To the Graduate Council:

I am submitting herewith a thesis written by Paul Scott Hethmon entitled "Empirical Observations on the Genetic Algorithm." 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 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:

Associate Vice Chancellor
and Dean of the Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature 

Date November 30, 1993

Empirical Observations on the Genetic Algorithm

A Thesis

Presented for the

Master of Science Degree

The University of Tennessee, Knoxville

Paul Scott Hethmon

December 1993

Dedication

This paper is dedicated to my wife, Marci, and to my son, Zachariah, who saw far too little of me.

Acknowledgments

This research was funded by a grant from the National Science Foundation. Grant number CDA-9115428.

I would like to thank Dr. Vose for his support through this process. I would also like to thank my committee, Dr. Mutchler and Dr. Straight.

Abstract

This thesis presents some observations from a large empirical study of the behavior of genetic algorithms. Specifically, comparisons between a model of theoretical behavior and actual observed behavior will be presented.

Contents

List of Figures

List of Tables

Chapter 1

Observations on the Genetic Algorithm

This paper looks at the behavior of the simple genetic algorithm against that predicted by a model. The particular implementation of genetic algorithms used is based on a framework designed by Battle [1] and the model is based on work done by Vose [2].

1.1 The Simple Genetic Algorithm

Genetic algorithms mimic the behavior of genes in organisms in an attempt to optimize some function. Genetic algorithms operate on populations of strings. These strings in this study consist of the binary digits 0 and 1. Each string is associated with a fitness determined by a fitness function which describes its "fitness" with respect to the other strings.

A step in producing the next generation consists of selecting two strings (a and b) based on their fitness to be parents to the new generation. A crossover point is selected

in the strings (one point crossover with a rate of 0.8 was used in this study) and the first part of string a is joined with the second part of string b. Likewise the second part of a is joined with the first part of b. At this point mutation is performed on the individual bits of the new children. Each bit is considered individually and based on a preset mutation rate (0.01 in this study) is either mutated or not. Mutation simply means flipping the bit to its binary complement.

The two new strings are then stored in the new population and the process is repeated until the new population has all of its members. This completes the formation of the next generation of the genetic algorithm.

The reader may wish to look at Goldberg [3] for more information on genetic algorithms.

1.2 The Theoretical Model

The theoretical model used in this study attempts to predict the behavior of the genetic algorithms. The model is simply a smooth function *inf* which takes as input a vector describing the members of the population and gives as output its expected next population. The vector is of length 2^l where l is the string length. The ith component of the vector is a number denoting the percentage of the population which are strings which can be interpreted as the binary number i. The component labelling starts from 0 and extends to $2^l - 1$. This corresponds to all possible strings.

The model's general behavior is convergence to *fixed points*. A fixed point is defined to be $inf(p) = p$ where p is a population vector. In this study, $inf(p) = p$ was defined to be true when all components of $inf(p)$ and p were within some tolerance factor $t = 1e - 10$.

1.3 The Study

In order to reduce the variation inherent in the behavior of genetic algorithms, a large number of test cases were conducted. The tests were broken up in a hierarchal fashion with the first parameter varied being the string length. String lengths from 3 to 7 were run. Next, the population size under each string length was varied. Population sizes used were 16, 32, 64, 128, 256, and 512. Under each of these population sizes, ten different fitness functions were used. Under each fitness function, ten different random seed values for the random number generator were used. Figure 1.1 highlights this breakdown.

Once these tests were conducted, the data was analyzed and the following questions arose:

1. How many fixed points are there?
2. How close are the ga's to the fixed points?
3. How far away are the ga's from the fixed points?
4. How many times does the ga change which fixed point it is close to?

Based on these questions, the data was analyzed more rigorously and equations were developed to model the behavior seen.

The first question is obvious as well as the analysis; however the last three are based on a concept of *distance* from a special fixed point. The distance d is calculated by the following formula where v is a vector representation of the population, f is the vector representation of the special fixed point and l is the string length:

$$d = \sum_{i=0}^{2^l} (v[i] \ln(\frac{v[i]}{f[i]})$$

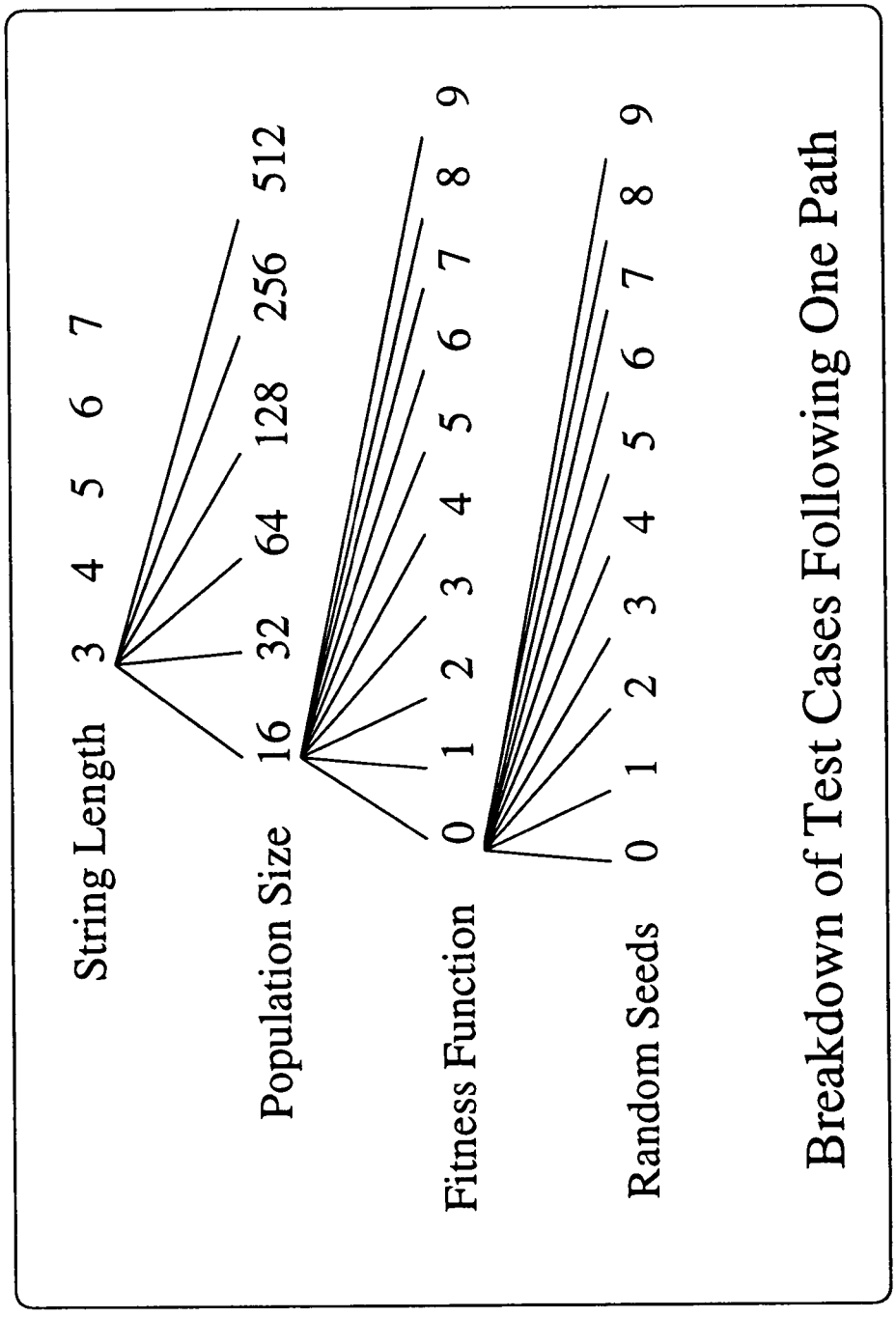


Figure 1.1: Test Cases Breakdown

Based on this formula, two dimensional graphs were produced for all test cases for visual inspection. From this the questions were formulated and then the data analyzed to develop the equations describing the behavior. Once this initial analysis of the data was over, the algorithms were run again on a higher string length to verify the results.

Chapter 2

Code Implementation

2.1 Serial Version

The serial version of the code was developed by Vose [2]. It incorporated a genetic algorithm framework by Battle [1], along with new work.

The goal of this program is to compare what we think the *ga* will do to what it actually does. The algorithm by Vose known as *inf* will search for fixed points in theoretical ga behavior using a quick method which manipulates the populations as floating point vectors. Once *inf* has found the fixed points, we compare this with *ga* behavior searching from the same initial population.

The breakdown of the algorithm is fairly straightforward. First using parts of Battle's framework, a user supplied parameter file is read which controls several parameters of execution: string length, population size, number of generations, mutation rate, crossover rate, fitness function, etc. Once the program has initialized data structures, it starts searching for fixed points using *inf*. For each run of *inf*, a random population is generated and then the transition to the expected next population is

iterated until a fixed point is reached.

When a fixed point has been found by *inf*, the program checks first to see if that fixed point has already been located. If not, it stores the fixed point and the initial population that reached it. If the point has already been found, the information is used to calculate the size of the basins of attractions for the other fixed points.

Once *inf* has been run the number of times requested by the user, the program then finds the special fixed point and reorders the fixed points according to this. The special fixed point is the one having the largest basin of attraction. After this is finished, the program then runs two routines on each initial population. The first is a modified version of *inf* which saves the distances from the special fixed point to the current population as it iterates the population through the expected transitions to the fixed point. The second is *ga* which has been modified from Battle's original to run as a subroutine and save the distance from the current population to the special fixed point. In this way, the theoretical behavior can be compared to the actual behavior.

With this information saved for each fixed point, the program writes out a parameter file with the needed information to graph the results. A separate program reads this parameter file and the information from each fixed point to produce a two-dimensional graph showing a pair of lines for each fixed point. The x axis represents generations and the y axis the distances. The first line represents the *inf* algorithm and shows where we think the *ga* will go. The second represents the *ga* and shows where it actually went. All fixed points are graphed together to obtain a clear picture of what is happening.

Because this is a serial program, running times are long for significant string lengths, populations sizes, and generations. If a user wants to start with 100 ini-

tial populations for a search space, the program must run 100 consecutive searches. Once that is done, for each fixed point found, the *ga* must run, then *inf* must run to save the data for display and study. All of this adds up to a significant amount of time.

It can easily be seen that many of the tasks in the program are not dependent and could be done in parallel. Each of the initial populations used is independent of all the others and could be done simultaneously given the processors. Likewise, each run of genetic and inf is independent and could be run simultaneously. These observations lead to the parallel version of the program.

2.2 Parallel Version

The first thing to be said about the parallel version of the program is the target architectures. The first machine used was an Intel iPSC/860 hypercube with 128 processors. Each node has 8 megabytes of memory and an i860 processor. The machine is located at the Oak Ridge National Labs. The second machine used was a Thinking Machines CM-5 with 32 processors. Each node has 32 megabytes of memory, a sparc processor, and a vector processor. This machine is located at The University of Tennessee.

The program is based on the obvious parallel divisions mentioned in the previous section. The first thing done was to divide the program into two programs. The first program is the master program. The master does no real processing, it only correlates the data and assigns tasks to other processors. The other processors act as slaves and run the second program which does the actual data processing.

By dividing the work up into a master-slave relationship, problems with distribut-

ing information globally have been avoided. There is no need for the different nodes to keep all of the data, the master node is responsible for managing the data and sending the necessary parts to each slave node as needed. As such, after an initial start-up phase to send information to the slave nodes, very little data is sent between nodes. The only data necessary to be sent after the start-up are: 1) the found fixed points from each slave node to the master, 2) the list of fixed points from the master to each slave, and 3) the information needed to assign a *ga* or *inf* task to a slave node.

The execution of the master program procedes as follows. The master node reads in the base user supplied parameter file. This file is inherited from Battle's *ga* framework and has been expanded to include some parameters needed for the parallel implementation. Once the master node has read in this parameter file, the information is packed into buffers and sent to the slave nodes. Only three messages are sent, one for each type of data included in the parameter file: character strings, integers, and floating point numbers. These messages are sent using a global broadcast for efficiency.

At this point, the slave nodes can begin some preliminary initialization work. Basically setting up buffers for later calculations. The master node, however, starts reading in more parameter files. The first to be read in is the fitness function. The fitness function is stored in vector format and is sent to each node in a global broadcast. This message is also sent asynchronously (non-blocking send) to free up the master. Next, the master reads in a random seeds file. This file contains an entry for each node for their random number generator. This way, the user can specify unique seeds for each node. These values are also stored in vector format and the entire vector is sent to each node to reduce processing that would otherwise have to

occur. The slave nodes find their seed value by keying into the vector with their node identity.

The last bit of initialization data sent to the nodes is the number of starting populations they should generate to search for fixed points with *inf*. This is the part of the code that actually starts the parallel work. By breaking up the set of starting populations over the slave nodes, we can see an improvement in throughput. The number sent to each node is calculated so that no single node has more than one extra from any other. Specifically, if we want to search from 100 initial populations and have 8 processors, one processor is the master, 5 processors each search from 14 initial populations and 2 processors each search from 15 initial populations to reach our total of 100. After this information is sent, the slave nodes complete their initialization and start the search for fixed points.

Before the master node can receive any data, it must complete its initialization. After that is done, the master node goes into a loop receiving the fixed point information from the nodes. When the master node receives an information packet from a node, it takes this information and checks to see if it is a new fixed point or not. If it is a new fixed point, the master node saves the starting population and the found fixed point for later processing. The initial population is also written to file for the *ga* to process. If the fixed point has already been found, the information is used to approximate the size of the basins of attraction of the fixed points.

Once an initial population/fixed point pair has been received for each requested, the master node calculates the special fixed point and reorders the fixed points to send out to the nodes. The array of fixed points is sent to all of the nodes at this time.

The next thing done is to send tasks off to nodes. Two tasks are generated for

each fixed point found. The first is to rerun the *inf* routine and save the information about where the fixed point is going for later graphing. The second is to run the *ga* with the saved initial population corresponding to the fixed point. This also saves information to graph the *ga* later against the fixed point.

By assigning these tasks to different nodes, we are able to gain a speedup over serial processing of the algorithm. If the number of tasks exceeds the number of nodes, then some nodes must run multiple tasks. This part of the algorithm scales well with the increase in problem size. The larger the problem size, the more processors that can be used effectively.

Once all nodes have reported back finished with their tasks, the master node tells each node to exit. The master node finally writes an information file to disk with the parameters needed to graph the information and then exits.

This version of the program works well for problems up to a certain size. In particular, a string length of 8 with a reasonable population size (32) and a search space of 100 will run to completion in approximately 1.5 hours using 16 nodes. With larger string lengths, this limitation becomes more apparent. The same problem with a string length of 9 will take approximately 7.5 hours to complete using 32 nodes, and with 16 nodes the time is approximately doubled.

The major contributing factor to this is that only one processor works with a fixed point at a time. That is, when all of the fixed points are found, one processor is assigned to run the genetic algorithm on it. So with large problem sizes, we still overload the processor.

However, for small problems sizes, the algorithm can find its answer quickly and the division of the parallelism is straight forward.

Chapter 3

Number of Fixed Points Found

The first category explored was that of the number of fixed points found. This information was simply gleaned from each individual test case. We look at it two ways.

3.1 From the Test Cases

The first way the information is compiled reflects the number of fixed points found based on string length and population size. This averages the data from the different fitness functions and random seed values as shown in Figure 3.1. This generally shows an increase in the number of fixed points as population size decreases and string length increases.

3.2 Looking at the Total Number Found

Of equal interest is how many *different* fixed points did we actually find within a string length. To see how well the different test cases did at finding *all* the fixed

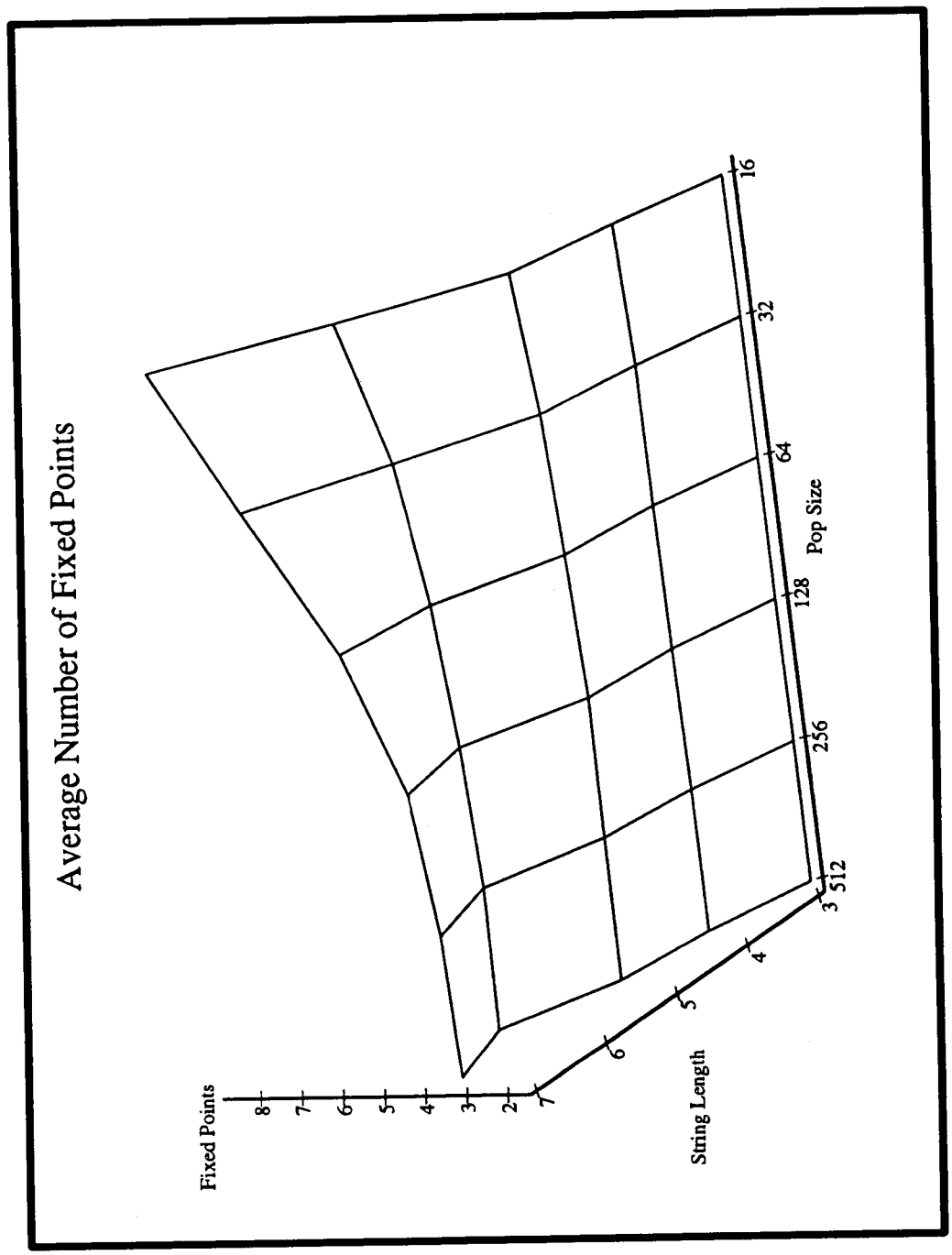


Figure 3.1: Average Number of Fixed Points

points, a special run of the code was made using all of the saved initial populations as base populations for *inf* to iterate from. From these results, Figure 3.2 corresponding to the total number of different fixed points was produced. This shows how many different fixed points were actually found in the original test cases.

