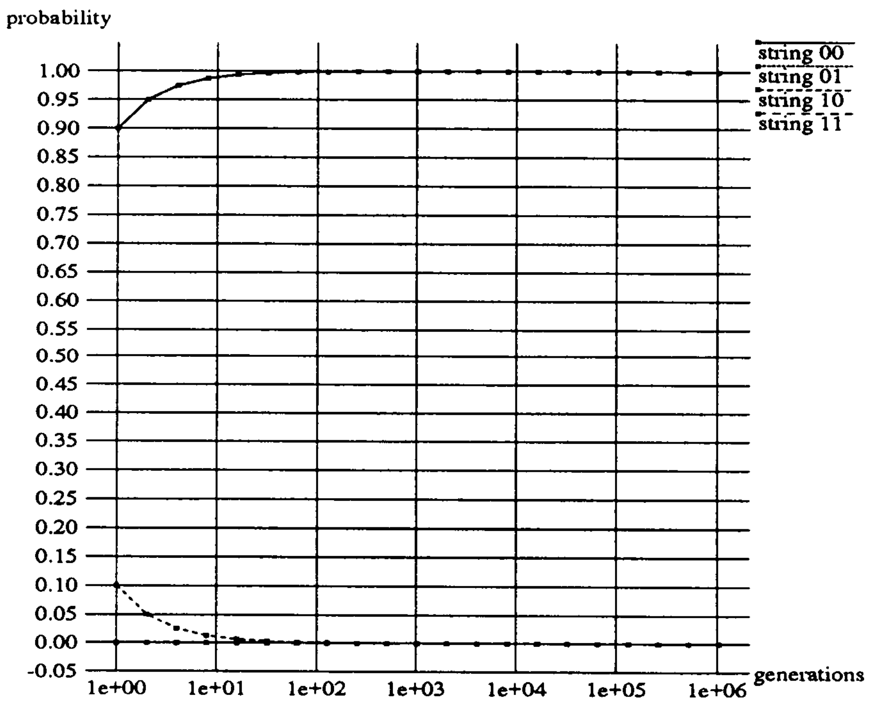


Figure 28. Probability of Encountering String i in k Generations for Population 10, Type 2A, Mutation 0

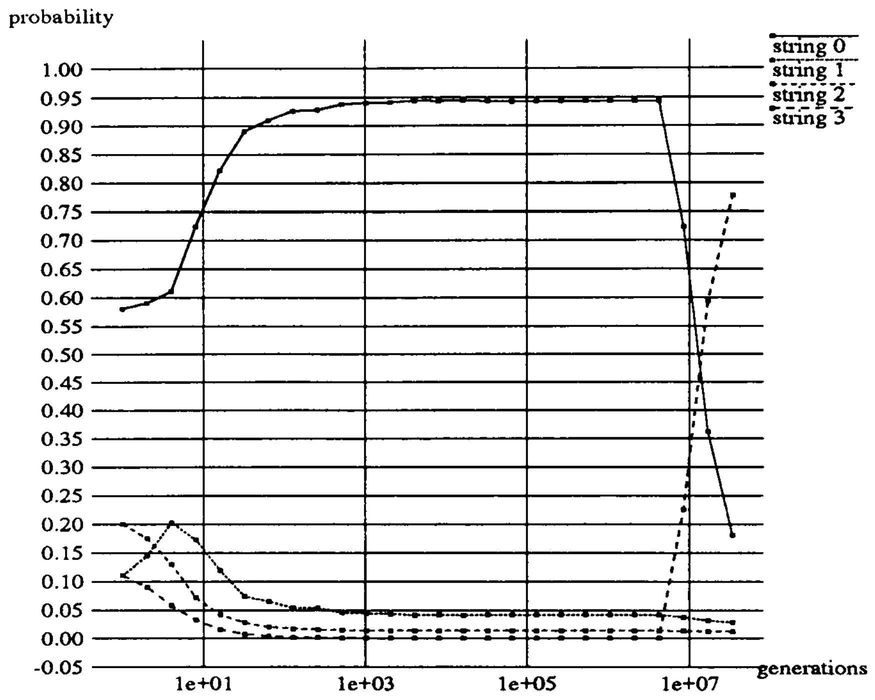


Figure 29. Probability of Encountering String i in k Generations for Population 100, Type 2A

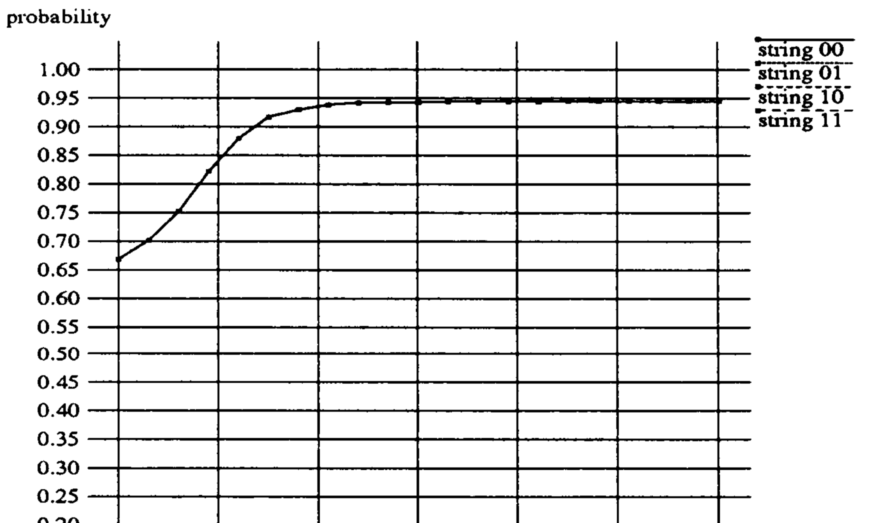


Figure 30. Probability of Encountering String i in k Generations for Population 1000, Type 2A

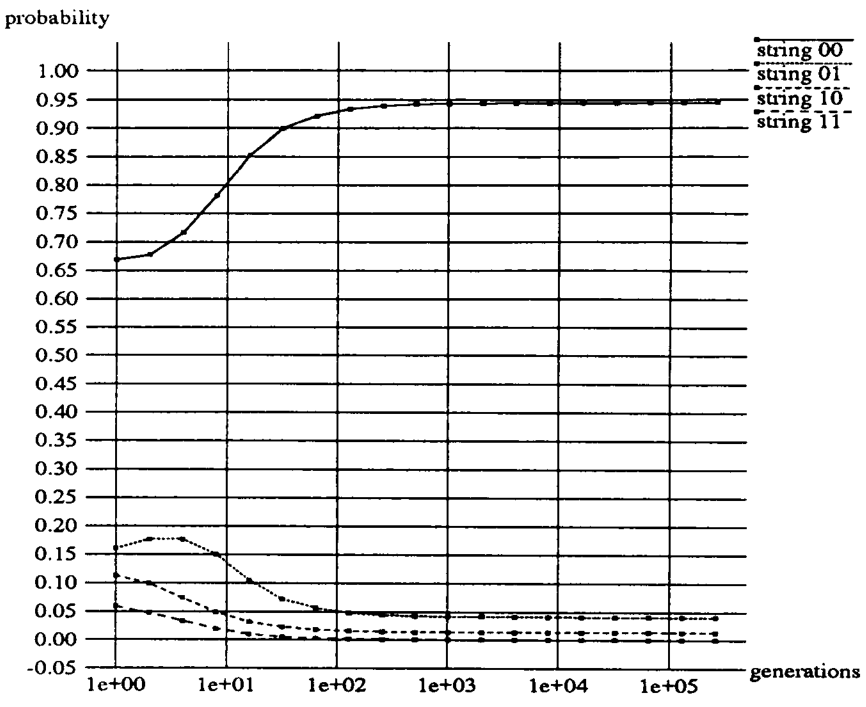


Figure 31. Probability of Encountering String i in k Generations for Population 10000, Type 2A