With this information in mind, we next look to deriving an equation to predict the number of different fixed points *inf* will find based on the string length. Figure 3.3 shows the behavior of Figure 3.2 with the number of fixed points plotted as the ln of the number of fixed points. This yields us a fairly straight line and leads us to try an equation of the form

$$f = ke^{\alpha l}$$

where f is the number of fixed points, k is the intercept, α is the slope, and l is the string length. So now we go to

$$\ln f = \ln k + \alpha l$$

and using linear regression, we solve for α and k and get

$$\alpha = 0.3944275048$$

$$\ln k = -0.160699686$$

$$k = 0.8515477644$$

From this, we now have an equation to predict the total number of different fixed points we will find based on string length. Figure 3.4 shows the graph of this.

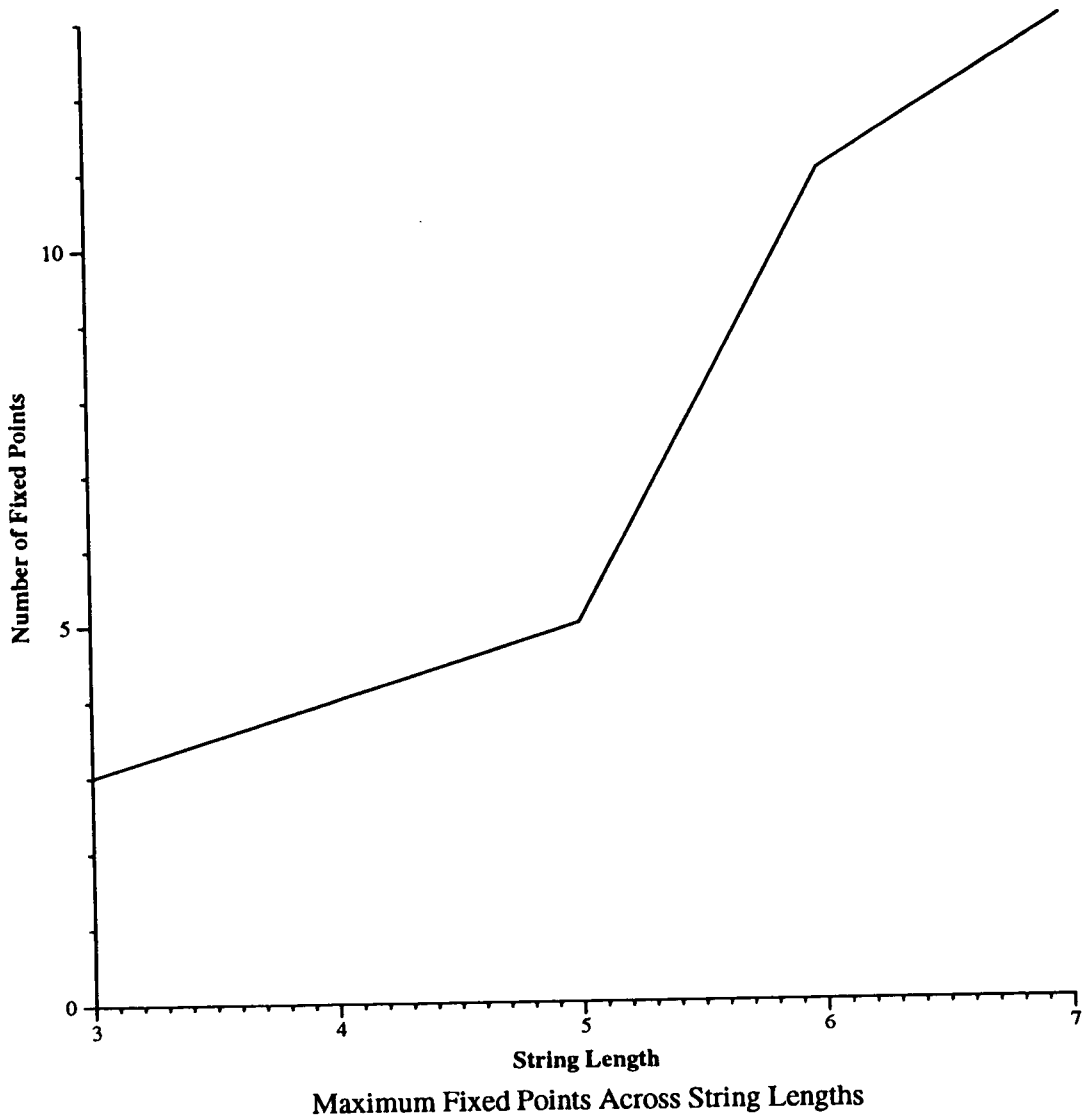


Figure 3.2: Revised Maximum Number of Fixed Points

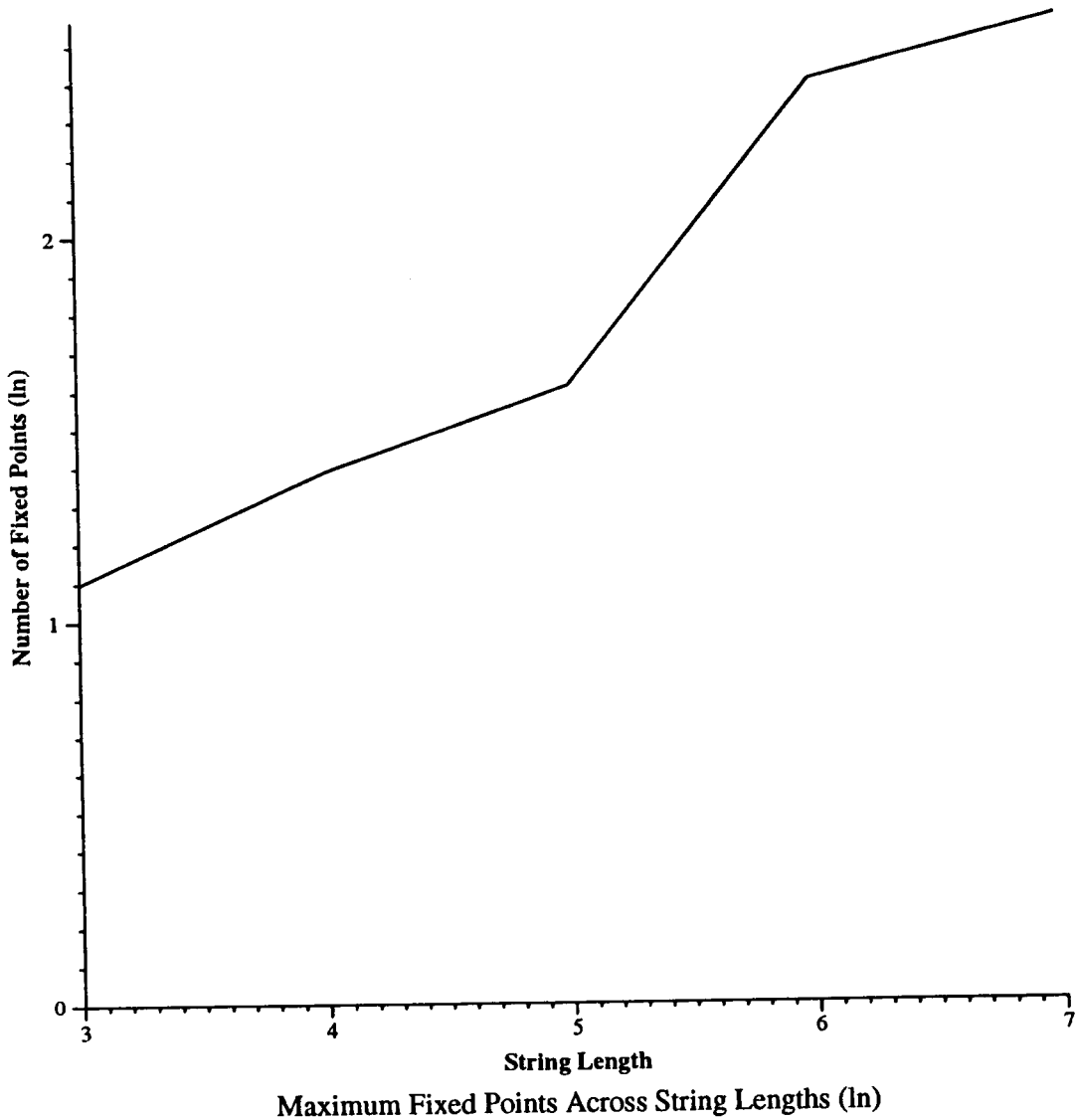


Figure 3.3: Behavior of Revised Maximum Fixed Points with ln Scale

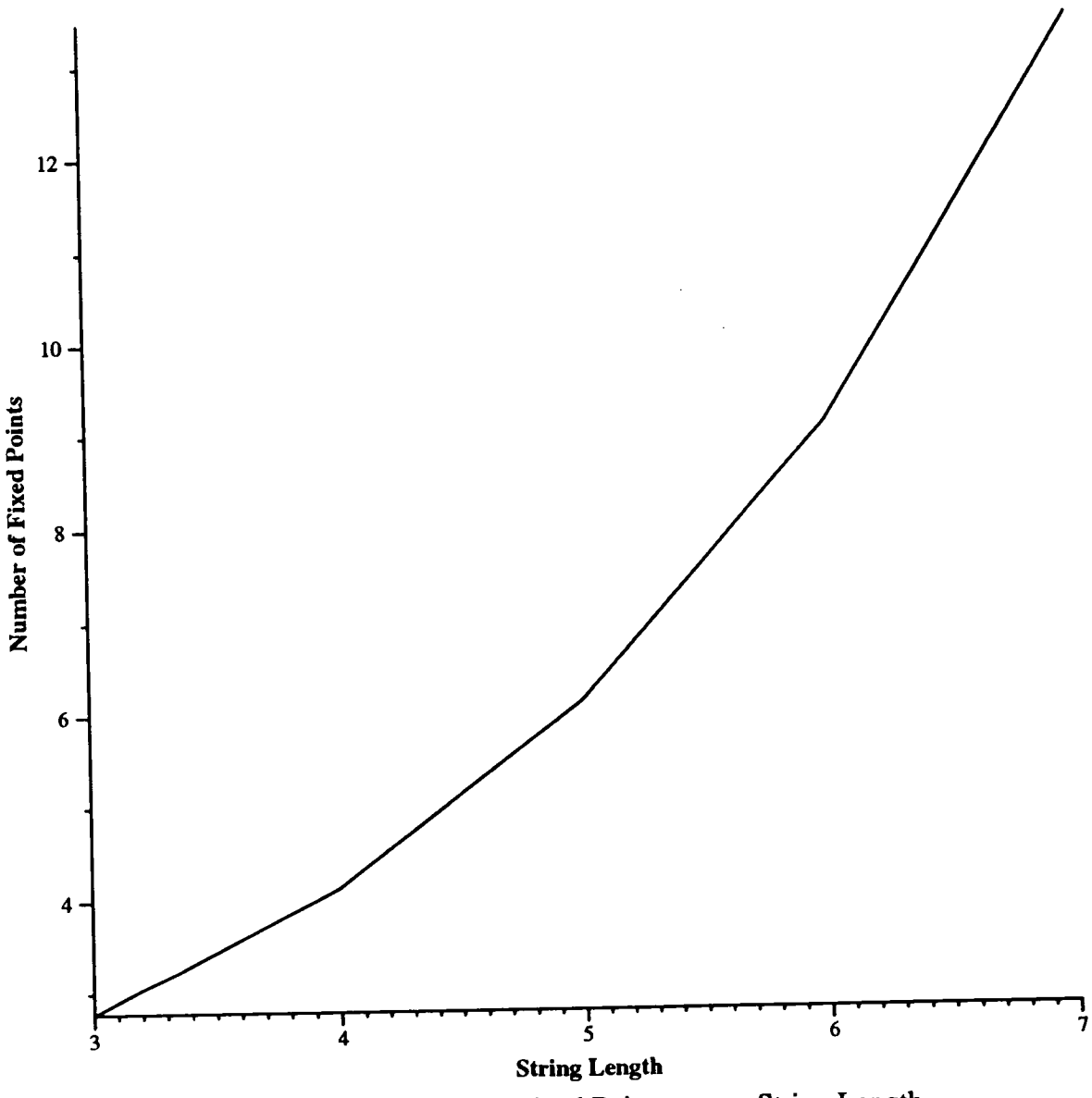


Figure 3.4: Predicted Maximum Number of Fixed Points

Chapter 4

Deviation from the Predicted Path

The second trend investigated was that of deviation of the ga from the expected path. That is, finding out how close the ga followed the inf path produced from the same initial population. Two different measures were taken for this. For both types, the deviation was found by summing the absolute differences between the ga and the inf sample points and dividing this by the number of sample points. A lower number is considered *good*.

4.1 Standard Deviation

The first measure taken is considered the *standard* deviation. This measures how far the *ga* is from its respective *inf* path. This measure ignores the fact that the ga might be following a different fixed point, and instead concentrates just on the fixed point found by inf from the same initial population. As shown by Figure 4.1, the

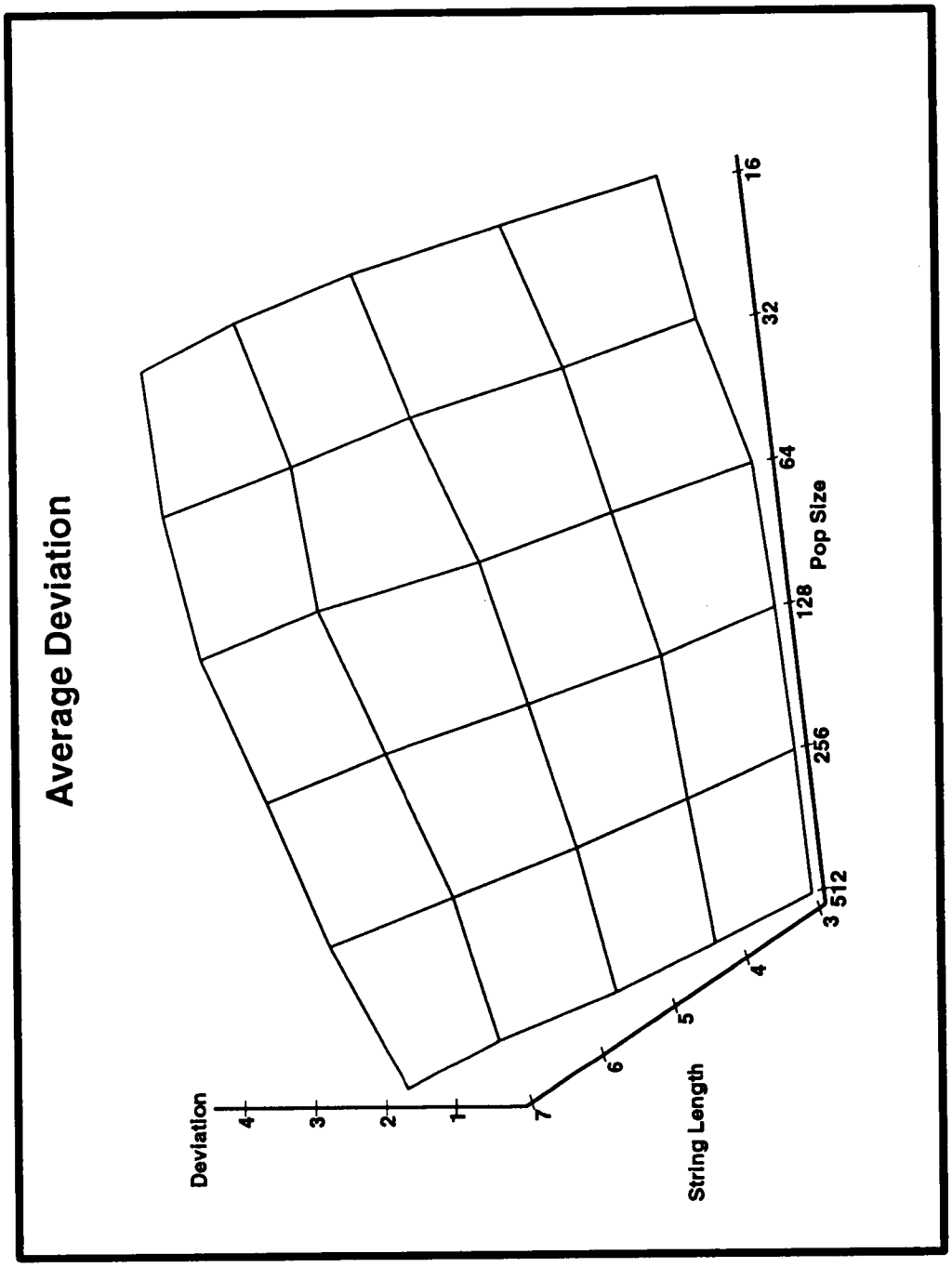


Figure 4.1: Average Deviation

average deviation follows a trend of being high at smaller population sizes and larger string lengths, and being low at larger population sizes and smaller string lengths. The overall curve of the graph shows no abnormalities. This follows from the fact that with larger string lengths and smaller population sizes we find more fixed points. The test cases individually exhibited an erratic behavior in the paths of the ga's with more fixed points found as shown by Figure 4.2.

With an increase in population size, the number of fixed points found drops and the graphs exhibit better behavior as shown by Figure 4.3.