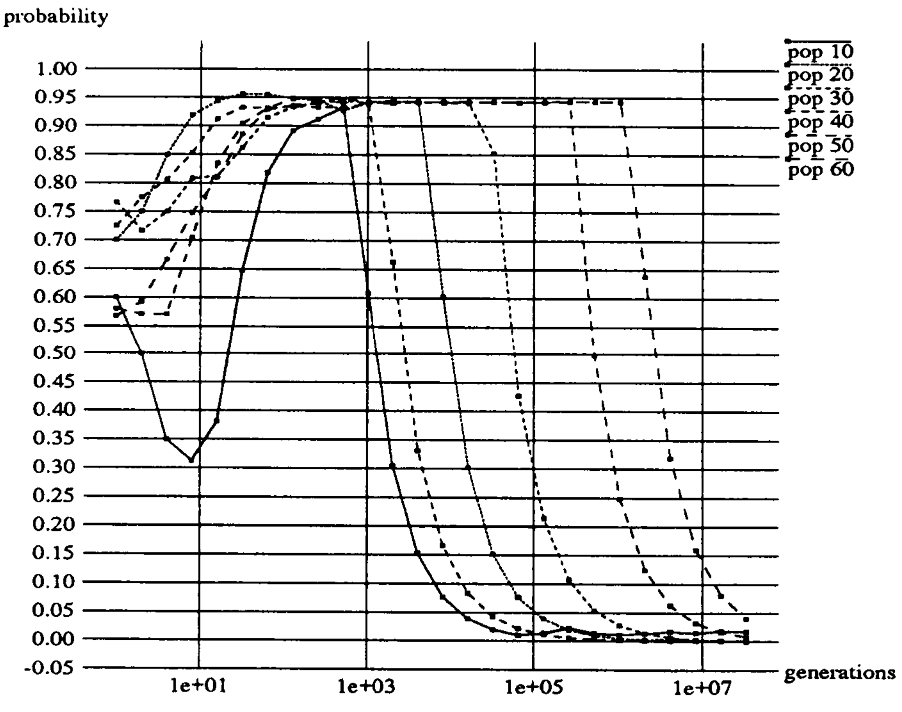


Figure 32. Probability of Encountering String 0 in k Generations for Populations 10, 20, 30, 40, 50, 60 Type 2A

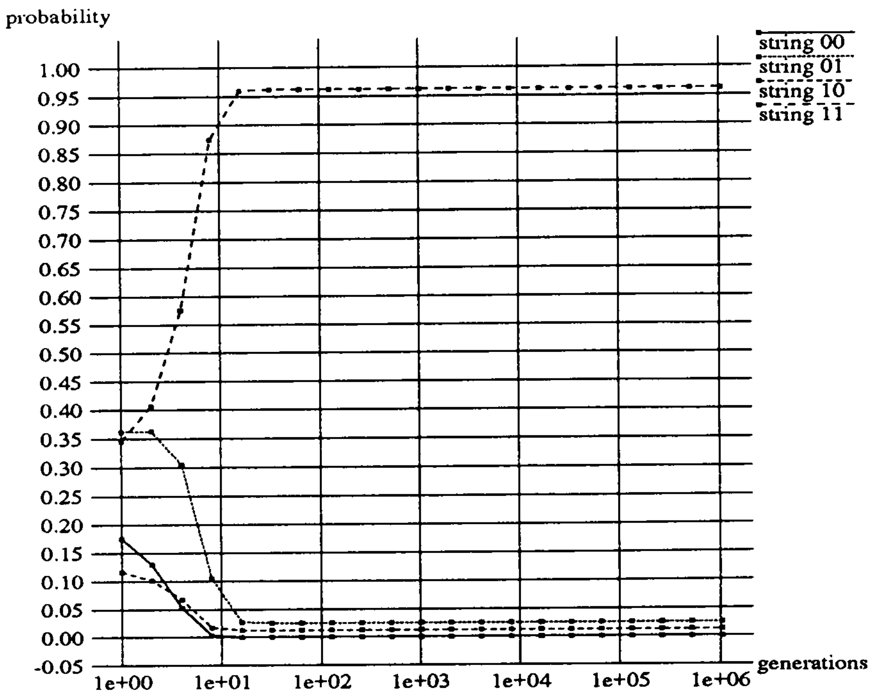


Figure 33. Probability of Encountering String i in k Generations for Infinite Population, Type 2B

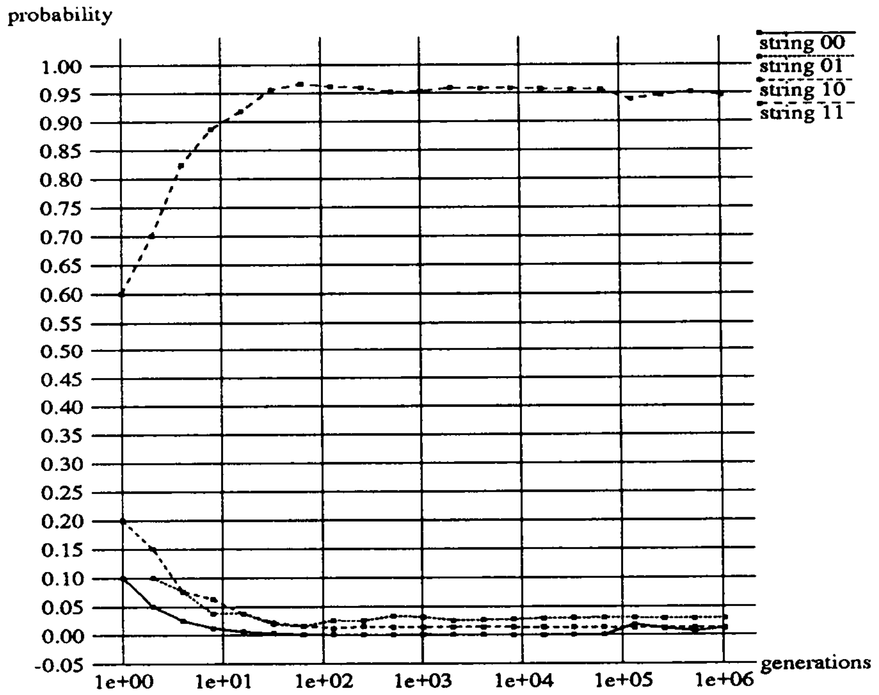


Figure 34. Probability of Encountering String i in k Generations for Population 10, Type 2B

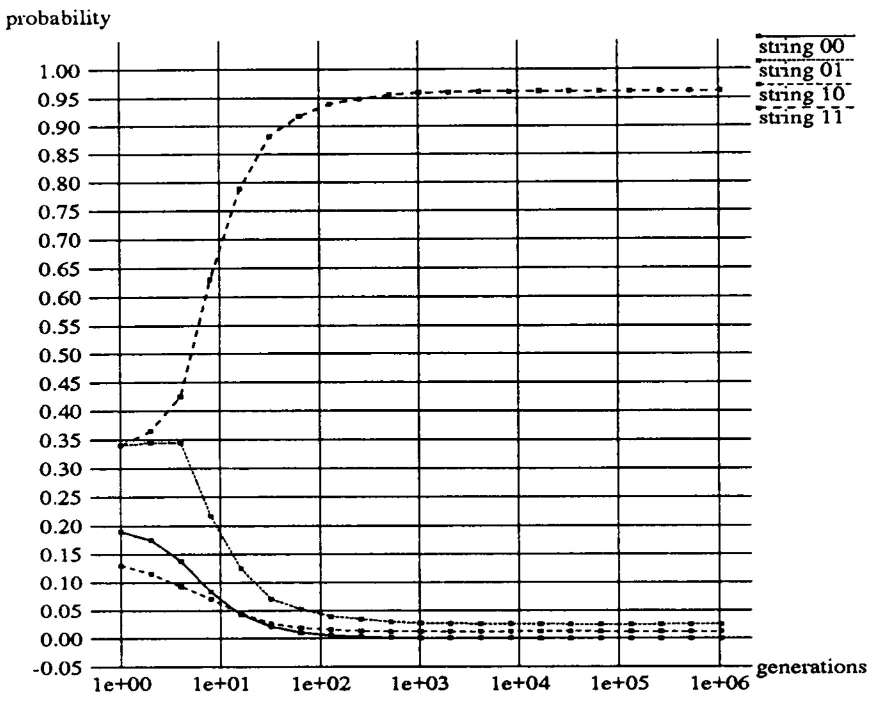


Figure 35. Probability of Encountering String i in k Generations for Population 100, Type 2B

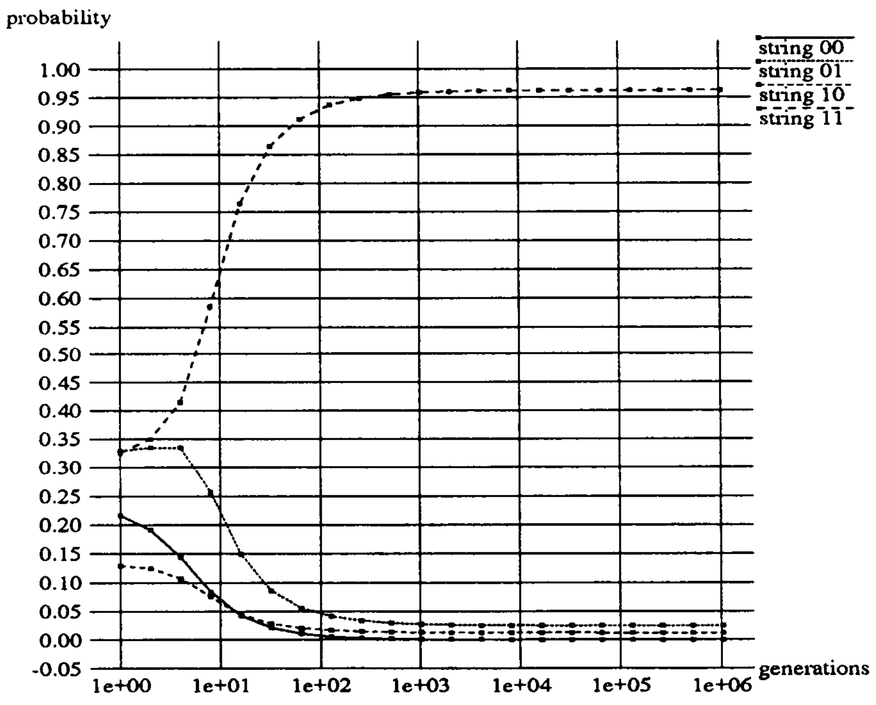


Figure 36. Probability of Encountering String i in k Generations for Population 1000, Type 2B