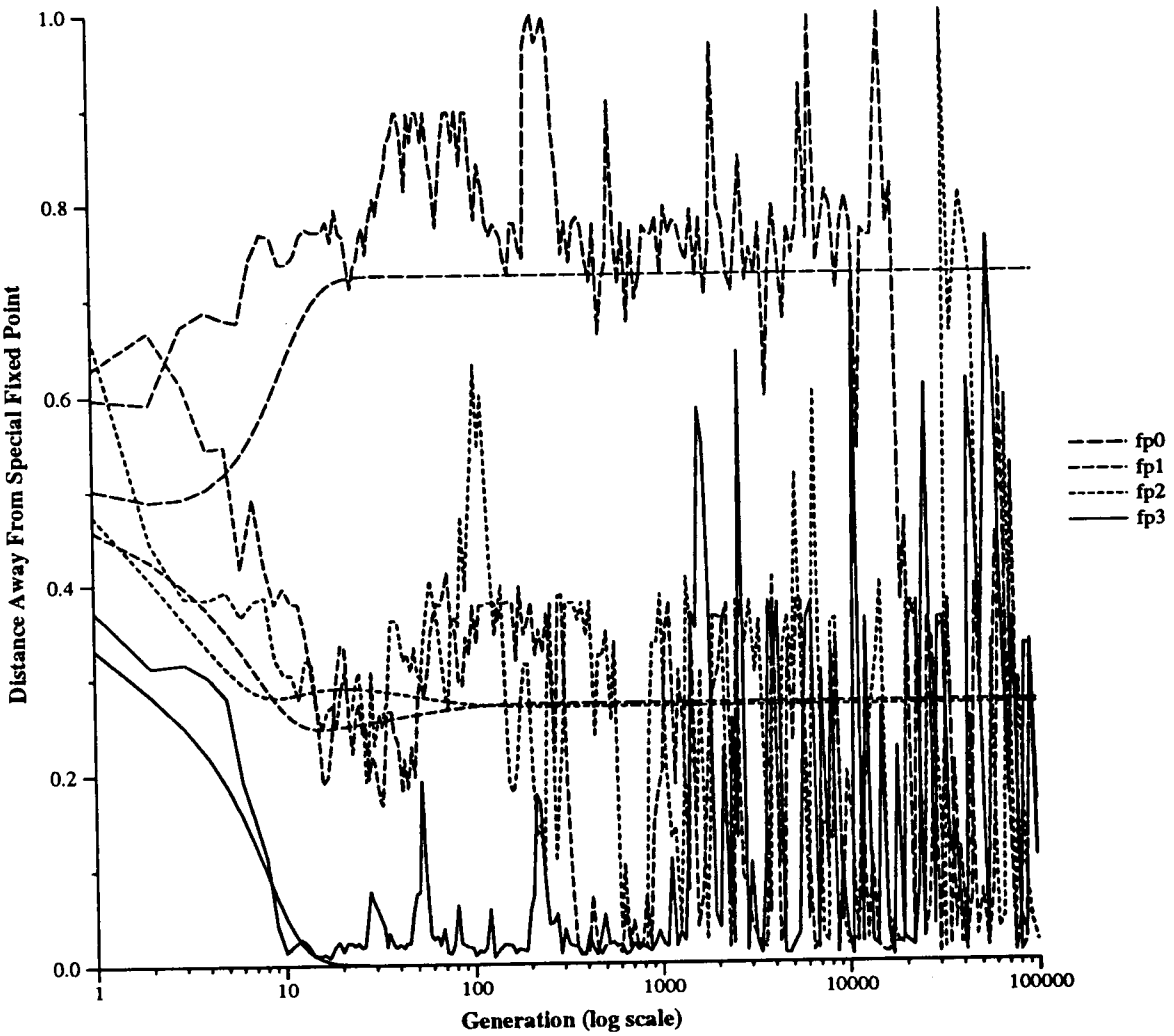
4.2 Adjusted Deviation

The second measure is considered the *adjusted* deviation. This measures how far the *ga* is from the *nearest inf* path. It doesn't matter if the path it is closest to started from the same or different initial population; we only look at how well it followed any path. This produces a much different set of results from the average deviation as shown in Figure 4.4. Overall, the results of this parallel the average deviation graphs. The behavior is better as indicated by the amount of deviation being one quarter that of the average deviation.

From Figure 4.4 we can see good behavior in the graph of the adjusted average deviation. This leads us to look at the individual lines making up the graph. As shown in Figure 4.5, the behavior of the deviation with the string length fixed at 3 and varying over the population sizes provides us with a fairly straight line when plotted using a ln ln graph.

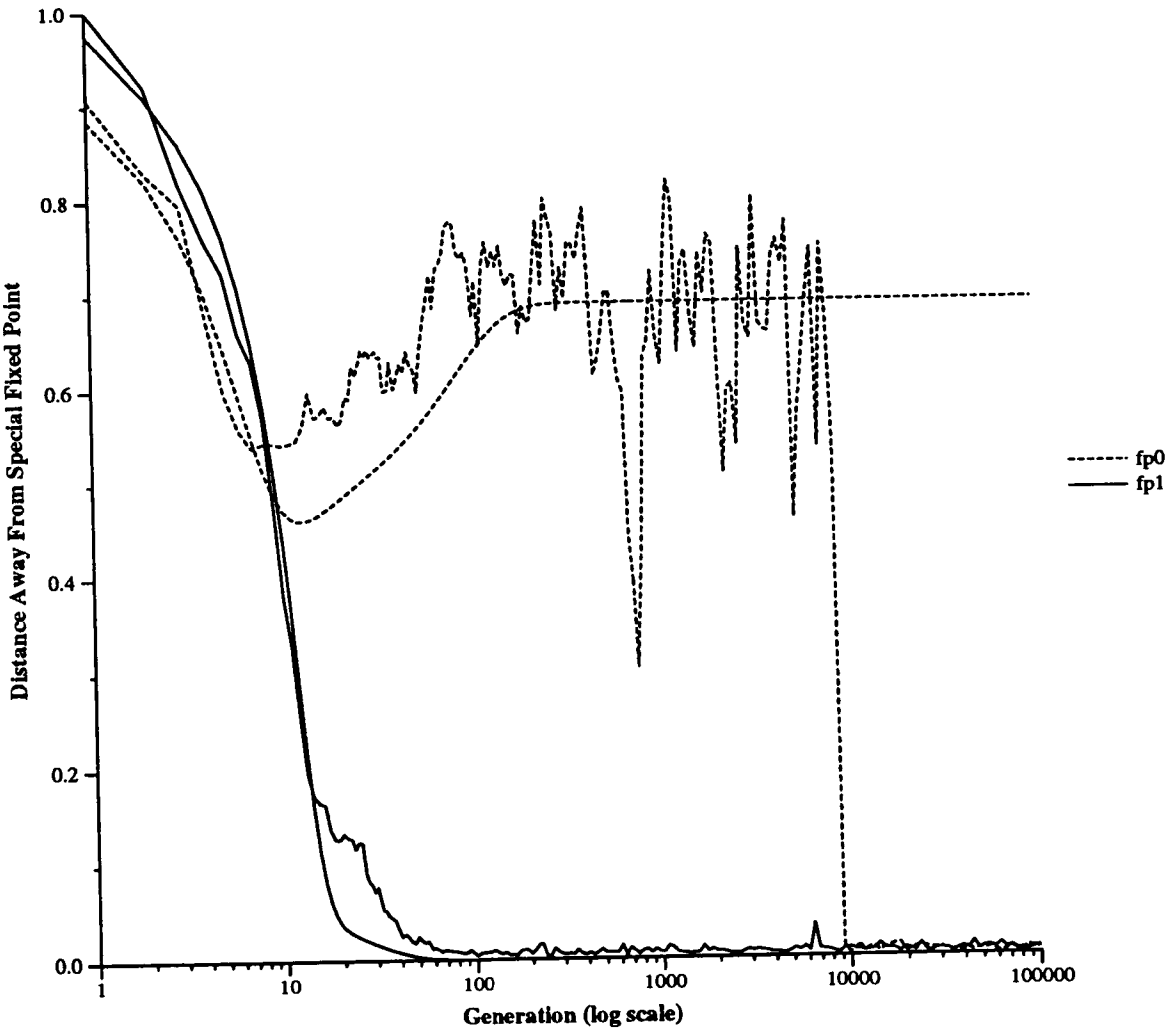
This leads us to try an equation of the form

$$d = kr^{\alpha}$$



Popsize 16 Bits 6 Fit 4 Seed 7

Figure 4.2: Erratic Behavior



Popsize 512 Bits 6 Fit 4 Seed 7

Figure 4.3: Good Behavior

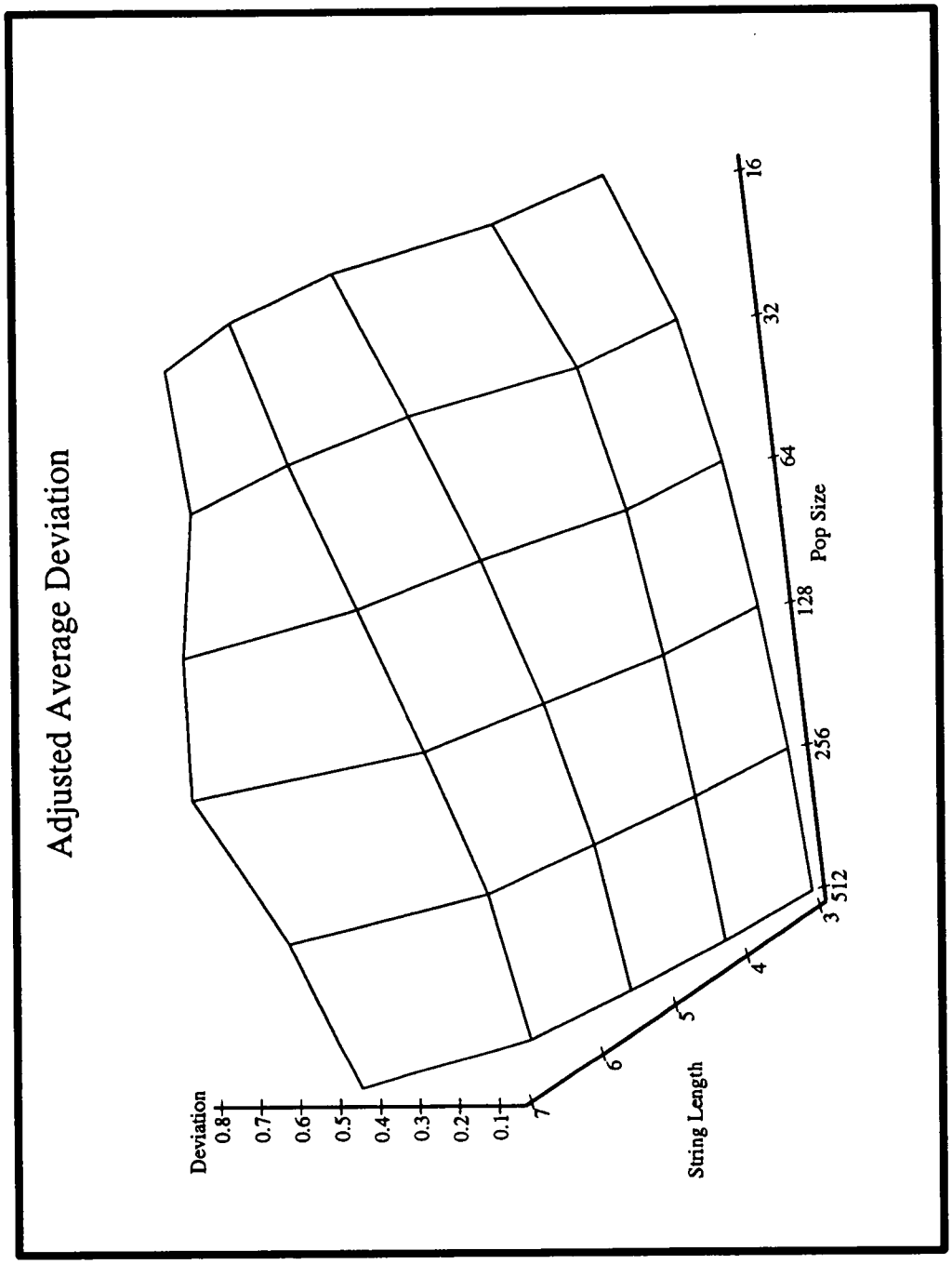


Figure 4.4: Adjusted Average Deviation

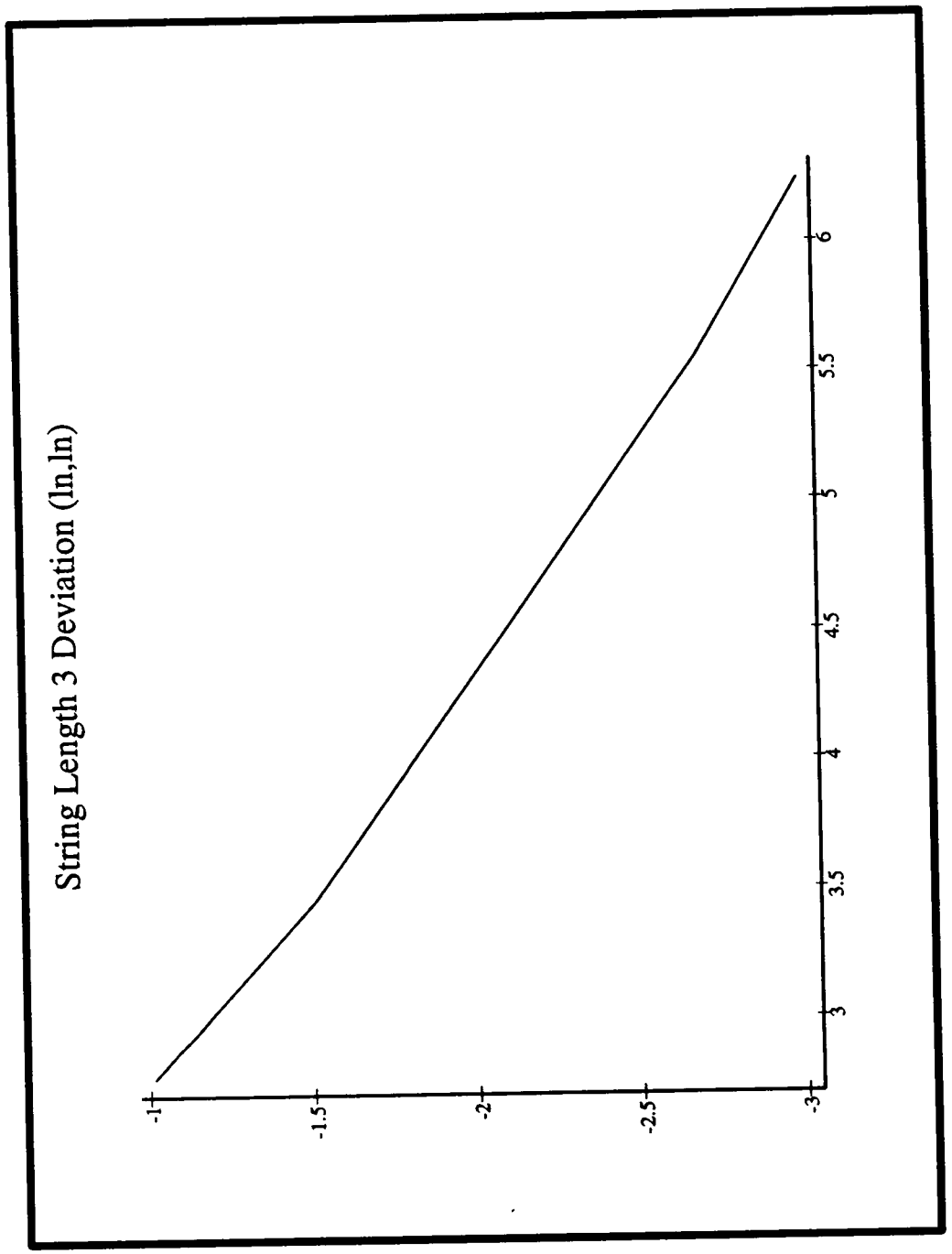


Figure 4.5: String Length 3 Behavior

where d is the deviation, k is the intercept, and α is the slope.

Likewise, with the Figure 4.6, a similar behavior is observed and the same equation can be tried substituting string length for population size.

$$d = kl^{\alpha}$$

However, because of abnormalities within the data, the equations cannot be solved for respectively higher string lengths and smaller population sizes. The reason for this is the higher deviations in these regions caused by failing to find all of the fixed points. In this instance, the deviation from string length 7 at population size 16 and string length 8 at population size 32 does not correspond with the norm and as such was not used (as shown in Figure 4.1).

As shown by Figures 4.7 and 4.8 the original test case produced a graph from which it is suspected that two additional fixed points were missed by *inf*. The after graph shows how the same test case with additional populations added to the search space performed. The additional populations were generated by taking a sample population when the *ga* is at a generation corresponding to the *missing* fixed points. With this additional information, four fixed points were found and the adjusted deviation performance goes from a value of 3.810993 to a value of 0.239563. For both graphs, the performance of *inf* is represented by the smooth line and of *ga* by the other line for each different fixed point.

With this in mind, we proceed with an equation of the form

$$d = kr^{\alpha}l^{\beta}$$

where d is the deviation, k is the intercept, r is the population size, and l is the string length. This leads us to