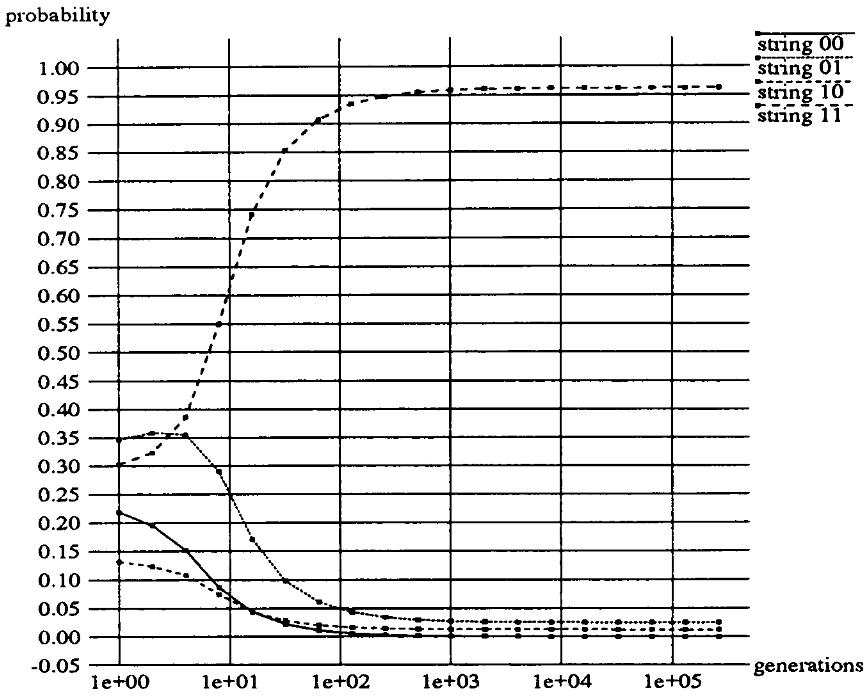


Figure 37. Probability of Encountering String i in k
Generations for Population 10000, Type 2B

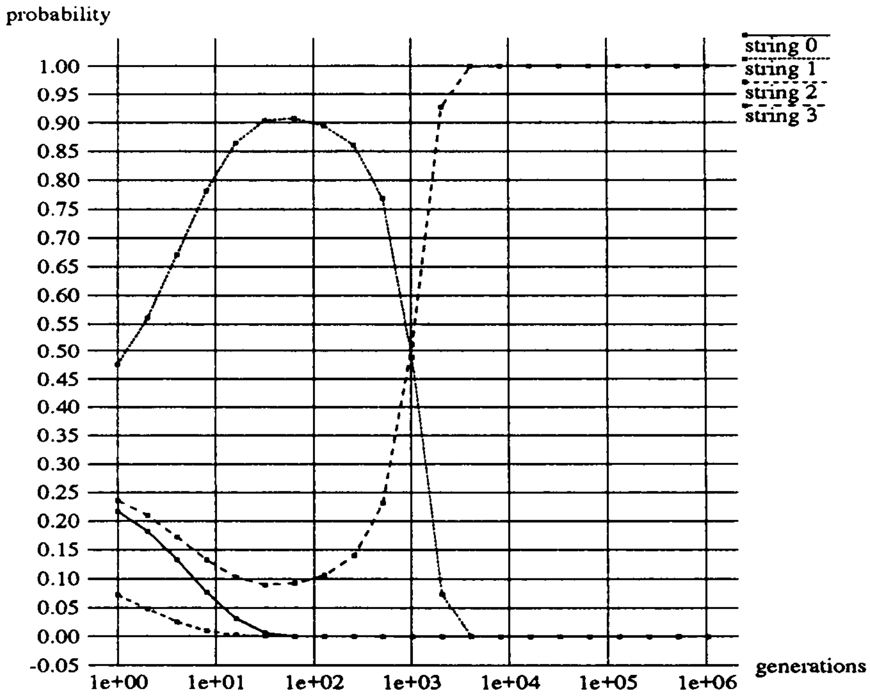
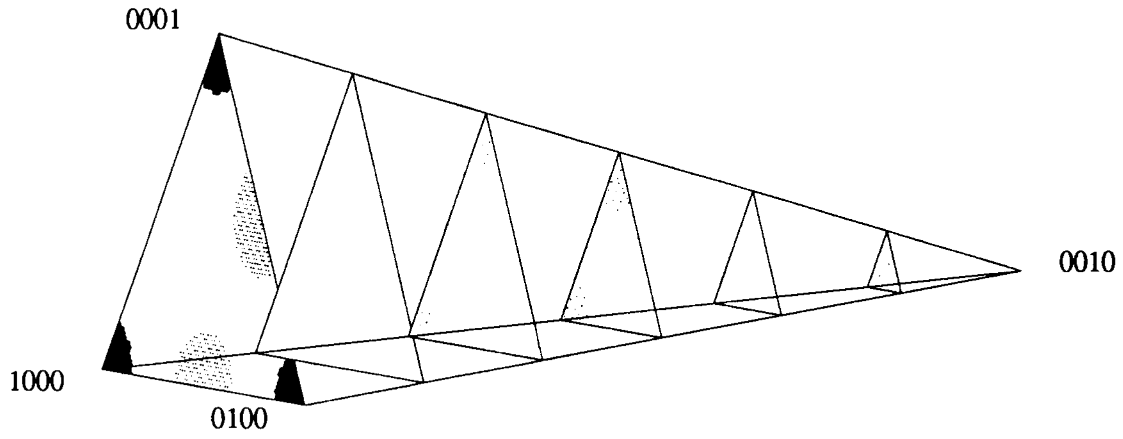
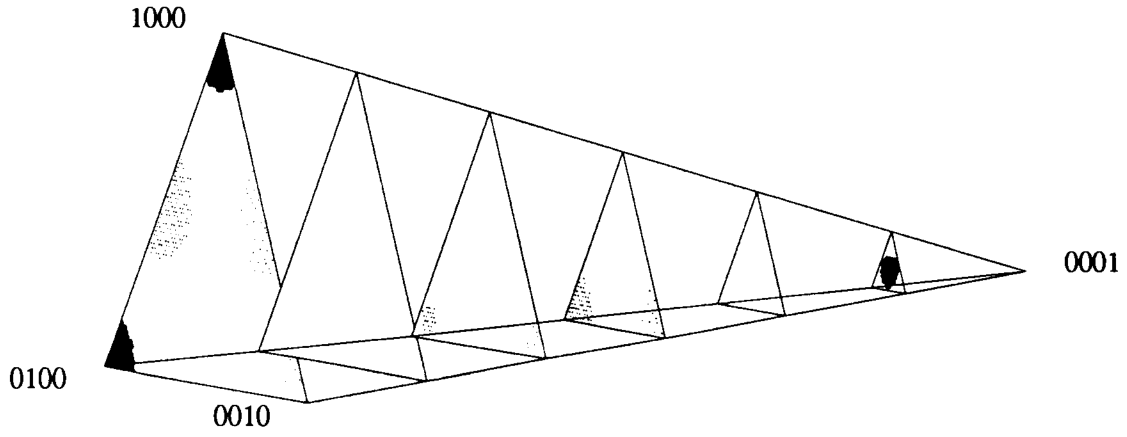


Figure 38. Probability of Encountering String i in k Generations
for Infinite Population, Type 1, Mutation 0

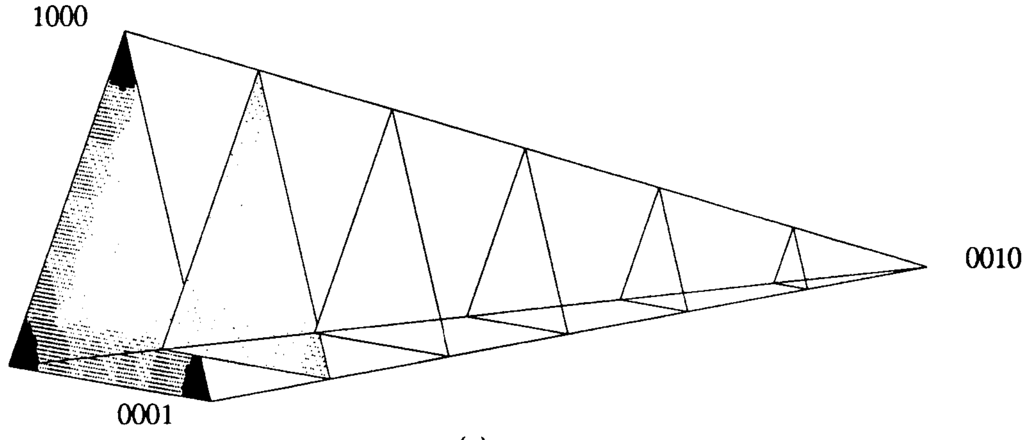


(a)

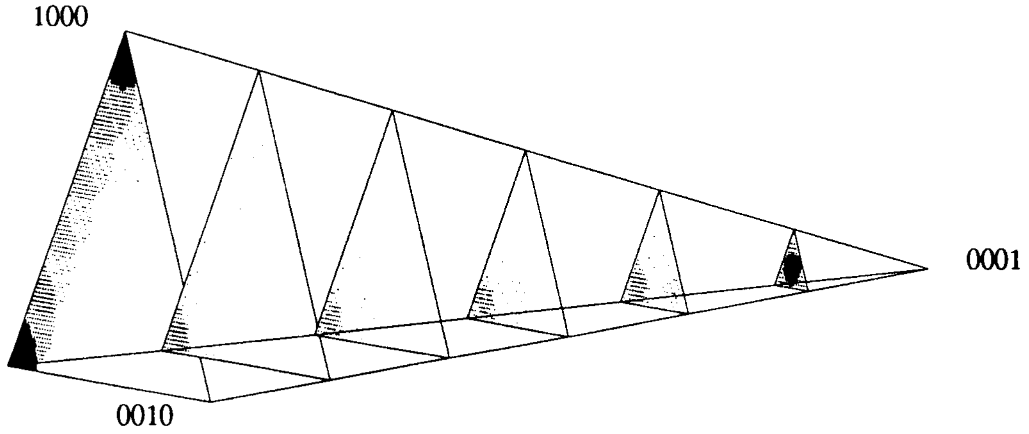


(b)

Figure 39. Probabilities of Populations within Radius r of Given Population Coordinates for Population 2, Type 1: (a) view 1, (b) view 2

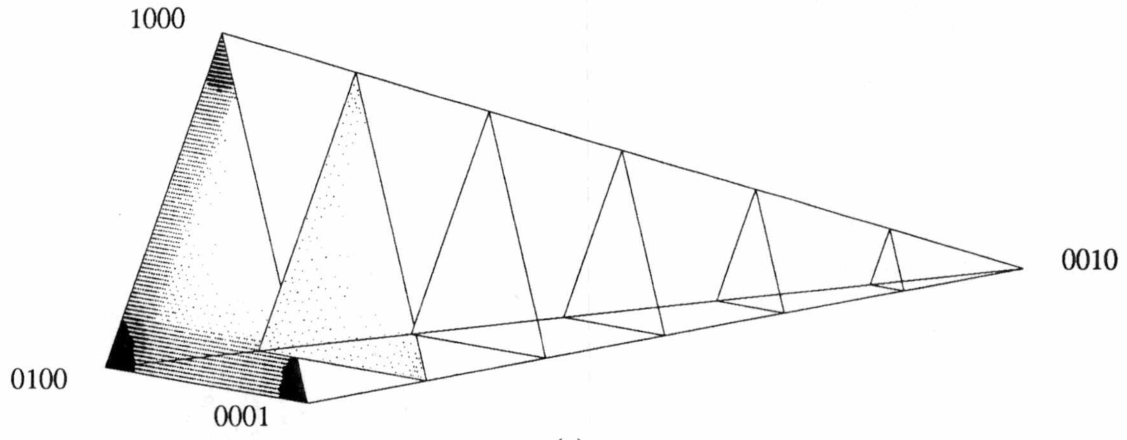


(a)

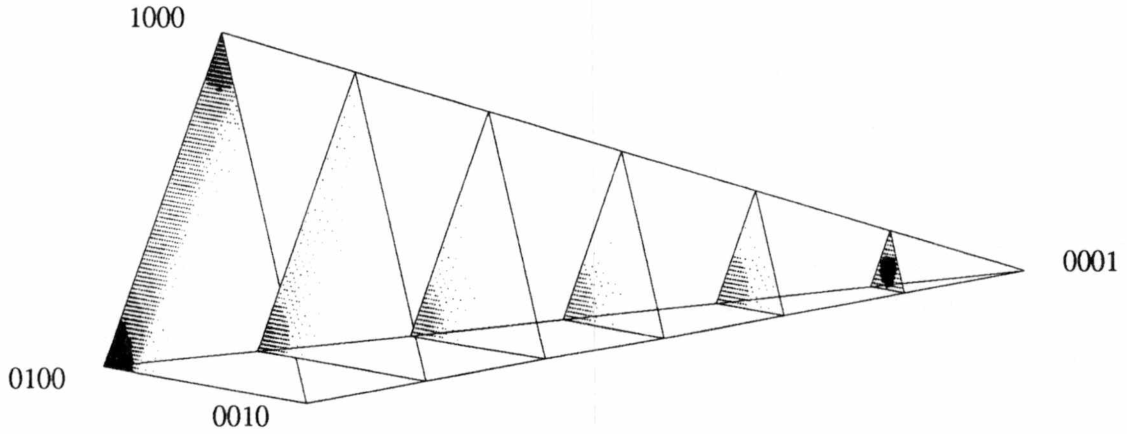


(b)

Figure 40. Probabilities of Populations within Radius r of Given Population Coordinates for Population 6, Type 1: (a) view 1, (b) view 2

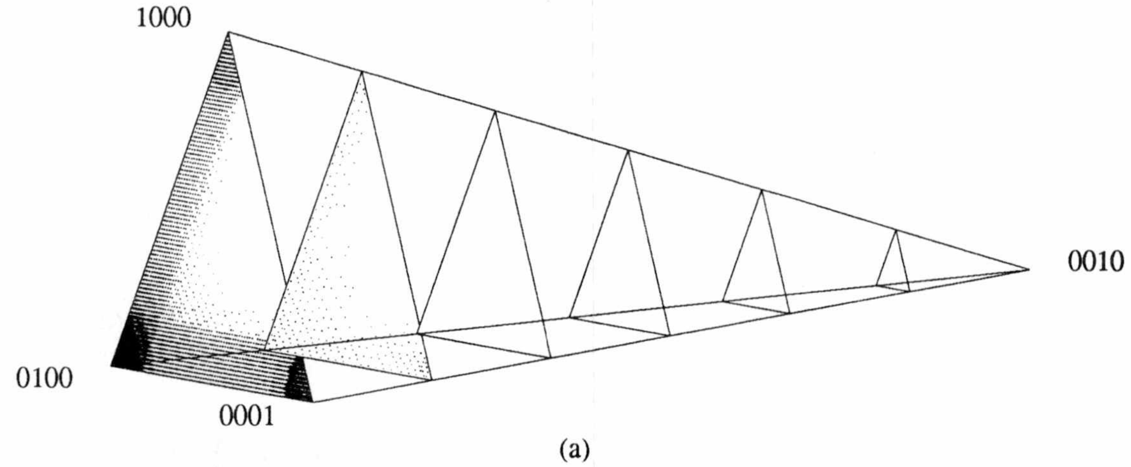


(a)



(b)

Figure 41. Probabilities of Populations within Radius r of Given Population Coordinates for Population 10, Type 1: (a) view 1, (b) view 2



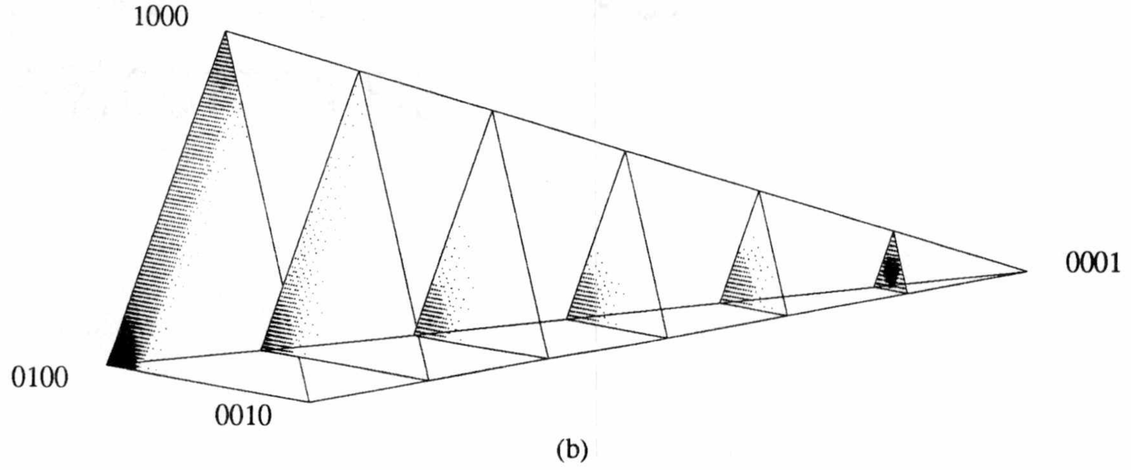
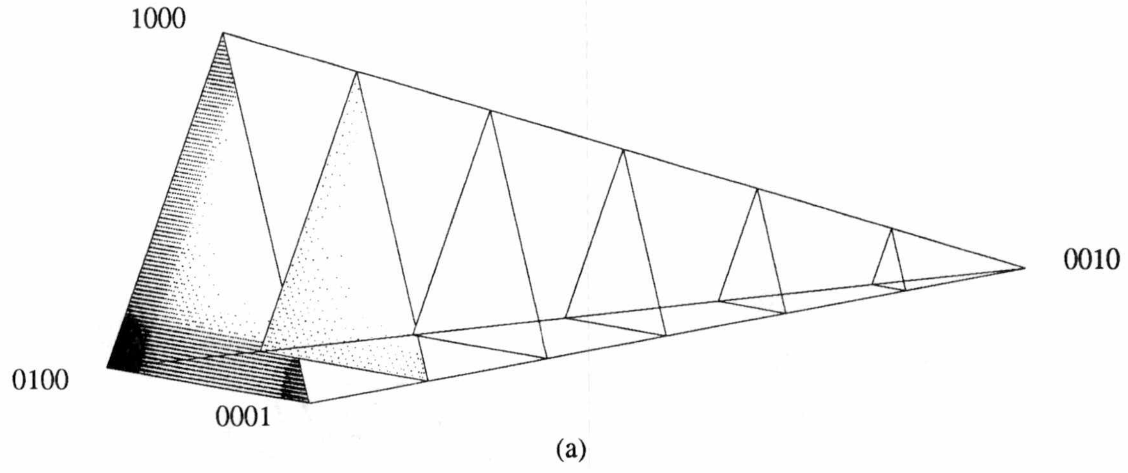