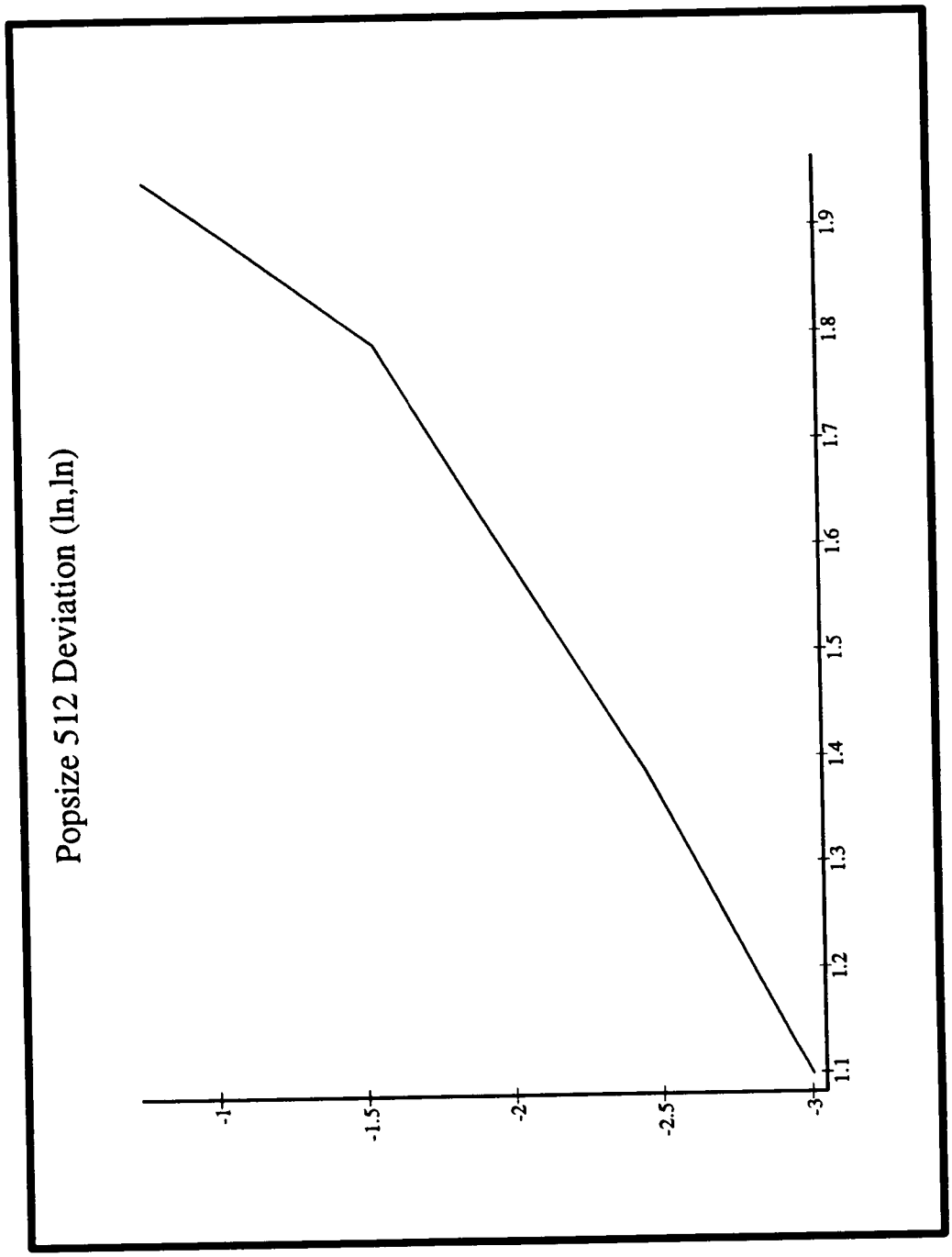
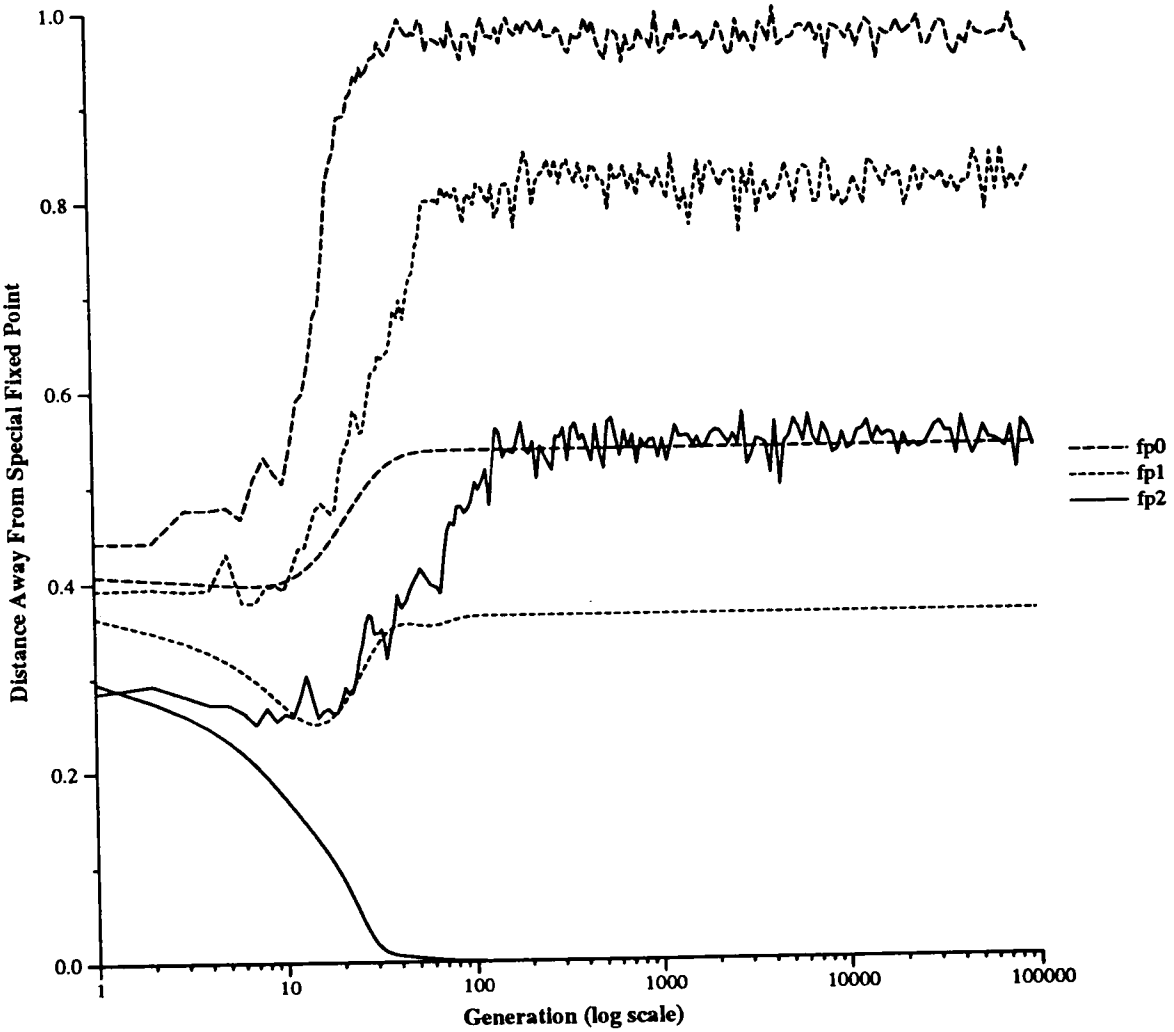
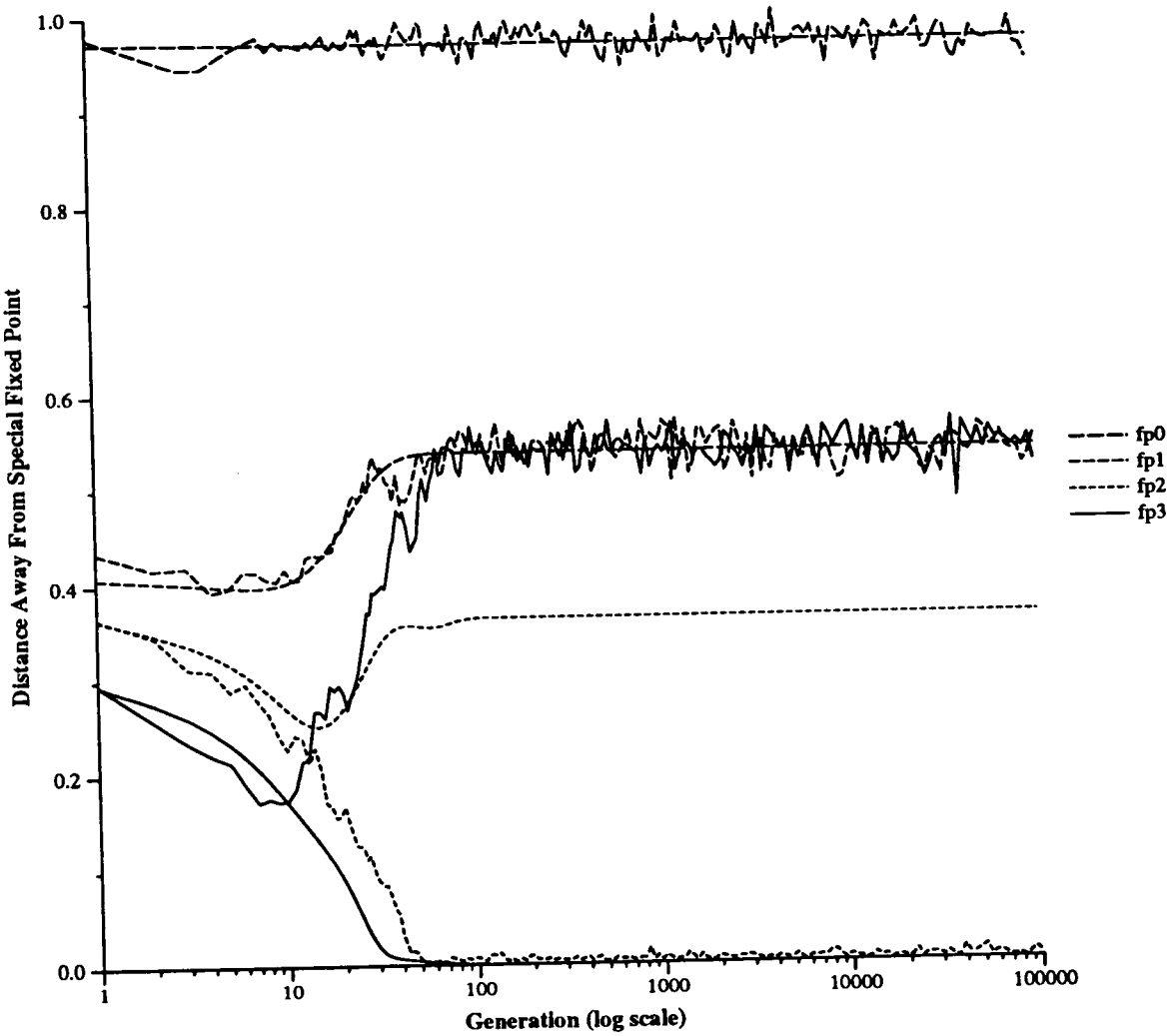


Figure 4.6: Population Size 512 Behavior



Popsize 256 Bits 7 Fit 0 Seed 4

Figure 4.7: Before Extra Populations



Popsize 256 Bits 7 Fit0 Seed4

Figure 4.8: After Extra Populations

$$\ln d = \ln k + \alpha \ln r + \beta \ln l$$

Let

$$\gamma = \ln k$$

We can now think of the data as triples of the form

$$\langle \ln r, \ln l, \ln d \rangle$$

These correspond to our 28 data points. If we call a data point

$$p = (p_x, p_y, p_z)$$

we want to minimize the error E over all p such that

$$E = \sum_p (p_z - (\gamma + \alpha p_x + \beta p_y))^2$$

Taking the partials, we get

$$\begin{aligned}
\frac{\partial}{\partial \gamma} &= -2 \sum_p (p_z - (\gamma + \alpha p_x + \beta p_y))(-1) = 0 \\
\frac{\partial}{\partial \alpha} &= -2 \sum_p (p_z - (\gamma + \alpha p_x + \beta p_y))(-p_x) = 0 \\
\frac{\partial}{\partial \beta} &= -2 \sum_p (p_z - (\gamma + \alpha p_x + \beta p_y))(-p_y) = 0
\end{aligned}$$

This leads us to

$$\begin{aligned}
\sum_p p_z &= \gamma \sum_p 1 + \alpha \sum_p p_x + \beta \sum_p p_y \\
\sum_p p_z p_x &= \gamma \sum_p p_x + \alpha \sum_p {p_x}^2 + \beta \sum_p p_y p_x \\
\sum_p p_z p_y &= \gamma \sum_p p_y + \alpha \sum_p p_x p_y + \beta \sum_p {p_y}^2
\end{aligned}$$

which gives

$$\begin{pmatrix} \sum_p 1 & \sum_p p_x & \sum_p p_y \\ \sum_p p_x & \sum_p {p_x}^2 & \sum_p p_y p_x \\ \sum_p p_y & \sum_p p_x p_y & \sum_p {p_y}^2 \end{pmatrix} \begin{pmatrix} \gamma \\ \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \sum_p p_z \\ \sum_p p_z p_x \\ \sum_p p_z p_y \end{pmatrix}$$

Let

$$A = \begin{pmatrix} \sum_p 1 & \sum_p p_x & \sum_p p_y \\ \sum_p p_x & \sum_p {p_x}^2 & \sum_p p_y p_x \\ \sum_p p_y & \sum_p p_x p_y & \sum_p {p_y}^2 \end{pmatrix}$$

and

$$B = \begin{pmatrix} \sum_p p_z \\ \sum_p p_z p_x \\ \sum_p p_z p_y \end{pmatrix}$$

then

$$\begin{pmatrix} \gamma \\ \alpha \\ \beta \end{pmatrix} = A^{-1} B$$

Solving for γ, α, and β gives

$$\begin{pmatrix} \gamma \\ \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} -2.51325607 \\ -0.4389659462 \\ 2.10824506 \end{pmatrix}$$

Solving for k gives

$$k = e^{\gamma} = 0.08100405453$$

From this we now have a formula to predict the adjusted average deviation for any string length and population size. Figure 4.9 shows the results of this formula

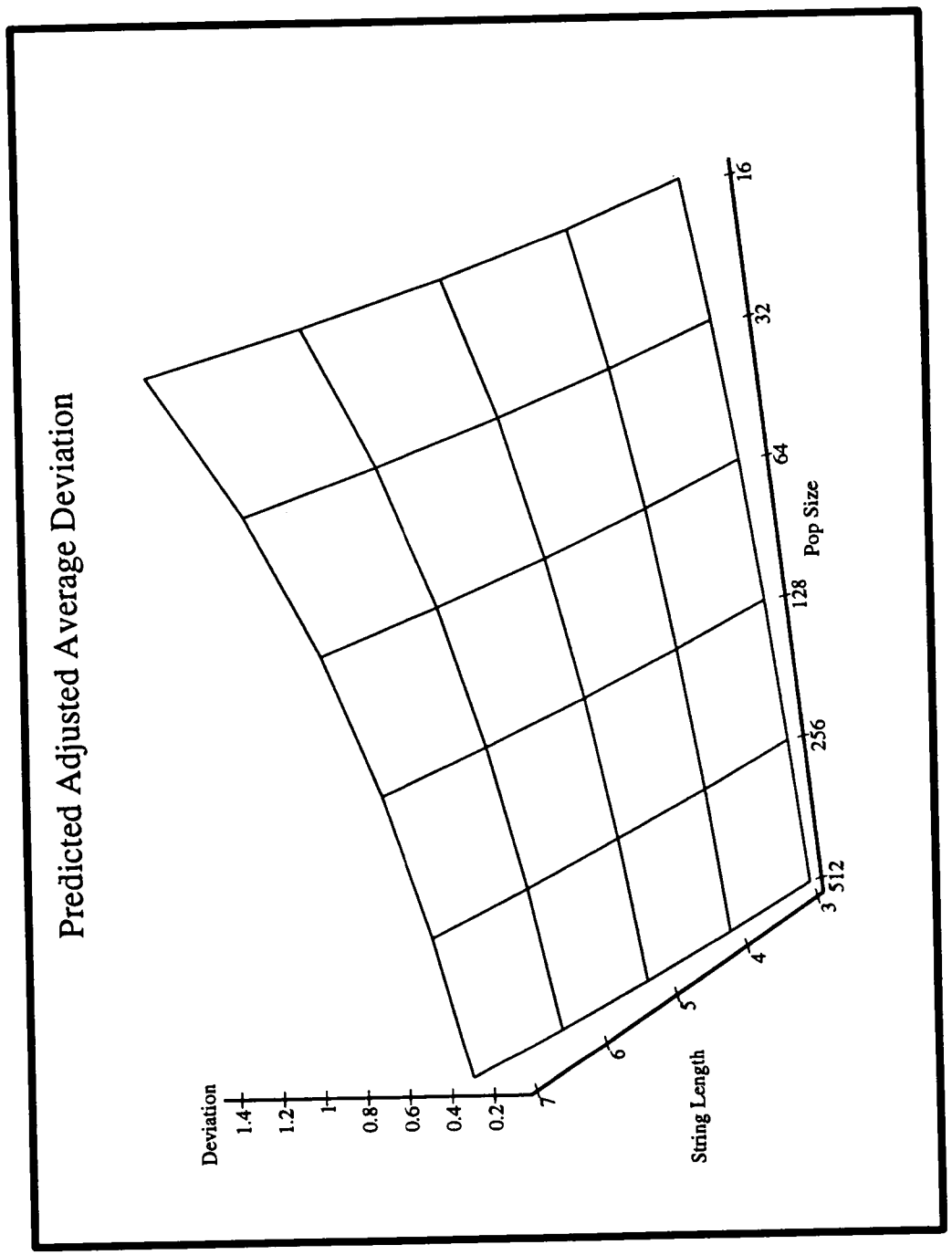


Figure 4.9: Predicted Adjusted Deviation

for string lengths 3 through 7 and population sizes 16 to 512 (population size doubles each time).

Chapter 5

Proportion Away from the Predicted Path

The third question we ask is how far away are the *ga's* from the fixed points? The way in which we answered this was to look at the data for each individual string length and population size combination and count the proportion of *ga's* that were farther away than a given deviation.

To do this, a normalized adjusted average deviation was produced for each *ga* found in a given test case. The normalized deviation was produced by dividing the distances recorded for all the fixed points by the largest distance so that all distances were relative on a scale of 0 to 1. By doing this, differences between test cases because of how far a *ga* was away from a particular fixed point could be compared on a reasonable basis.

For each individual test case, the number of *ga's* further away than a given deviation were counted and a percentage calculated. This was done for deviation values from 0.0 to 1.0 for each individual test case. From these numbers for each test case, a

simple average percentage was calculated for each combination of string lengths and population sizes.

Figures 5.1, 5.2, and 5.3 show some representative examples of the results. From this, we looked at a semi-log graph of the results as shown in Figures 5.4, 5.5, and 5.6.

Based on this information, an attempt was made to find a simple model of the behavior of the slope of the graphs. A three dimensional graph was made showing the slope values of the individual graphs graphed against string length and population size. This is shown in Figure 5.7. A simple trend was not readily apparent from this information. As shown in the graph, the line representing string length 3 over all populations shows a decrease as the population size increases. However, the same information over string length 7 shows an increase. Because of the lack of a pattern, no further attempts were made to represent this behavior with equations.