Figure 42. Probabilities of Populations within Radius r of Given Populat ion Coordinates for Population 14, Type 1: (a) view 1, (b) view 2



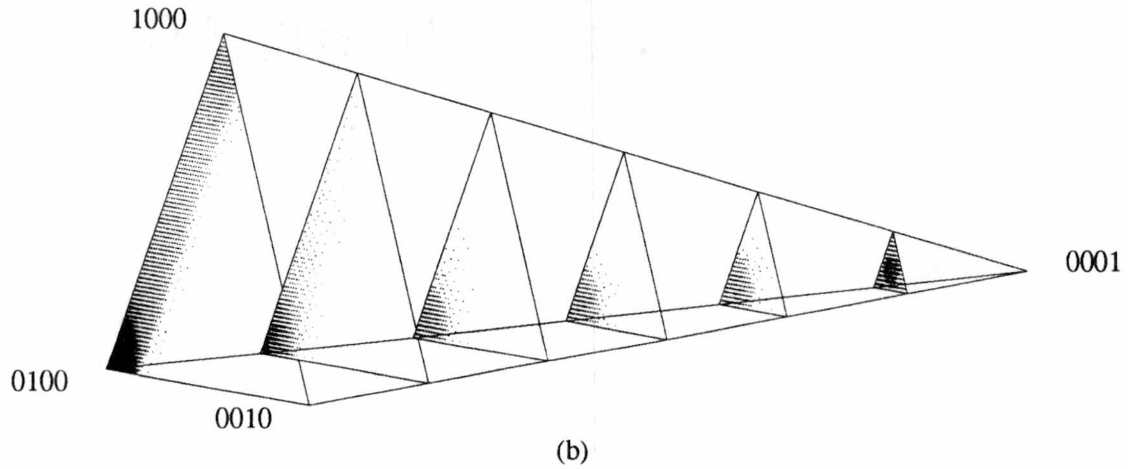
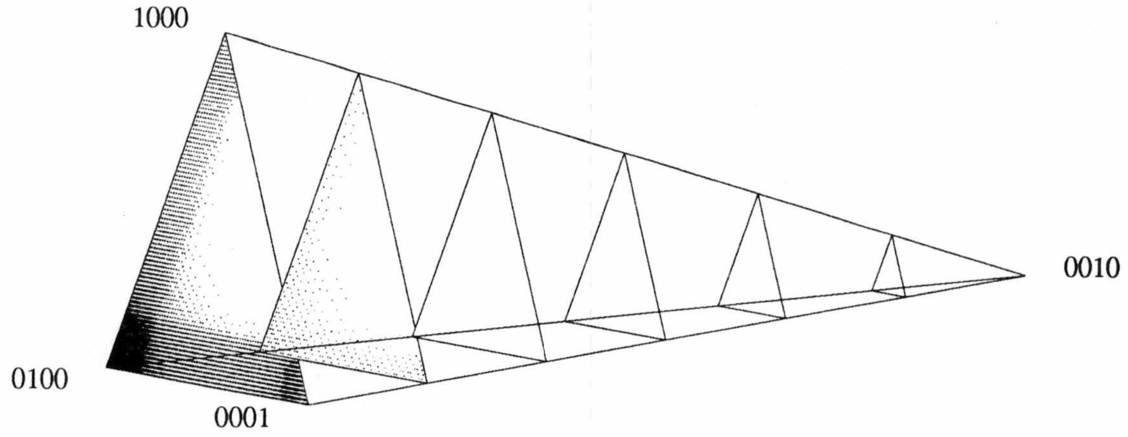
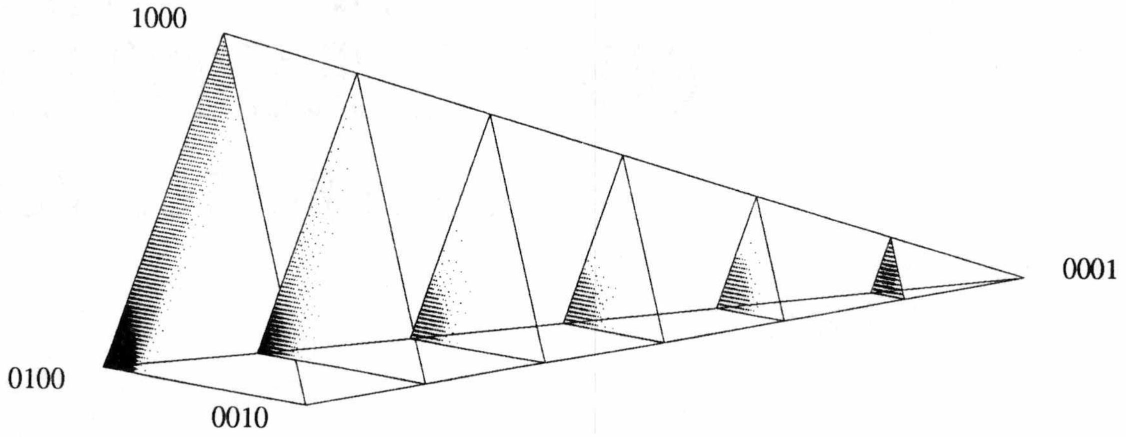


Figure 43. Probabilities of Populations within Radius r of Given Populat ion Coordinates for Population 18, Type 1: (a) view 1, (b) view 2



(a)



(b)

Figure 44. Probabilities of Populations within Radius r of Given Population Coordinates for Population 22, Type 1: (a) view 1, (b) view 2

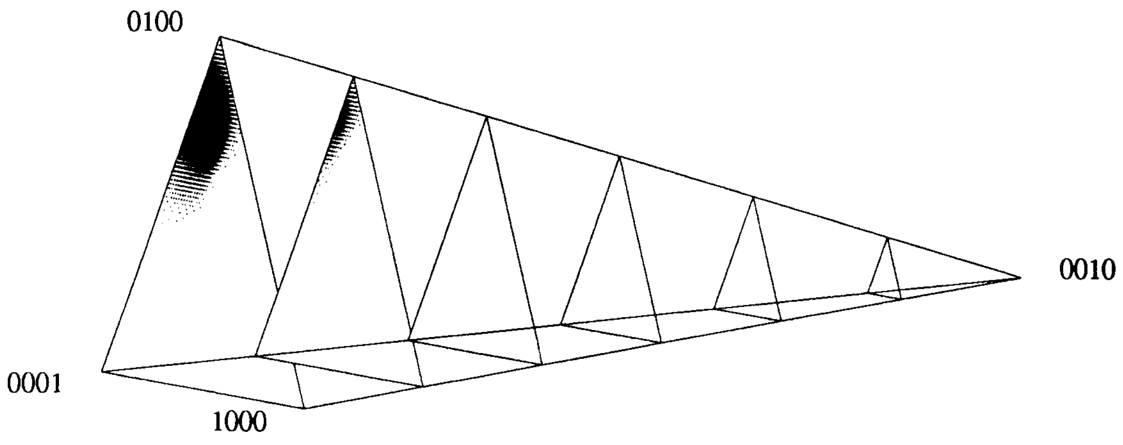
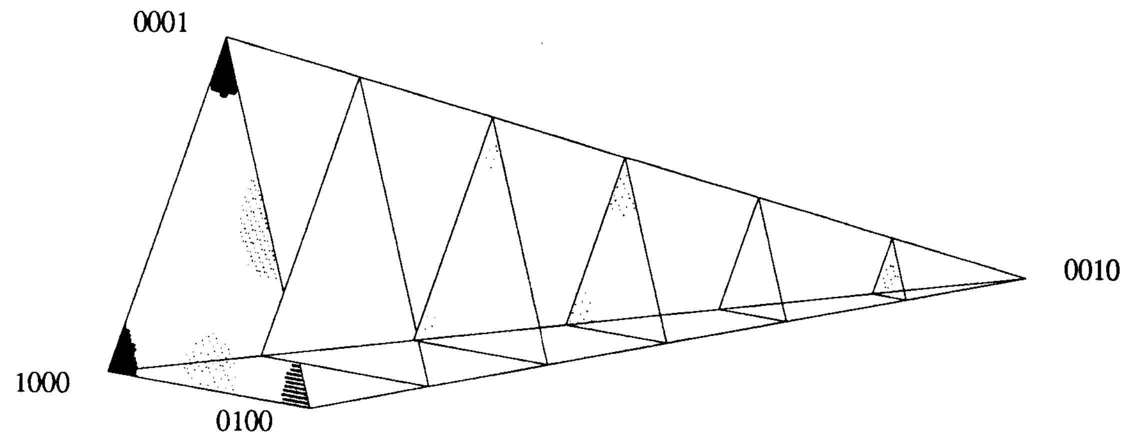
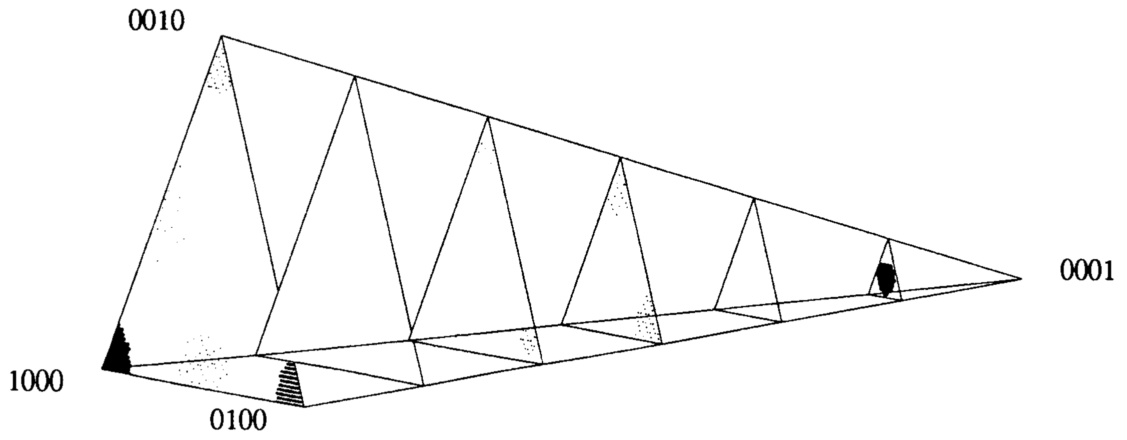


Figure 45. Probabilities of Populations within Radius r of Given Population Coordinates for Population 1000, Type 1

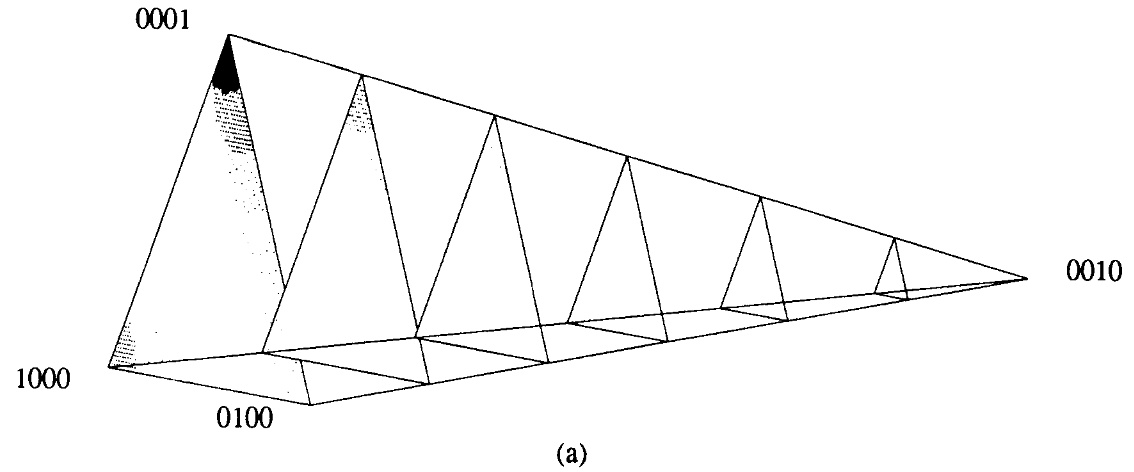


(a)



(b)

Figure 46. Probabilities of Populations within Radius r of Given Population Coordinates for Population 2, Type 2: (a) view 1, (b) view 2



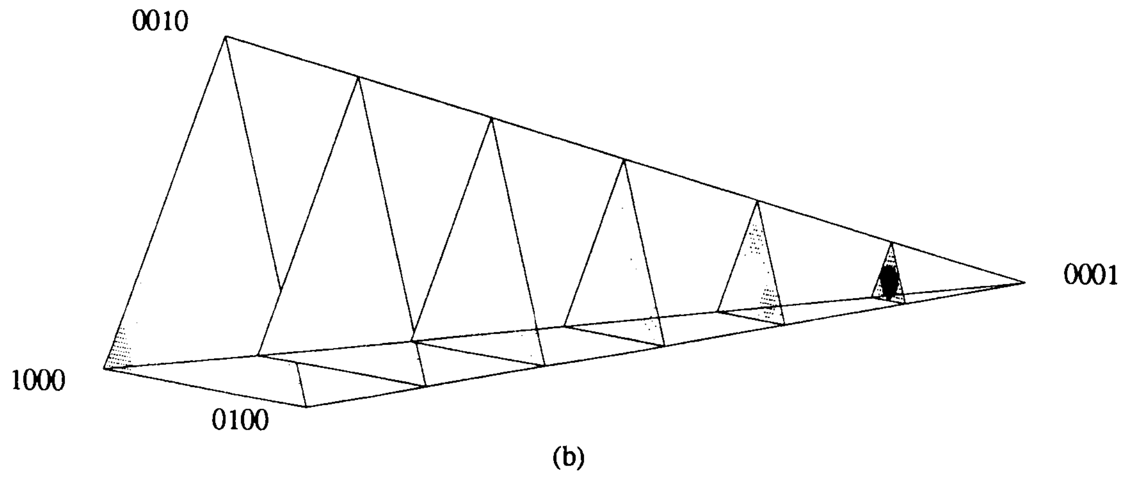
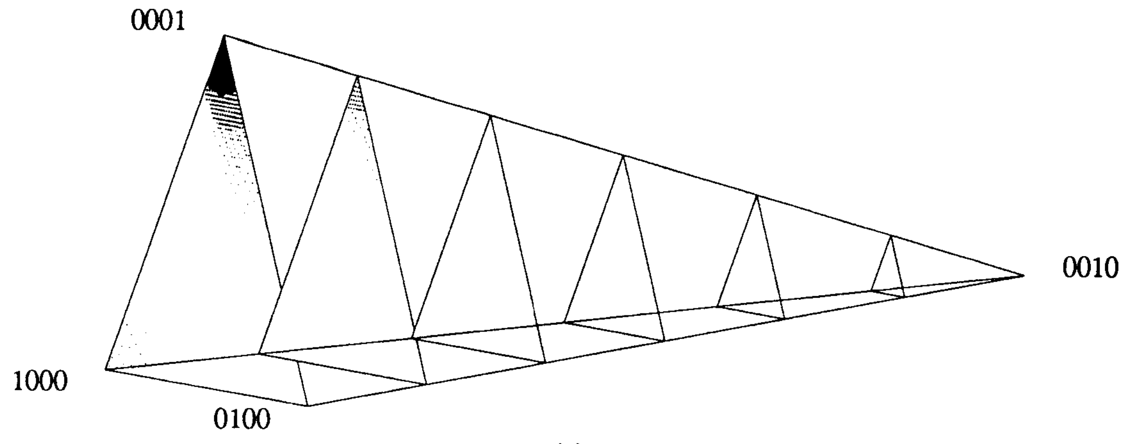
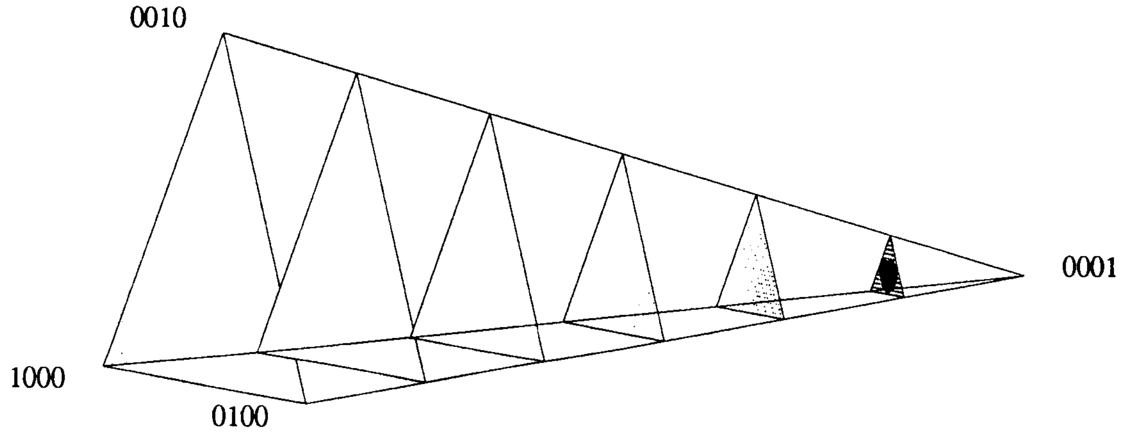