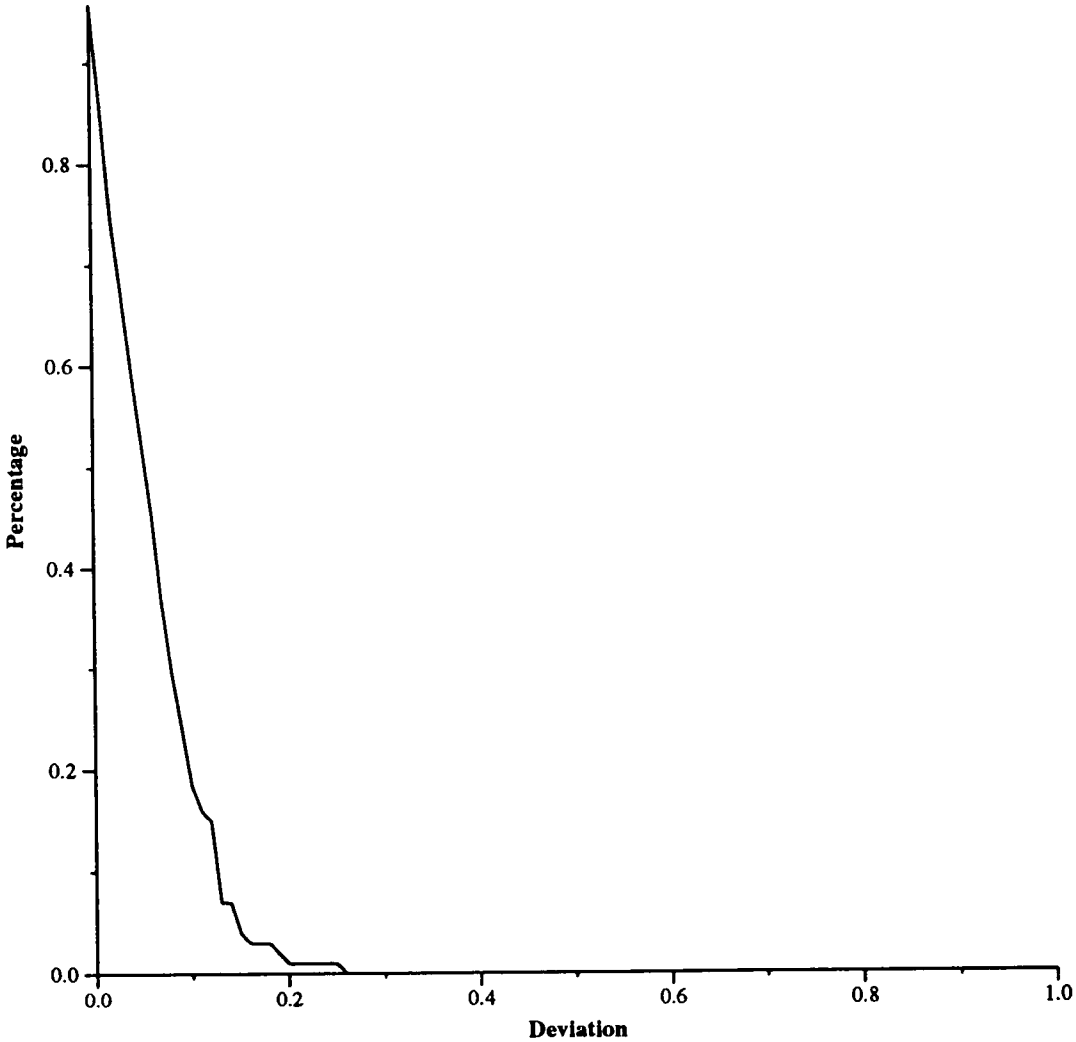


Figure 5.1: String Length 3 Population Size 16 Proportions

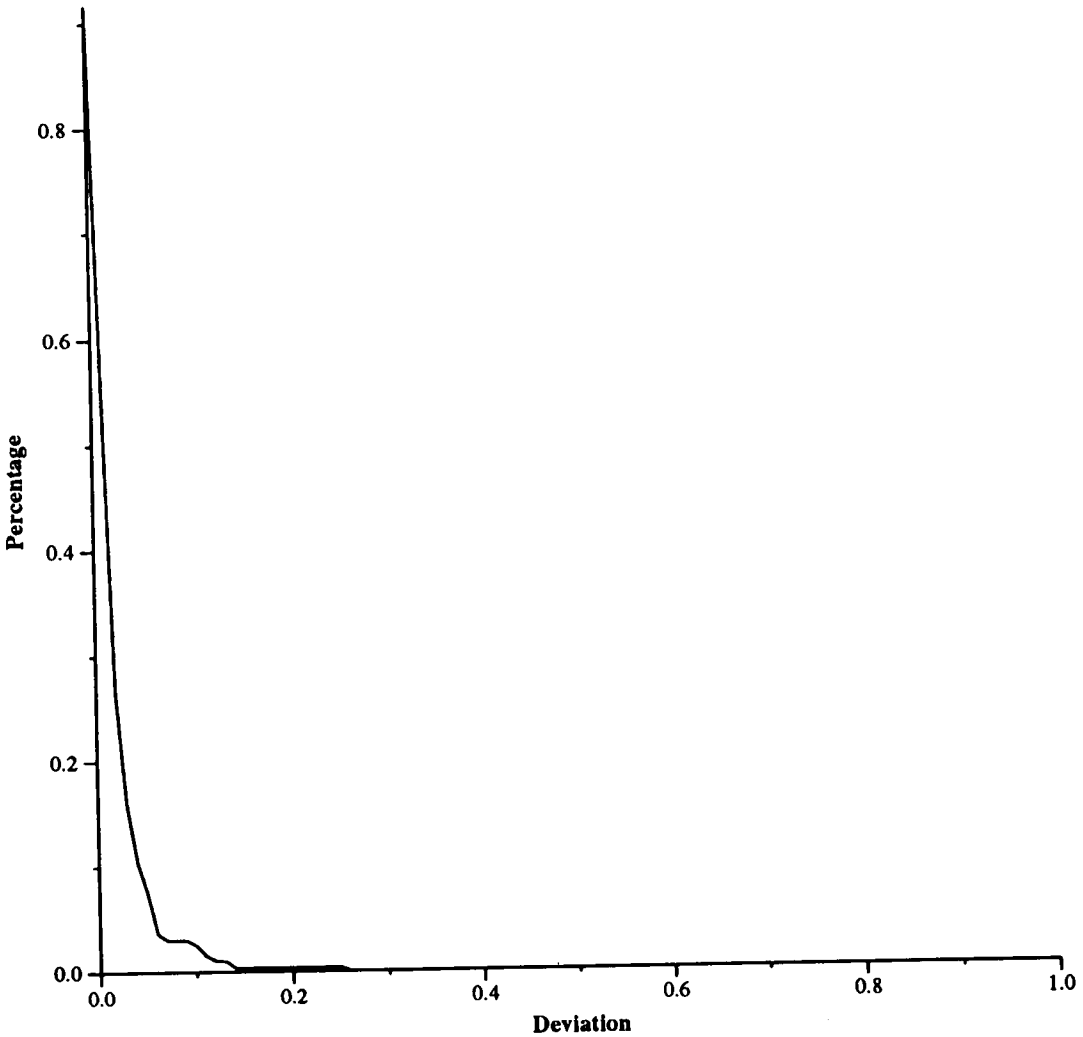


Figure 5.2: String Length 6 Population Size 256 Proportions

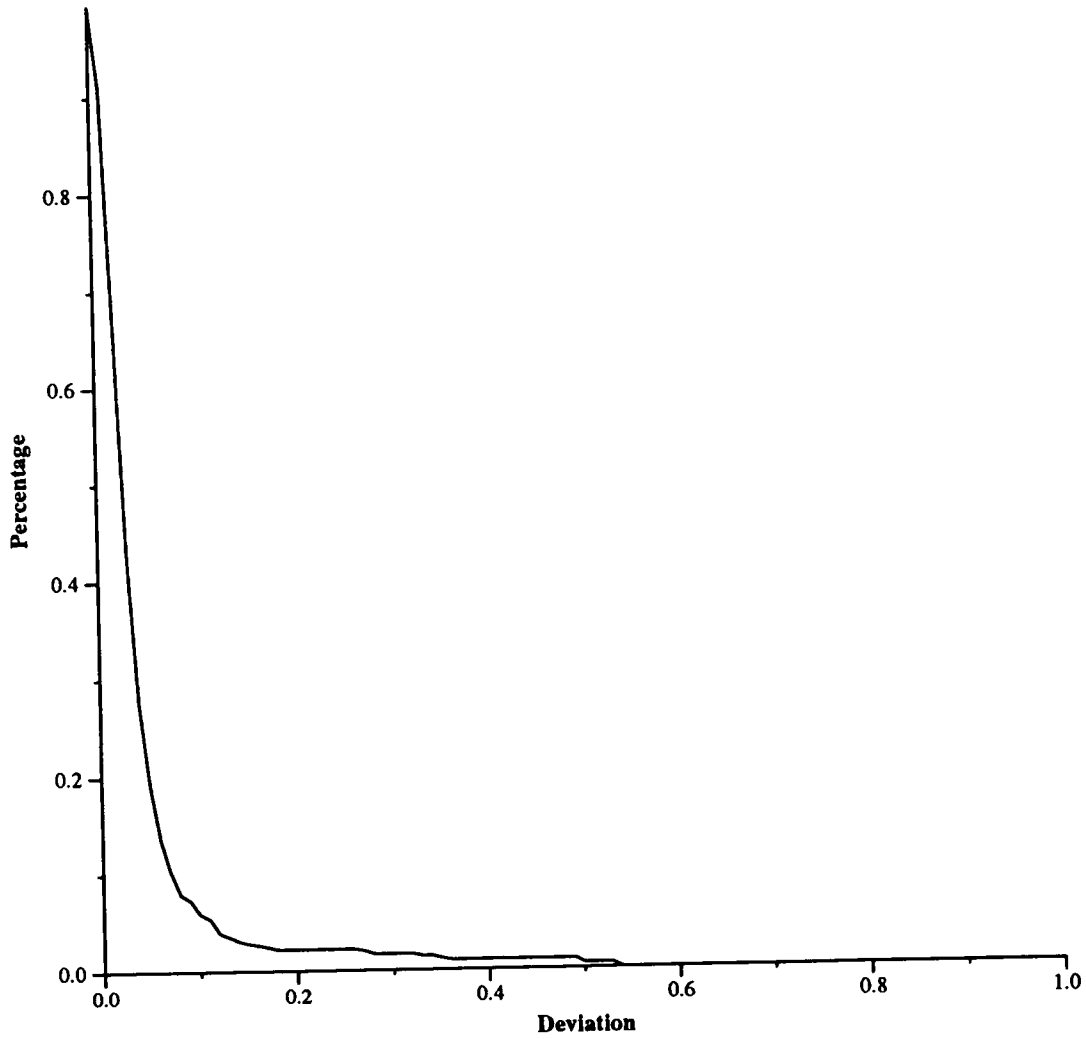


Figure 5.3: String Length 7 Population Size 64 Proportions

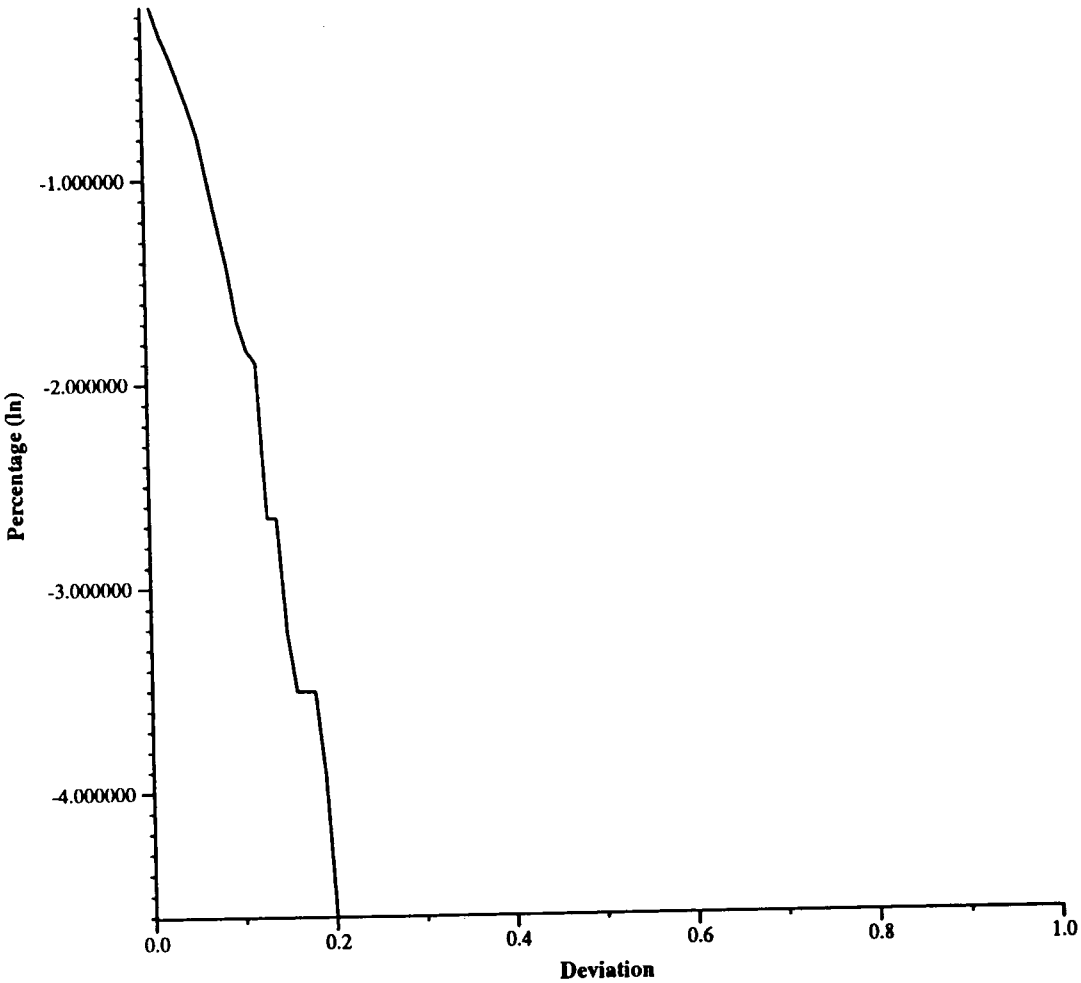


Figure 5.4: String Length 3 Population Size 16 Proportions (ln)

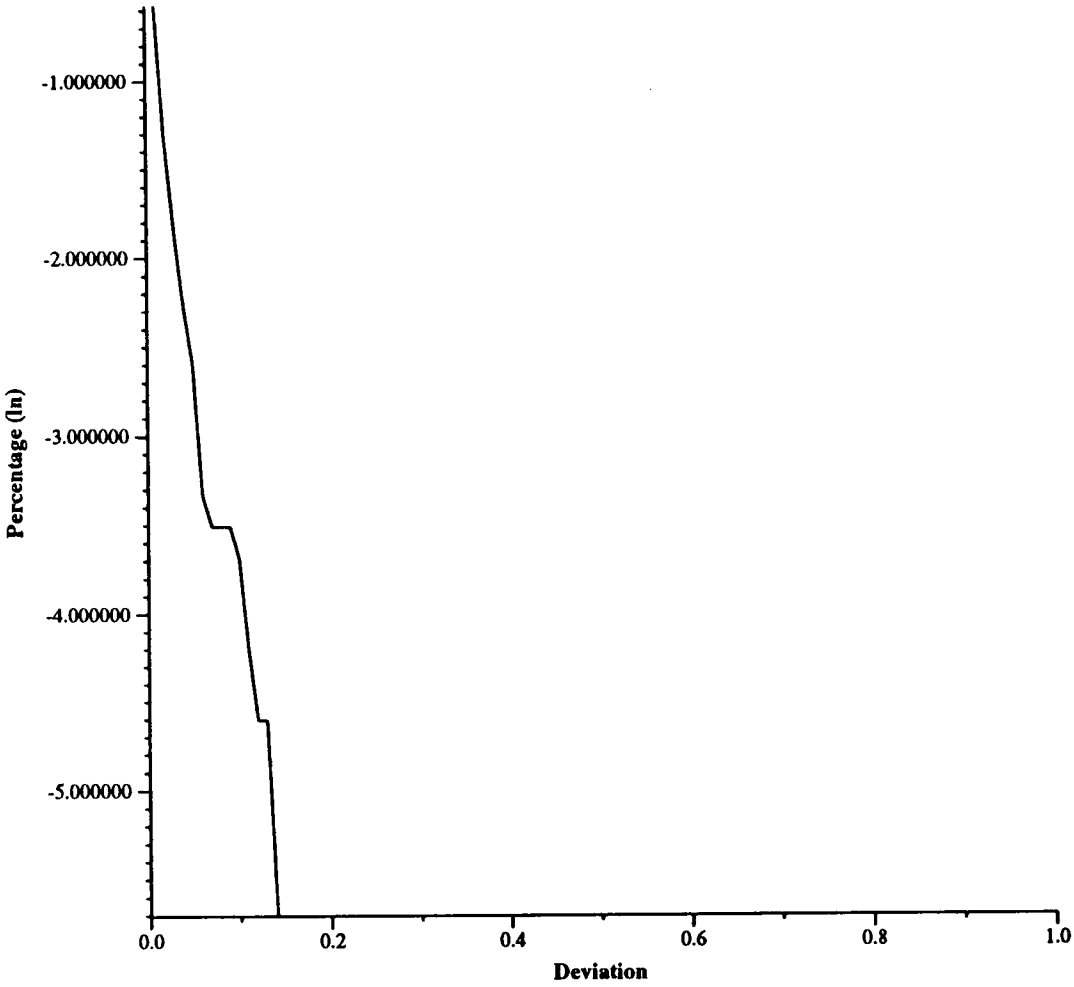
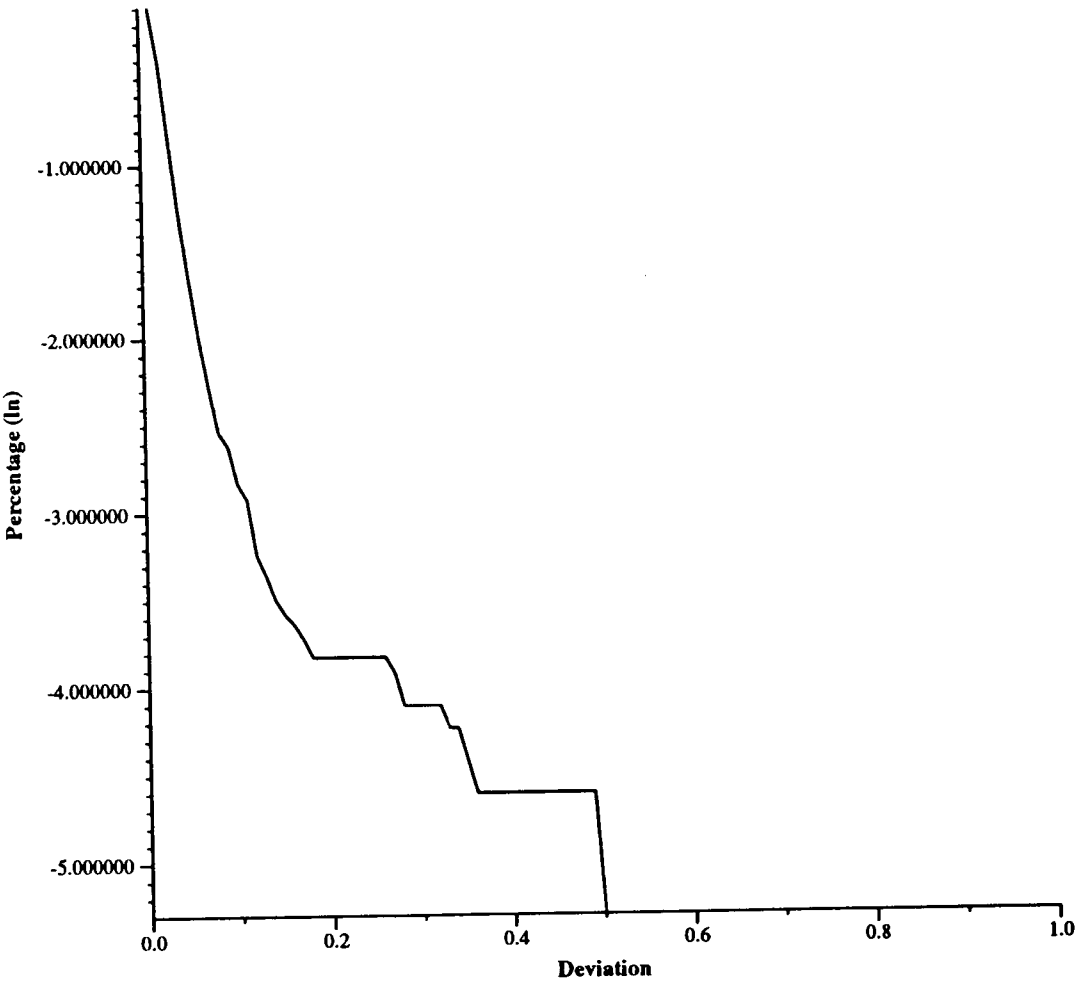


Figure 5.5: String Length 6 Population Size 256 Proportions (ln)



Proportion of GA's Farther Away Than Normalized Deviation
String Length 7 Population Size 64
(ln of percentage plotted)

Figure 5.6: String Length 7 Population Size 64 Proportions (ln)

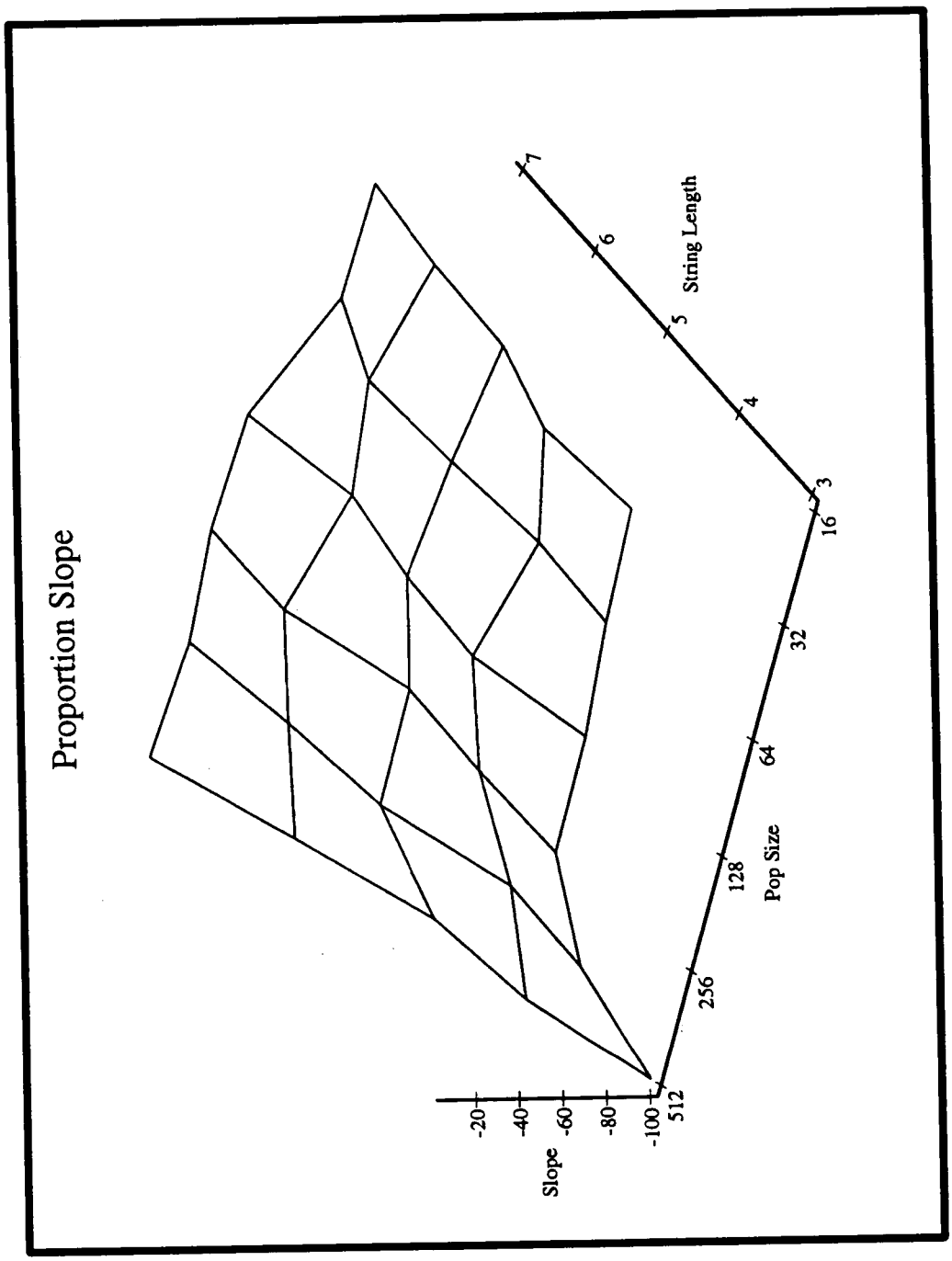


Figure 5.7: Proportion Slope Over String Length and Population Size

Chapter 6

Number of Shifts

The final way of analyzing the data looks at how many times the *ga's shift* from following one fixed point to another. Figure 6.1 shows the average number of shifts counted over string lengths from 3 to 7 and over population sizes from 16 to 512 (doubling population size each time).