Figure 47. Probabilities of Populations within Radius r of Given Population Coordinates for Population 6, Type 2: (a) view 1, (b) view 2



(a)



(b)

Figure 48. Probabilities of Populations within Radius r of Given Population Coordinates for Population 10, Type 2: (a) view 1, (b) view 2

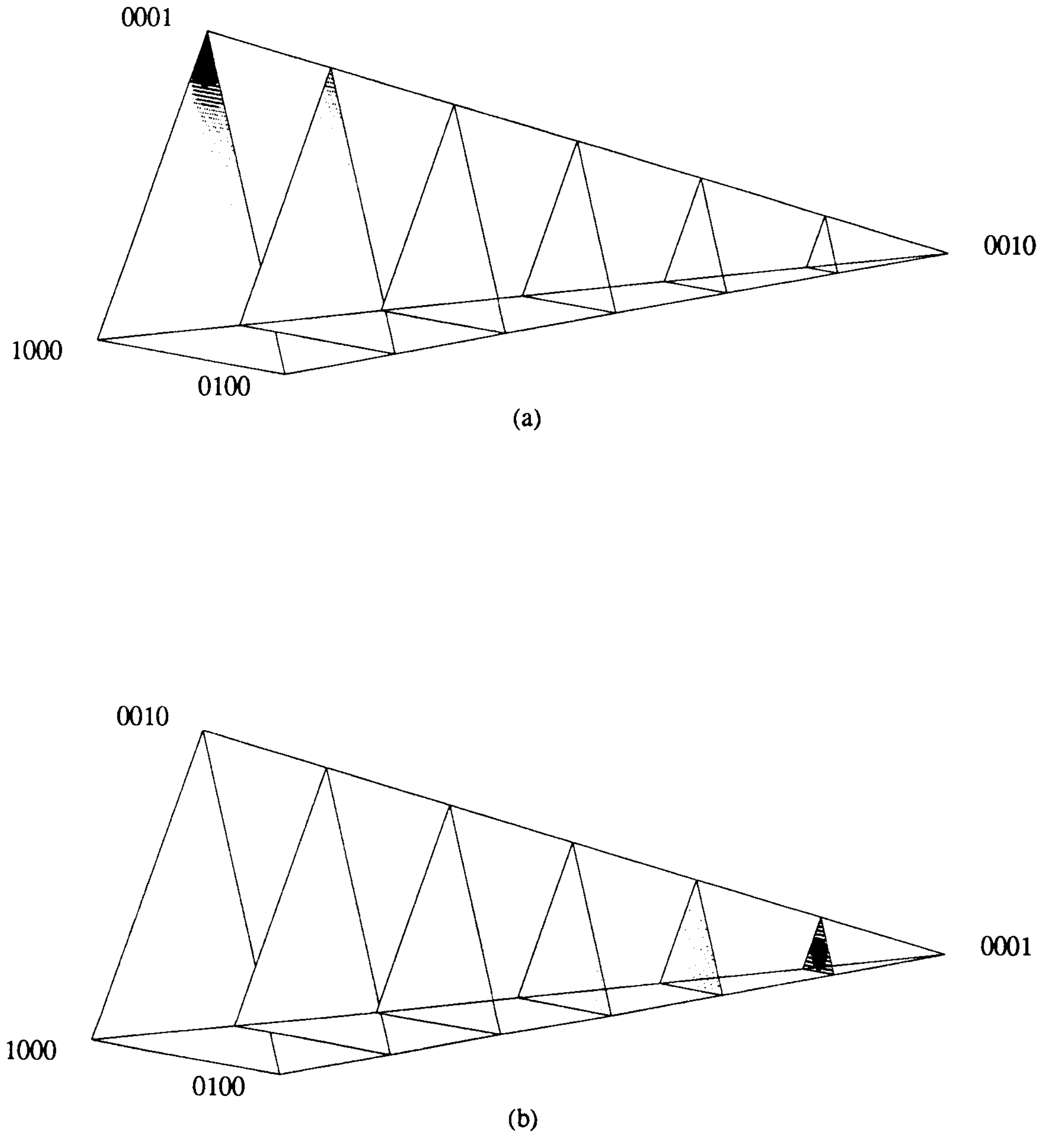
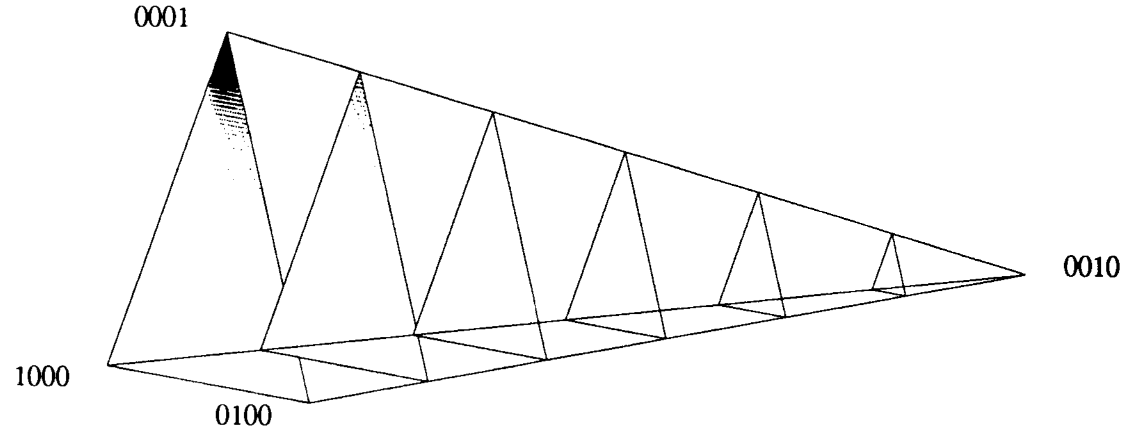
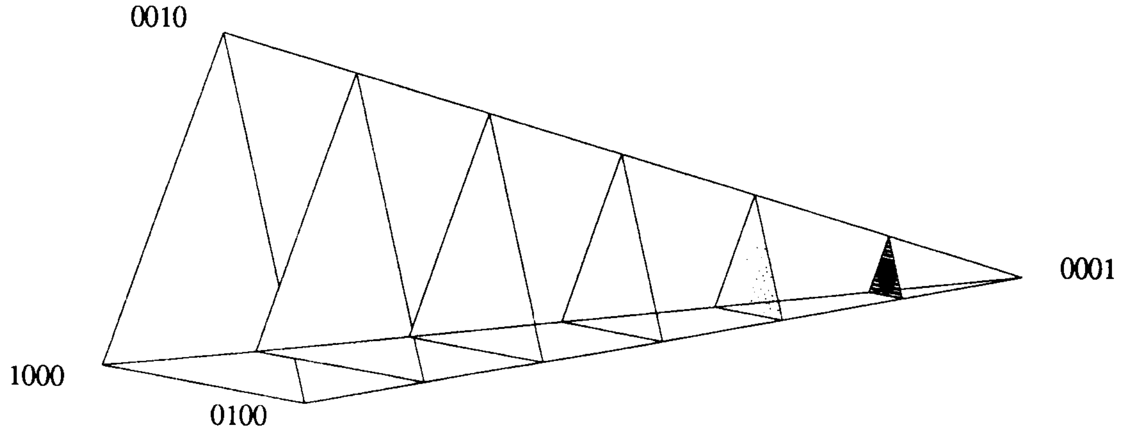


Figure 49. Probabilities of Populations within Radius r of Given Population Coordinates for Population 14, Type 2: (a) view 1, (b) view 2



(a)



(b)

Figure 50. Probabilities of Populations within Radius r of Given Population Coordinates for Population 18, Type 2: (a) view 1, (b) view 2

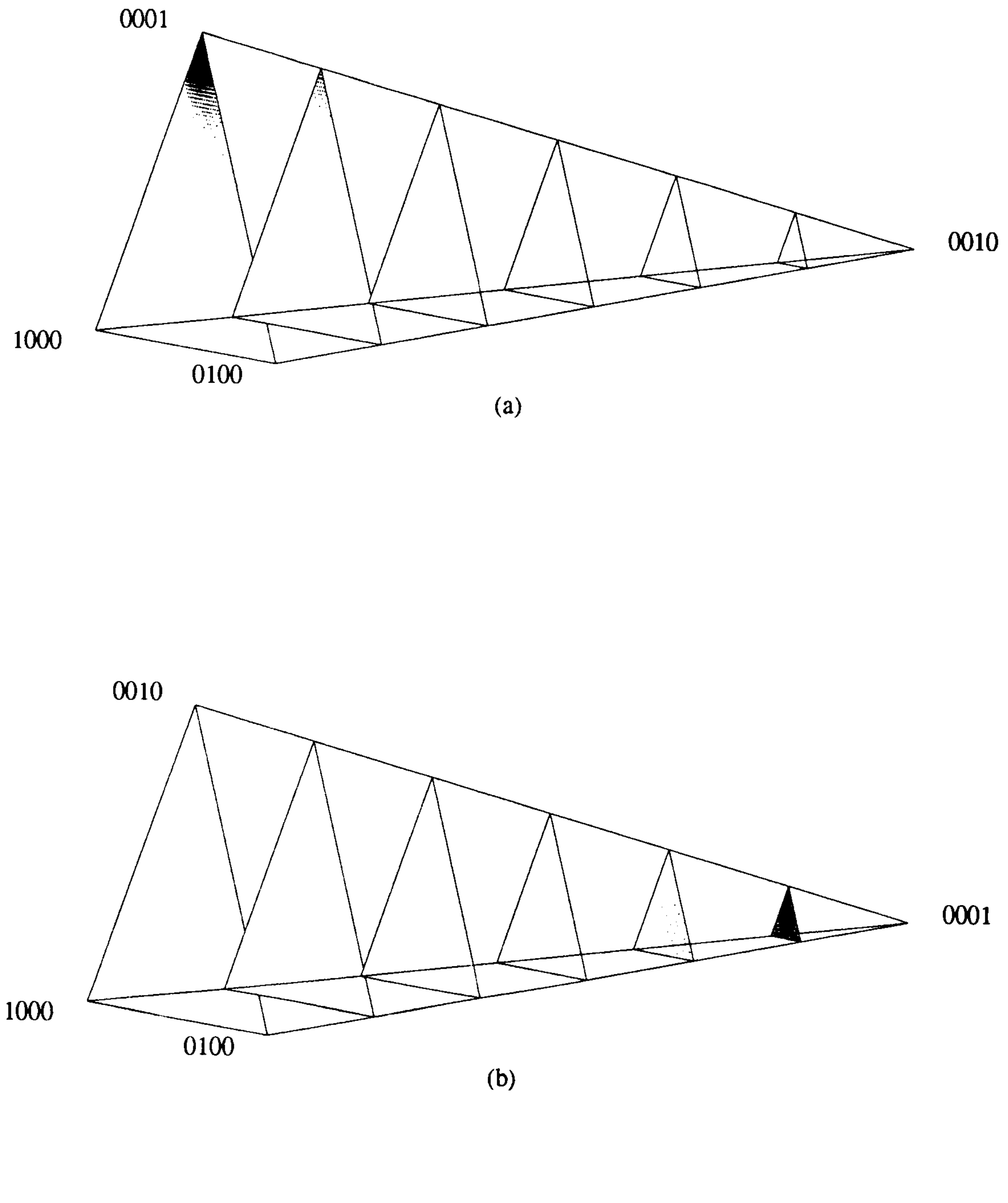


Figure 51. Probabilities of Populations within Radius r of Given Population Coordinates for Population 22, Type 2: (a) view 1, (b) view 2

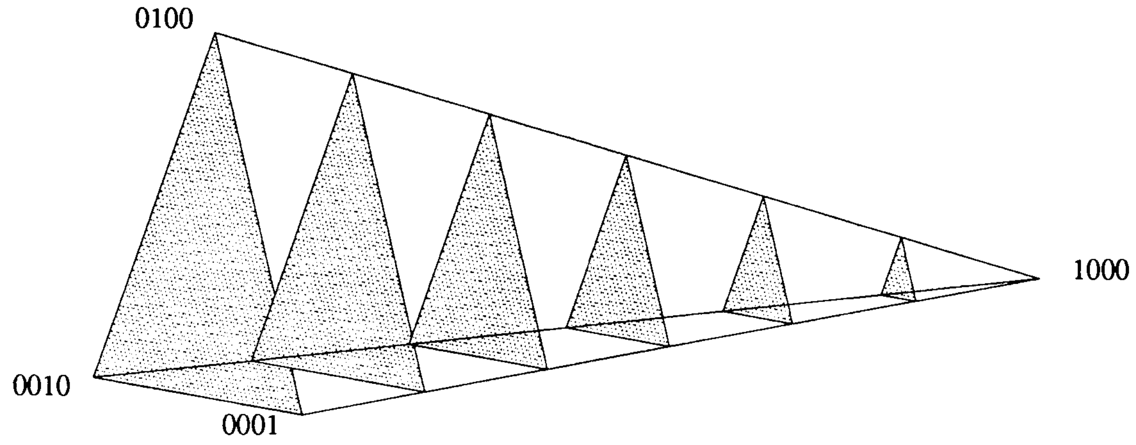


Figure 52. Basin of attraction for Type 1 problem

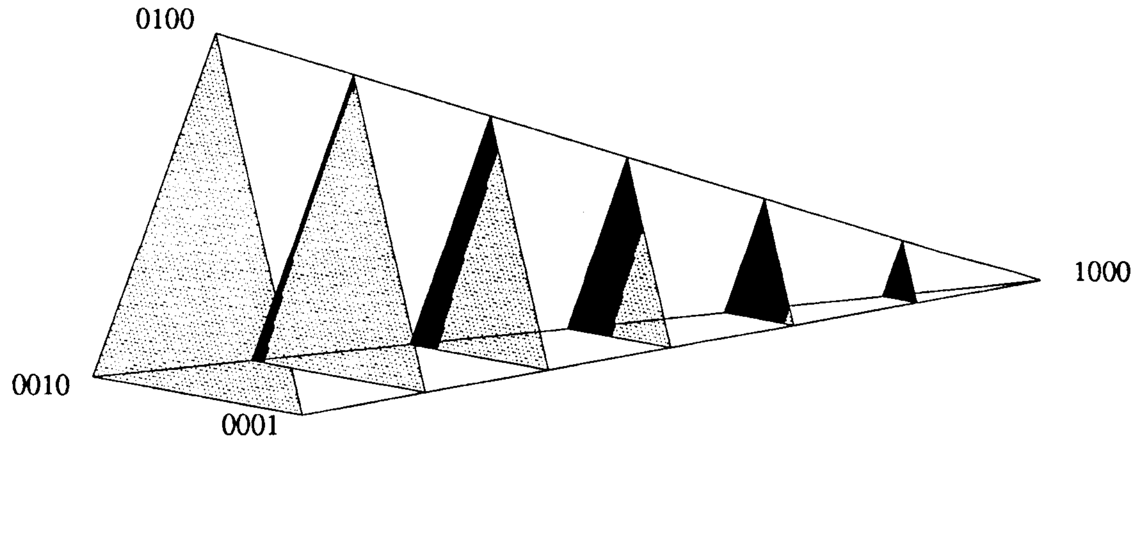


Figure 53. Basin of Attraction for Type 2 Problem

## VITA

Allen Eugene Nix was born in Newport, Tennessee on February 17, 1956. He attended elementary schools in Cocke County and graduated from Cocke County High School in 1974. He received an ROTC scholarship and entered East Tennessee State University the following September. On March 13, 1976 he married Terry Lee Owen of Knoxville, Tennessee. He received a Bachelor of Science degree in Biology in June 1978 and was commissioned a 2nd Lieutenant in the Signal Corps, United States Army. He was assigned to Fort Bragg, North Carolina where he served with the 35th Signal Brigade and the 82nd Airborne Division. In 1982, he was promoted to Captain and was transfered to Quantico, Virginia where he attended the Marine Corps Advanced Communications Officers Course. In 1983, he was transfered to Elefsis, Greece where he served with a NATO Artillery Group. In 1985, he was transfered to Fort Hood where he served with the 13th Signal Battalion, 1st Cavalry Division. In 1989, he entered the University of Tennessee and in August, 1991 received a Masters of Science Degree in Computer Science. In 1990, he was promoted to the rank of Major. During his military career, he has worked with automated tactical switching, single and multichannel communications, cabeling systems, and repair of tactical equipment. He has served as a Platoon Leader, Company Commander, Brigade Communications Officer, Assistant Division Signal Officer, and Battalion Operations Officer. His service has required him to visit ten states and the countries of Germany, Greece, Italy, Turkey, and Belgium. In August, 1991, he will travel to South Korea to work as a Data Processing Officer.

He presently has five children, Jeremy, Brian, Jessica, Sarah, and Mackenzie who range in ages from nine months to ten years.