The definition for a shift used in this study is that a shift of a *ga* has occurred when it moves from being closer to one fixed point to being closer to another fixed point. That is, we simply look at the data from an individual test case and find out which fixed point a particular *ga* is close to at generation n. If at generation $n+1$, the ga has moved closer to a different fixed point, then a shift has occurred.

With the data in place, we next want to derive an equation that will predict the number of shifts based on the string length and population size. From Figure 6.1, we can see good behavior exhibited over all lines in the graph. As such, all 30 points constituting the graph were used to derive an equation. Figure 6.2 shows how the data corresponding to string length 4 across all population sizes plots when using a ln ln scale.

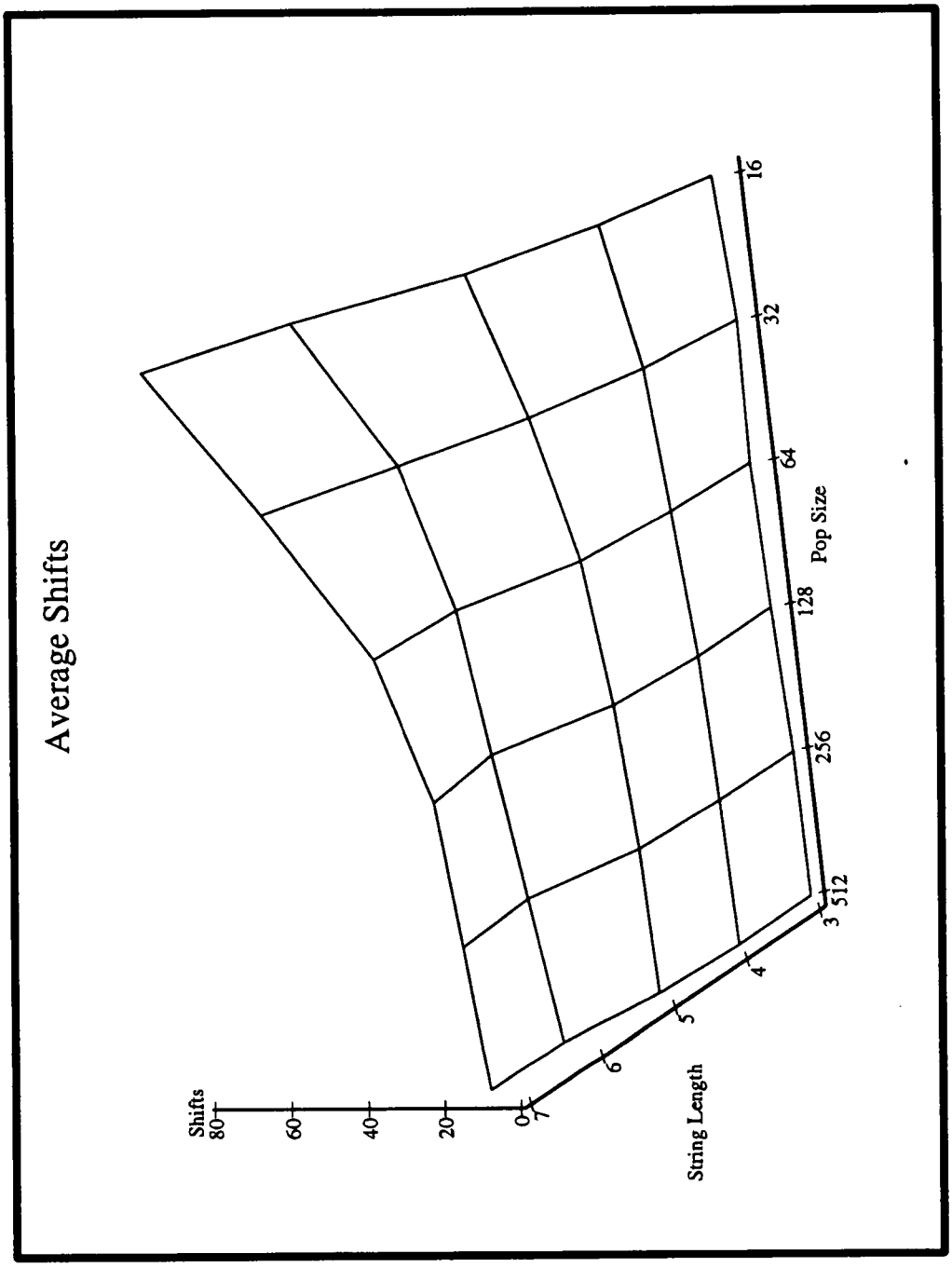


Figure 6.1: Average Number of Shifts

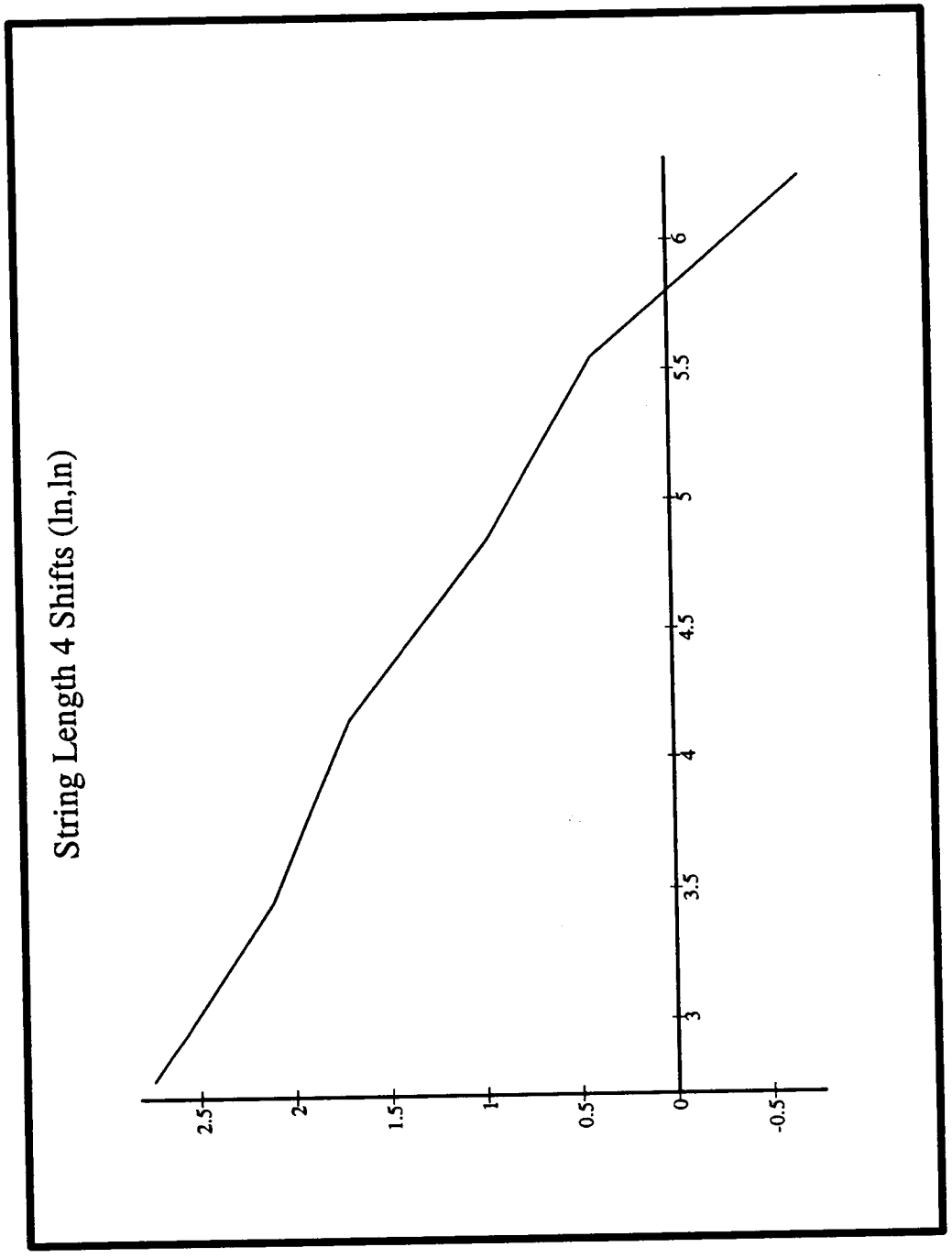


Figure 6.2: String Length 4 Behavior

So let s be the number of shifts, r be the population size, and l be the string length, then we have (because of the ln ln behavior of the graph)

$$s = kr^{\alpha}l^{\beta}$$

and

$$\ln s = \ln k + \alpha \ln r + \beta \ln l$$

Let $\gamma = \ln k$. We can now think of the data as triples of the form

$$\langle \ln r, \ln l, \ln s \rangle$$

These are our 30 data points. If we call a data point

$$p = (p_x, p_y, p_z)$$

we want to minimize the error E over all p such that

$$E = \sum_{p} (p_z - (\gamma + \alpha p_x + \beta p_y))^2$$

Taking the partials, we get

$$\begin{aligned}
\frac{\partial}{\partial \gamma} &= -2 \sum_{p} (p_z - (\gamma + \alpha p_x + \beta p_y))(-1) = 0 \\
\frac{\partial}{\partial \alpha} &= -2 \sum_{p} (p_z - (\gamma + \alpha p_x + \beta p_y))(-p_x) = 0 \\
\frac{\partial}{\partial \beta} &= -2 \sum_{p} (p_z - (\gamma + \alpha p_x + \beta p_y))(-p_y) = 0
\end{aligned}$$

This leads us to

$$\begin{aligned}
\sum_p p_z &= \gamma \sum_p 1 + \alpha \sum_p p_x + \beta \sum_p p_y \\
\sum_p p_z p_x &= \gamma \sum_p p_x + \alpha \sum_p {p_x}^2 + \beta \sum_p p_y p_x \\
\sum_p p_z p_y &= \gamma \sum_p p_y + \alpha \sum_p p_x p_y + \beta \sum_p {p_y}^2
\end{aligned}$$

which gives

$$\begin{pmatrix} \sum_p 1 & \sum_p p_x & \sum_p p_y \\ \sum_p p_x & \sum_p {p_x}^2 & \sum_p p_y p_x \\ \sum_p p_y & \sum_p p_x p_y & \sum_p {p_y}^2 \end{pmatrix} \begin{pmatrix} \gamma \\ \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \sum_p p_z \\ \sum_p p_z p_x \\ \sum_p p_z p_y \end{pmatrix}$$

Let

$$A = \begin{pmatrix} \sum_p 1 & \sum_p p_x & \sum_p p_y \\ \sum_p p_x & \sum_p {p_x}^2 & \sum_p p_y p_x \\ \sum_p p_y & \sum_p p_x p_y & \sum_p {p_y}^2 \end{pmatrix}$$

and

$$B = \begin{pmatrix} \sum_p p_z \\ \sum_p p_z p_x \\ \sum_p p_z p_y \end{pmatrix}$$

then

$$\begin{pmatrix} \gamma \\ \alpha \\ \beta \end{pmatrix} = A^{-1} B$$

Solving for γ, α, and β gives

$$\begin{pmatrix} \gamma \\ \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0.19188781 \\ -0.6736859486 \\ 3.13910110 \end{pmatrix}$$

Solving for k gives

$$k = e^{\gamma} = 1.211534587$$

We now have an equation to predict the number of shifts based on population size and string length. Figure 6.3 shows the results of this equation.

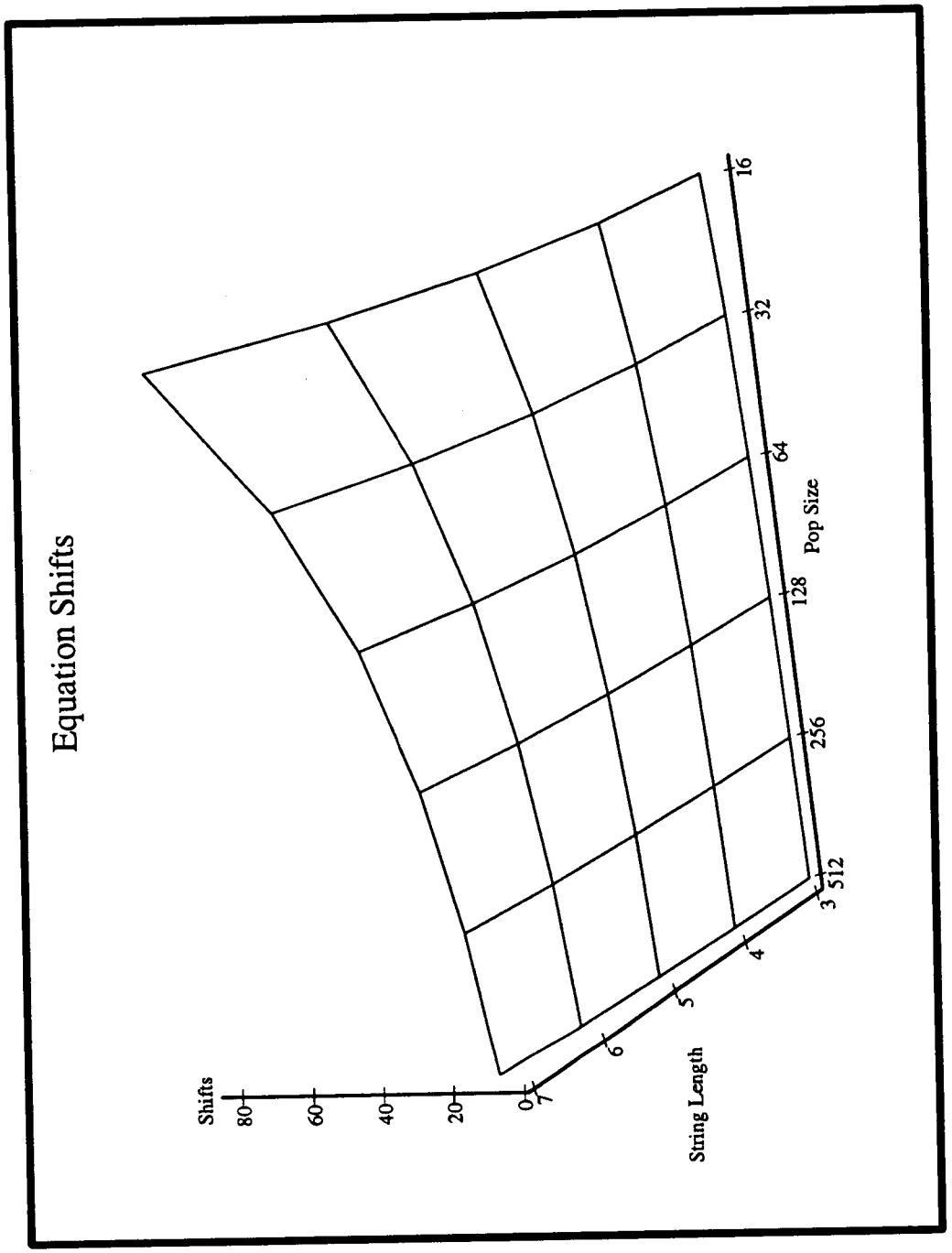


Figure 6.3: Predicted Number of Shifts

Chapter 7

Validation of Results

7.1 Shortcomings

In order to validate the results of this study, a few shortcomings in the methodology needed to be overcome. In particular, the fitness functions used overlapping values between different string lengths as did the random seed values for the random number generator. The fitness functions used had overlapping values as the string length increased. For a string length of 3, 8 fitness values were used. For a string length of 4, 16 fitness values were used. The first 8 values for string length 4 were the same ones used for string length 3. This overlapping proceeded through string length 7.

To a degree, the random seed values used were the same for each string length. That difference was the number of processing nodes used for a particular test case. A different seed value was assigned to each different node. So, if one test case was run with 8 nodes, then 8 random seed values were used. Likewise, if 16 nodes were used then 16 random seed values were used.

To correct these shortcomings, a new set of fitness functions and random seed

values were generated for string length 8. These values were generated in a way to guarantee uniqueness from the previous values used.

With these new parameters in place, we now look at the data from string length 8.

7.2 Number of Fixed Points Found

Extending Figure 3.4 we arrive at Figure 7.1 which shows a predicted value of 19.98 fixed points to be found for string length 8. Now, looking at Figure 7.2, we can see the actual number of different fixed points found for each population size. From this, we arrive at an average of 33.3 fixed points for string length 8, about twice what our model predicted.

This can be expected given the fact that we did not find all of the fixed points. By starting the test cases from more initial populations a better figure could be calculated.

7.3 Adjusted Average Deviation

From the equations derived in Chapter 4, we can predict what the behavior of the adjusted average deviation will be for string length 8. From the data collected for string length 8, we can calculate the actual adjusted average deviation. These results are presented in Figure 7.3. As shown by the graph, the deviation predicted and observed follow very closely.

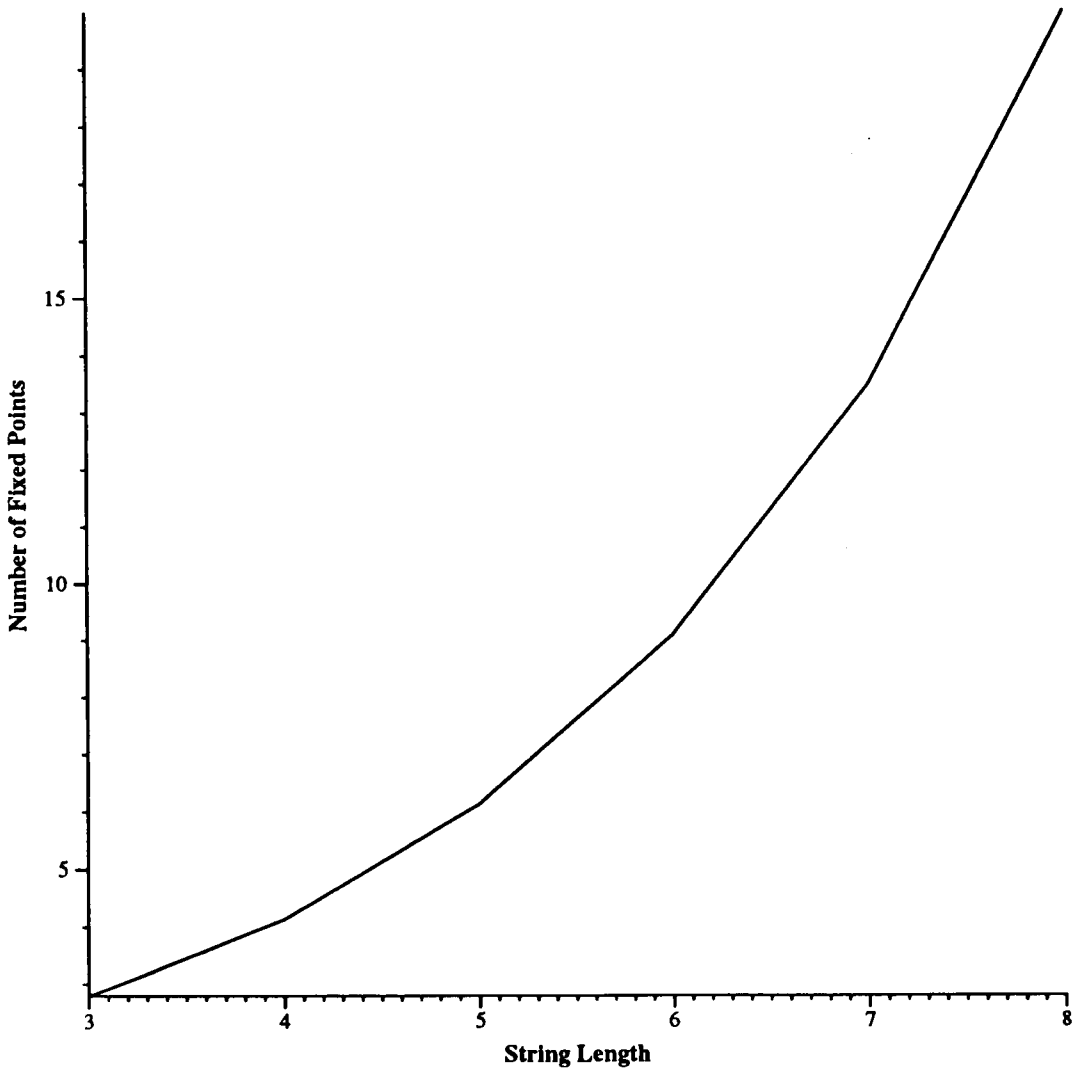


Figure 7.1: Extended Predicted Number of Fixed Points

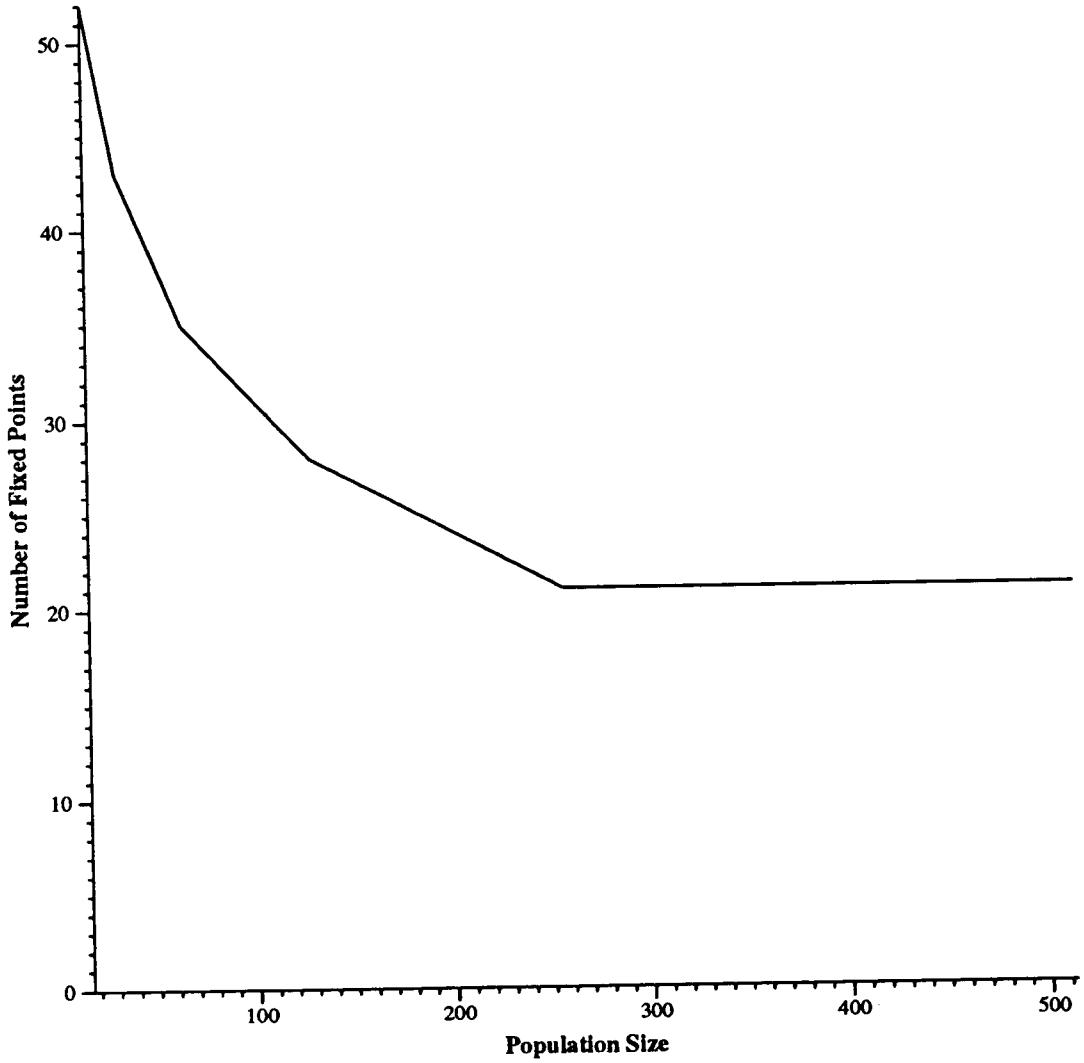


Figure 7.2: Maximum Fixed Points Over Population Size

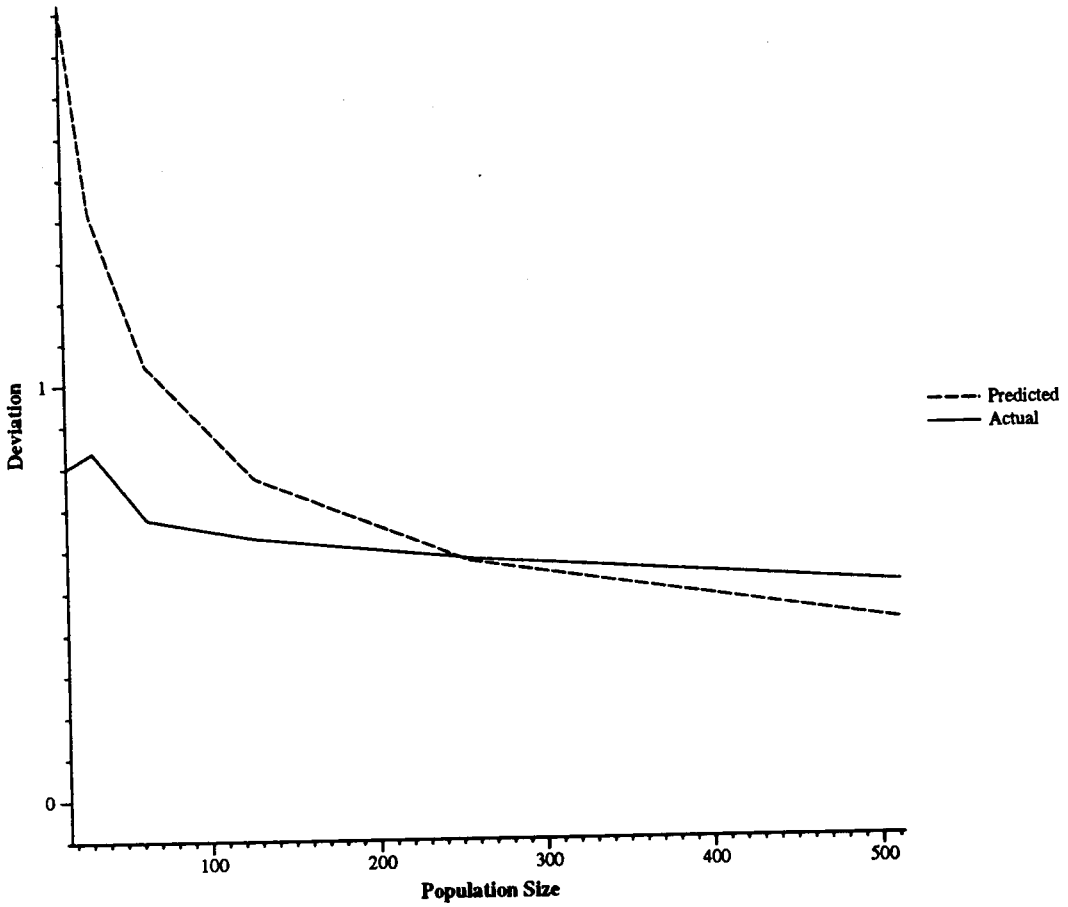


Figure 7.3: Comparison of Adjusted Average Deviations

7.4 Proportion Away from the Predicted Path

With the inconclusive results in Chapter 5, there was no opportunity to validate what was presented. However, the same information graphed in Figure 5.7 is presented here with the corresponding information from string length 8 also included. This information is presented in Figure 7.4. The graph presented was the best view that could be obtained to show the most information; however the reader may wish to consult Figure 5.7 to obtain a clearer picture of the information.

7.5 Number of Shifts

From the work done in Chapter 6, we can predict the behavior of the number of shifts observed for string length 8. As shown in Figure 7.5, the number of shifts predicted and the number of shifts observed are well within expectations.

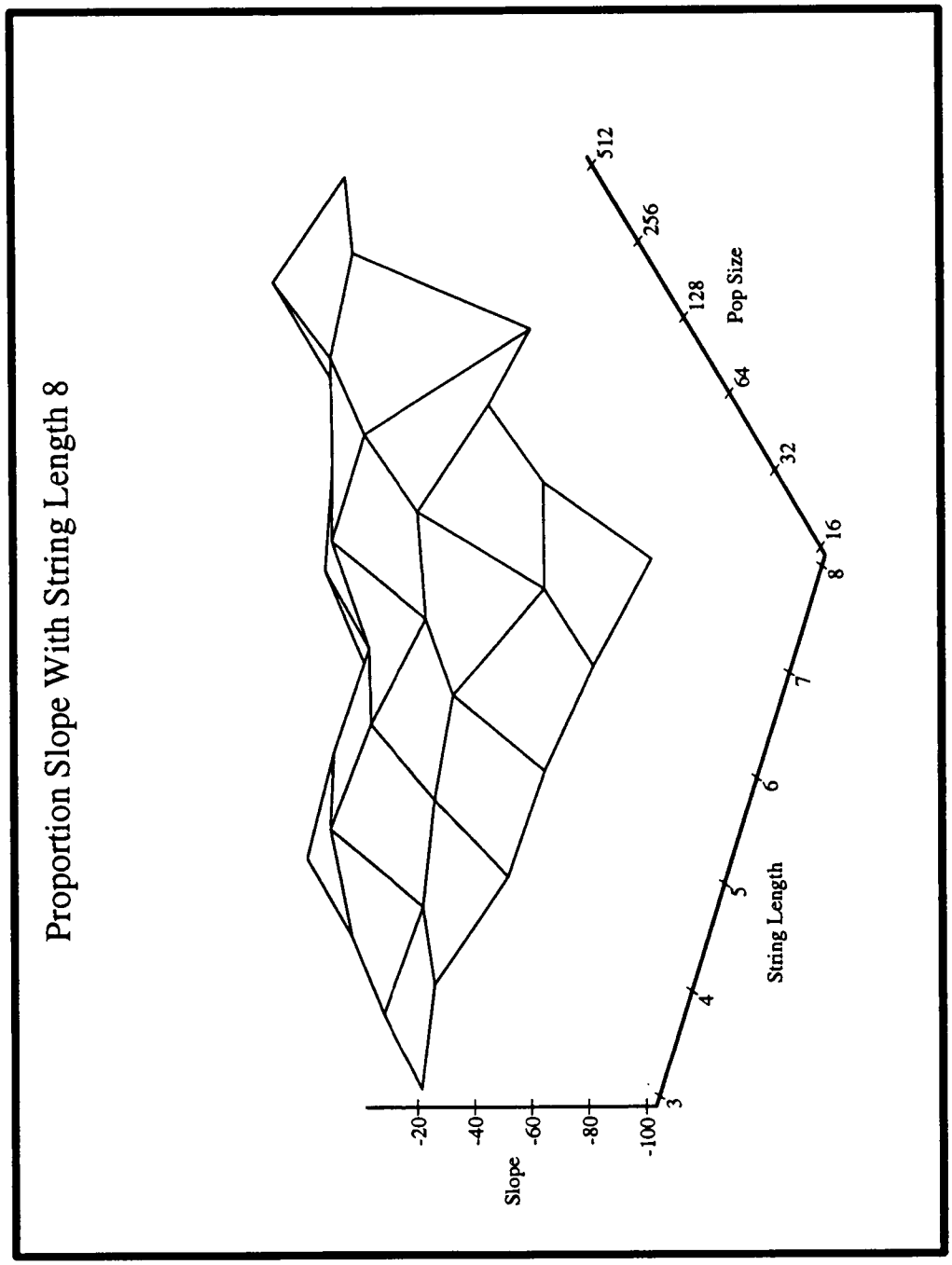


Figure 7.4: Revised Proportion Slope Over String Length and Population Size

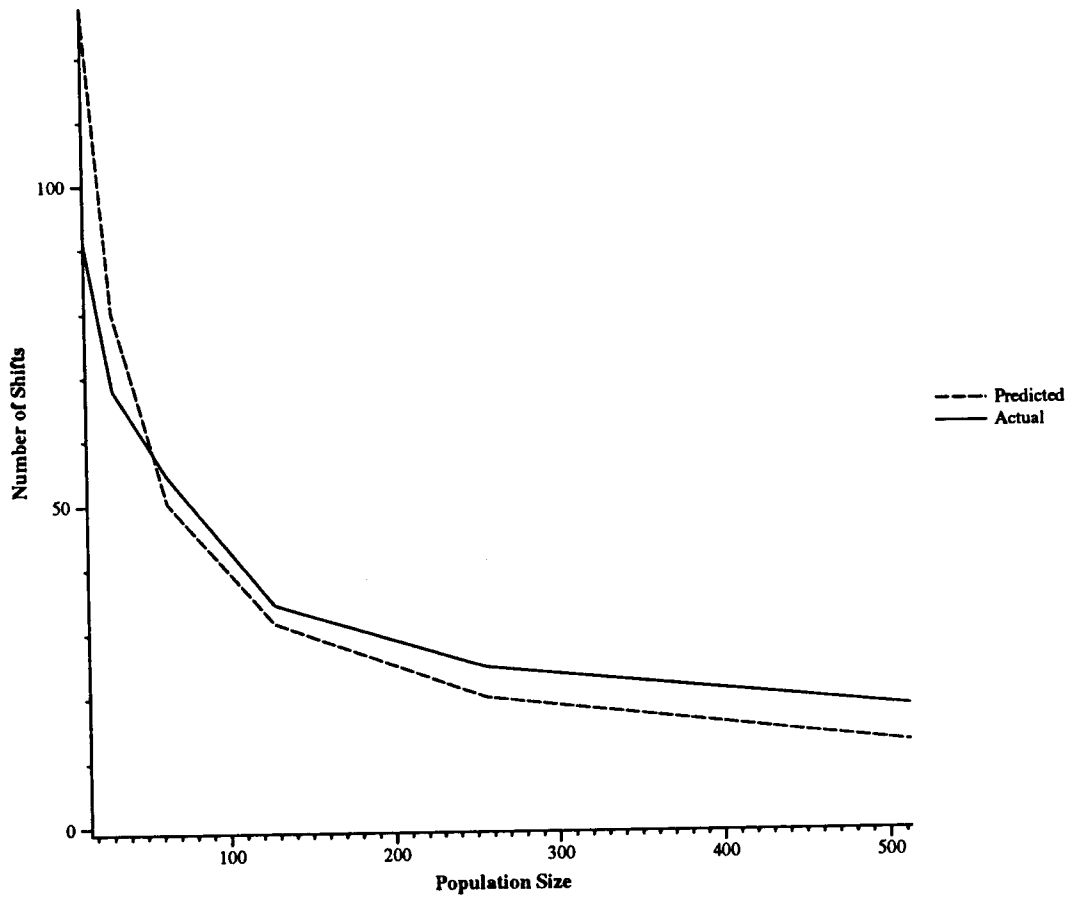


Figure 7.5: Comparison of Number of Shifts

Bibliography

Bibliography

[1] David L. Battle. *Implementing Genetic Algorithms*. Master's Thesis. The University of Tennessee, Knoxville, Tennessee. 1991.

[2] Michael D. Vose. Unpublished work.

[3] David E. Goldberg. *Genetic Algorithms in Search Optimization & Machine Learning*. Addison-Wesley, Reading, Massachusetts. 1989.

Appendices

Appendix A

Test Parameters

The fitness tables are shown with the different fitness functions labelled across columns. One column consists of one complete fitness function.

Table A.1: Fitness Functions for String Length 3-7

0	1	2	3	4	5	6	7	8	9
0.596768	0.250014	0.119808	0.926863	0.446639	0.773856	0.952663	0.401549	0.747484	0.468470
0.307223	0.810289	0.158017	0.074177	0.421327	0.063064	0.567444	0.613009	0.385138	0.418251
0.828477	0.636045	0.794732	0.195819	0.401995	0.397893	0.574484	0.431836	0.180459	0.562020
0.655839	0.947585	0.692670	0.888902	0.129460	0.322455	0.958127	0.849052	0.590678	0.911639
0.405479	0.824820	0.429069	0.150149	0.755946	0.579940	0.181842	0.066837	0.101123	0.087268
0.413300	0.852594	0.025122	0.350824	0.694484	0.695474	0.249108	0.274430	0.407220	0.805002
0.311095	0.670546	0.178711	0.429845	0.048934	0.551568	0.128898	0.579866	0.386475	0.984164
0.500497	0.885463	0.668865	0.175863	0.485183	0.247581	0.825204	0.587741	0.600314	0.341988
0.940863	0.648521	0.805004	0.245624	0.279672	0.046138	0.479822	0.211637	0.320222	0.292179
0.069869	0.679648	0.466711	0.583597	0.324847	0.286165	0.765247	0.690858	0.699293	0.320497
0.200601	0.288031	0.413411	0.359610	0.679511	0.204391	0.954619	0.348286	0.211497	0.575782
0.189413	0.975646	0.411403	0.237582	0.230051	0.134453	0.820839	0.922048	0.248909	0.184769
0.339938	0.189473	0.163769	0.007184	0.385232	0.059710	0.081829	0.234531	0.195757	0.071229
0.809724	0.092418	0.516116	0.264611	0.488055	0.392752	0.062428	0.309586	0.790414	0.235086
0.154318	0.955563	0.182713	0.162618	0.263149	0.476247	0.533564	0.583902	0.956573	0.678079
0.170382	0.400104	0.921196	0.327066	0.671381	0.440473	0.800206	0.401813	0.478078	0.470464
0.401248	0.568172	0.823161	0.946445	0.567741	0.887081	0.885960	0.311109	0.692862	0.742361
0.179300	0.583483	0.785058	0.869049	0.606988	0.049684	0.840670	0.022860	0.259831	0.982878
0.304951	0.906865	0.110365	0.660154	0.188012	0.464412	0.784043	0.628135	0.781184	0.673455
0.354890	0.530275	0.793364	0.142021	0.724347	0.281841	0.585493	0.501804	0.318178	0.491316
0.415459	0.340885	0.880847	0.202259	0.913957	0.207091	0.778221	0.120238	0.073660	0.198127
0.153548	0.366604	0.750214	0.176765	0.308692	0.150328	0.496615	0.258754	0.398852	0.368931
0.262473	0.459208	0.728240	0.951325	0.959413	0.384192	0.353639	0.601848	0.911378	0.472615
0.432216	0.473888	0.387806	0.881311	0.710973	0.706108	0.053672	0.595920	0.575864	0.351091
0.357269	0.091791	0.260143	0.740998	0.015852	0.530818	0.530870	0.383317	0.644002	0.919228
0.249991	0.691480	0.558594	0.402393	0.034447	0.577764	0.611881	0.916216	0.405056	0.134028
0.601081	0.965935	0.192486	0.666604	0.360731	0.240069	0.240019	0.508806	0.504182	0.744118
0.791321	0.899240	0.525354	0.712051	0.809675	0.637291	0.203270	0.862401	0.606299	0.888781
0.315409	0.285538	0.695472	0.197349	0.263552	0.945531	0.360816	0.948166	0.976597	0.481702
0.681015	0.039478	0.134399	0.927802	0.564424	0.258779	0.592768	0.223397	0.871974	0.102815
0.876649	0.895597	0.920117	0.877143	0.160674	0.634928	0.292614	0.794804	0.873199	0.373657
0.808424	0.067719	0.117727	0.539512	0.462151	0.628157	0.950586	0.064506	0.842449	0.972021
0.076239	0.854802	0.701318	0.599140	0.130409	0.052154	0.082370	0.209491	0.038335	0.329163
0.668527	0.707630	0.542355	0.407345	0.380779	0.643751	0.143912	0.796486	0.825744	0.899550
0.979452	0.365231	0.500328	0.024670	0.796691	0.820676	0.101940	0.458583	0.348323	0.603078
0.880250	0.022079	0.153661	0.882397	0.393490	0.987415	0.234256	0.695225	0.569821	0.792316
0.984756	0.604849	0.173539	0.570914	0.638008	0.171149	0.356819	0.512964	0.777909	0.944286
0.776029	0.317902	0.666636	0.334897	0.389260	0.422889	0.220838	0.974600	0.755412	0.209209
0.106117	0.997677	0.778782	0.597521	0.243439	0.293821	0.038985	0.115607	0.734722	0.575594
0.993722	0.456609	0.053593	0.990470	0.774817	0.040091	0.203848	0.563061	0.785164	0.706271
0.223660	0.339282	0.078396	0.210455	0.250254	0.811901	0.976443	0.718701	0.312443	0.908518
0.455167	0.620088	0.690267	0.003345	0.818236	0.485316	0.534152	0.814079	0.964520	0.747950
0.764808	0.492069	0.908526	0.572122	0.648622	0.107586	0.595178	0.357753	0.332085	0.266148
0.244689	0.331019	0.410102	0.990007	0.659570	0.188570	0.121259	0.008144	0.188239	0.682623
0.899418	0.697266	0.019631	0.153539	0.812844	0.841993	0.575803	0.217434	0.262064	0.036016
0.601234	0.867469	0.725413	0.266471	0.856458	0.401792	0.569291	0.419452	0.140580	0.990667
0.401677	0.778449	0.499297	0.097505	0.764807	0.046608	0.507613	0.808817	0.892070	0.080056
0.413676	0.535997	0.535809	0.788936	0.154125	0.079165	0.234544	0.779180	0.200182	0.531624
0.993697	0.366737	0.284331	0.777203	0.250657	0.793199	0.837268	0.875837	0.480732	0.180064
0.983314	0.914756	0.062741	0.700664	0.846763	0.954491	0.347171	0.424705	0.955323	0.006617
0.240694	0.847591	0.296675	0.110303	0.970085	0.231500	0.979594	0.639620	0.902349	0.810521
0.344402	0.174002	0.862340	0.562968	0.116480	0.266071	0.373313	0.887705	0.768900	0.189296
0.767160	0.641446	0.538982	0.187785	0.442534	0.876011	0.882232	0.045701	0.562327	0.186799
0.557849	0.294471	0.826477	0.684281	0.107324	0.429049	0.194427	0.948712	0.619646	0.211405
0.159229	0.662499	0.493894	0.915802	0.918777	0.574816	0.897164	0.697009	0.970384	0.824770
0.788344	0.182295	0.002082	0.387350	0.984487	0.145700	0.002077	0.337043	0.905035	0.496450
0.230984	0.955487	0.456698	0.475038	0.290918	0.010910	0.485073	0.403518	0.346804	0.089089
0.159949	0.928415	0.252377	0.788474	0.021216	0.754142	0.430572	0.635351	0.354715	0.662470
0.676388	0.582354	0.192342	0.864232	0.332769	0.501779	0.856187	0.390470	0.242354	0.308561
0.525228	0.802742	0.275407	0.267752	0.362456	0.592525	0.947586	0.371612	0.531302	0.294952
0.428544	0.247745	0.851583	0.779910	0.056476	0.524325	0.892289	0.761465	0.629311	0.860716
0.535066	0.352644	0.512074	0.094947	0.659674	0.128679	0.908060	0.605266	0.631063	0.774954
0.394381	0.887786	0.890082	0.578342	0.241744	0.953759	0.786219	0.472134	0.865592	0.766395
0.947141	0.191912	0.751411	0.255154	0.504855	0.006048	0.275974	0.648576	0.535058	0.585908

Table A.1: (continued)

0	1	2	3	4	5	6	7	8	9
0.846209	0.340367	0.388315	0.373142	0.074593	0.474264	0.788804	0.972157	0.386850	0.411979
0.745434	0.667943	0.723144	0.356265	0.861275	0.719075	0.420467	0.534206	0.246977	0.827832
0.424605	0.483577	0.502877	0.665460	0.581429	0.026867	0.225661	0.564295	0.916825	0.918621
0.095249	0.858455	0.753667	0.017176	0.725662	0.871140	0.960569	0.226387	0.007518	0.388607
0.910306	0.395152	0.496486	0.111072	0.675211	0.550759	0.486625	0.092152	0.528349	0.199070
0.553084	0.088094	0.457299	0.896147	0.406691	0.074455	0.964274	0.164450	0.815993	0.687412
0.768706	0.621655	0.421898	0.229561	0.906574	0.393865	0.292592	0.592996	0.586007	0.390408
0.987571	0.032175	0.287352	0.157509	0.413616	0.807916	0.651416	0.531929	0.492680	0.210737
0.185603	0.216746	0.500727	0.091846	0.356331	0.256485	0.003402	0.147023	0.779100	0.802814
0.321638	0.992109	0.047623	0.959490	0.341248	0.509921	0.436872	0.203431	0.825862	0.666838
0.114195	0.682683	0.496689	0.031717	0.754262	0.050342	0.605900	0.862184	0.415829	0.680795
0.071057	0.166882	0.018507	0.639291	0.797477	0.941020	0.404908	0.232533	0.304761	0.008831
0.386388	0.725158	0.211232	0.988979	0.866158	0.274317	0.614383	0.213054	0.836525	0.182132
0.704624	0.164737	0.901763	0.267045	0.852089	0.955143	0.159213	0.653136	0.291732	0.261210
0.272987	0.811389	0.893911	0.965509	0.792196	0.131292	0.156508	0.898911	0.605480	0.526320
0.568925	0.909496	0.258061	0.353648	0.031365	0.385119	0.528793	0.046274	0.738967	0.422779
0.019008	0.735993	0.101895	0.927355	0.743529	0.566854	0.126808	0.512698	0.058252	0.044940
0.441131	0.037520	0.940109	0.878131	0.339515	0.485927	0.809447	0.873455	0.149468	0.081648
0.114933	0.316885	0.333011	0.847819	0.476906	0.135511	0.347083	0.471932	0.363945	0.580220
0.790181	0.482797	0.420064	0.929597	0.901095	0.353006	0.413230	0.576554	0.445295	0.186750
0.252471	0.791733	0.282816	0.147892	0.507948	0.734454	0.700479	0.461932	0.242663	0.242099
0.341583	0.542953	0.408042	0.782195	0.501000	0.506250	0.384554	0.189537	0.242136	0.598702
0.414043	0.179934	0.227644	0.961170	0.220407	0.674397	0.164366	0.592372	0.976857	0.205626
0.129098	0.662890	0.949907	0.343986	0.625554	0.046106	0.806396	0.560915	0.503277	0.743254
0.822319	0.367263	0.154040	0.034203	0.306186	0.169487	0.355109	0.824329	0.438895	0.487571
0.234018	0.697288	0.777184	0.668405	0.935416	0.101600	0.681473	0.924376	0.101346	0.775245
0.455645	0.538502	0.650784	0.216937	0.812061	0.960548	0.008595	0.130929	0.652996	0.873695
0.889507	0.746395	0.419872	0.553737	0.912346	0.300009	0.396250	0.286578	0.770391	0.555680
0.865723	0.922751	0.170151	0.223825	0.714050	0.872130	0.734213	0.882448	0.227063	0.010139
0.553033	0.909583	0.321483	0.701374	0.836747	0.219366	0.074711	0.951157	0.918729	0.888182
0.225017	0.834954	0.631695	0.760909	0.868243	0.646226	0.911256	0.970065	0.199157	0.315862
0.236089	0.307107	0.791044	0.052946	0.836638	0.003986	0.325027	0.186772	0.819763	0.697781
0.269564	0.403342	0.189540	0.911499	0.461905	0.228831	0.530750	0.667057	0.185420	0.945136
0.443170	0.499960	0.860903	0.612632	0.307374	0.597665	0.158306	0.154322	0.506223	0.599310
0.130494	0.648335	0.913413	0.958290	0.905309	0.138228	0.515359	0.145960	0.772410	0.001827
0.828361	0.530384	0.001124	0.514248	0.015367	0.900974	0.170895	0.984901	0.957304	0.027185
0.214846	0.142311	0.514181	0.277491	0.990299	0.127475	0.954907	0.206398	0.304055	0.808535
0.697052	0.613712	0.597534	0.830460	0.912718	0.091465	0.348401	0.155681	0.600338	0.818846
0.140689	0.724608	0.641898	0.823427	0.361929	0.947873	0.078036	0.880268	0.594702	0.005303
0.424772	0.457241	0.026270	0.423555	0.219292	0.408080	0.308475	0.829563	0.741765	0.757285
0.964306	0.178763	0.960846	0.773309	0.103234	0.387638	0.220363	0.912006	0.897071	0.961678
0.799563	0.810072	0.356566	0.232173	0.630570	0.745573	0.170146	0.766165	0.752881	0.728873
0.229469	0.857117	0.529329	0.715149	0.639574	0.130560	0.026230	0.415774	0.404955	0.609077
0.976979	0.158649	0.118918	0.258189	0.541438	0.523005	0.469001	0.469147	0.117032	0.000048
0.305378	0.502737	0.543661	0.536388	0.599376	0.694595	0.493948	0.486780	0.376982	0.969306
0.817646	0.119546	0.085852	0.133607	0.417777	0.068566	0.512610	0.507471	0.728248	0.226068
0.374301	0.002361	0.774438	0.426180	0.764080	0.471302	0.837711	0.744671	0.928178	0.290824
0.101885	0.292596	0.506791	0.131052	0.665364	0.964804	0.678677	0.842603	0.843527	0.345834
0.337631	0.561152	0.098337	0.754270	0.715030	0.584655	0.075463	0.811022	0.915820	0.174899
0.442370	0.885066	0.415158	0.195827	0.397353	0.400179	0.174714	0.466093	0.141008	0.533316
0.069583	0.264240	0.624623	0.050815	0.550396	0.631977	0.938991	0.240683	0.878305	0.421256
0.539037	0.501350	0.431711	0.226172	0.144130	0.224316	0.496039	0.474887	0.858920	0.305036
0.669343	0.429893	0.341924	0.871122	0.945624	0.256549	0.173847	0.722818	0.404001	0.764815
0.841348	0.978203	0.568600	0.876969	0.404997	0.734393	0.711508	0.520977	0.946863	0.878213
0.722124	0.356958	0.119716	0.494245	0.636613	0.359822	0.364719	0.678512	0.335901	0.270046
0.610120	0.033260	0.597271	0.320196	0.237956	0.470278	0.463777	0.785385	0.567087	0.714197
0.475870	0.264600	0.533604	0.444765	0.399370	0.490245	0.889717	0.867149	0.061557	0.882697
0.981435	0.983617	0.641974	0.052828	0.274055	0.429202	0.067354	0.409973	0.410602	0.319311
0.964755	0.210119	0.840254	0.058886	0.820353	0.732912	0.445210	0.080427	0.235108	0.386779
0.081944	0.864768	0.495362	0.596824	0.659844	0.649785	0.315729	0.107251	0.571046	0.171885
0.338238	0.945784	0.943118	0.618656	0.416392	0.946980	0.009366	0.958052	0.511938	0.878877
0.071653	0.007943	0.824364	0.399101	0.993856	0.302400	0.944192	0.437315	0.985669	0.571562
0.846882	0.307567	0.645454	0.334082	0.051687	0.860043	0.573380	0.651660	0.897978	0.205433
0.760831	0.759505	0.474456	0.668291	0.137039	0.848405	0.694927	0.317459	0.037335	0.045529

Table A.2: Seed Values for Random Number Generator for String Length 3-7

seed	0	1	2	3	4
1	1401261800	357293397	1460808642	416840239	1520355484
2	819509505	1937462122	907931091	2025883708	996352677
3	745227174	753467483	761707792	769948101	778188410
4	279744999	2125664504	1824100361	1522536218	1220972075
5	764981652	423229905	81478158	1887210059	1545458312
6	1553392701	812858422	72324143	1479273512	738739233
7	1777857586	86853431	543332924	999812417	1456291910
8	1000760707	193217700	1533158341	725615334	2065555975
9	1595495168	1863541261	2131587354	252149799	520195892
10	1729267513	1445914050	1162560587	879207124	595853661
11	388833662	1912938451	1289559592	666180733	42801874
12	65001183	1276315408	340145985	1551460210	615290787
13	692100908	916525577	1140950246	1365374915	1589799584
14	151739893	302619662	453499431	604379200	755258969
15	821417866	1551361583	133821652	863765369	1593709086
16	2132083195	1586070588	1040057981	494045374	2095516415
17	1169064472	692045253	215026034	1885490463	1408471244
18	226550897	2043112218	1712189891	1381267564	1050345237
19	1709302358	212252235	862685760	1513119285	16069162
20	413647575	931702824	1449758073	1967813322	338384923
21	1712695236	2136734017	413289150	837327931	1261366712
22	367941293	157359846	2094262047	1883680600	1673099153
23	1806814690	1638427687	1470040684	1301653681	1133266678
24	641447283	1926504148	1064077365	201650582	1486707447
25	44530736	833480829	1622430922	263897367	1052847460
26	1715908777	831538034	2094650939	1210280196	325909453
27	564323886	723887043	883450200	1043013357	1202576514
28	705836495	46027840	1533702833	873894178	214085523
29	591407964	305500025	19592086	1881167795	1595259856
30	1957736037	1125852350	293968663	1609568624	777684937
31	192624442	11843871	1978546948	1797766377	1616985806
32	1596350443	1783919724	1971489005	11574638	199143919

seed	5	6	7	8	9
1	476387081	1579902326	535933923	1639449168	595480765
2	2114305294	1084774263	55243232	1173195849	143664818
3	786428719	794669028	802909337	811149646	819389955
4	919407932	617843789	316279646	14715503	1860635008
5	1203706565	861954818	520203071	178451324	1984183225
6	2145688602	1405154323	664620044	2071569413	1331035134
7	1912771403	221767248	678246741	1134726234	1591205727
8	1258012968	450469961	1790410602	982867595	175324588
9	788241985	1056288078	1324334171	1592380264	1860426357
10	312500198	29146735	1893276920	1609923457	1326569994
11	1566906663	943527804	320148945	1844253734	1220874875
12	1826605012	890435589	2101749814	1165580391	229410968
13	1814224253	2038648922	115589943	340014612	564439281
14	906138738	1057018507	1207898276	1358778045	1509657814
15	176169155	906112872	1636056589	218516658	948460375
16	1549503808	1003491201	457478594	2058949635	1512937028
17	931452025	454432806	2124897235	1647878016	1170858797
18	719422910	388500583	57578256	1874139577	1543217250
19	666502687	1316936212	1967369737	470319614	1120753139
20	856440172	1374495421	1892550670	263122271	781177520
21	1685405493	2109444274	385999407	810038188	1234076969
22	1462517706	1251936259	1041354812	830773365	620191918
23	964879675	796492672	628105669	459718666	291331663
24	624280664	1909337529	1046910746	184483963	1469540828
25	1841797553	483263998	1272214091	2061164184	702630629
26	1589022358	704651615	1967764520	1083393777	199023034
27	1362139671	1521702828	1681265985	1840829142	2000392299
28	1701760516	1041951861	382143206	1869818199	1210009544
29	1309351917	1023443978	737536039	451628100	165720161
30	2093284898	1261401211	429517524	1745117485	913233798
31	1436205235	1255424664	1074644093	893863522	713082951
32	386713200	574282481	761851762	949421043	1136990324

Table A.3: Fitness Functions for String Length 8

0	1	2	3	4	5	6	7	8	9
0.824259	0.916450	0.153205	0.484968	0.174051	0.527432	0.176080	0.074430	0.591738	0.416248
0.295529	0.963484	0.380519	0.149067	0.812406	0.712910	0.340029	0.236029	0.934362	0.111152
0.801598	0.317741	0.466219	0.063868	0.205717	0.090736	0.509522	0.594275	0.904307	0.065198
0.096171	0.135741	0.385091	0.665998	0.149099	0.233249	0.606022	0.366088	0.232000	0.032443
0.441550	0.935097	0.143528	0.179675	0.255028	0.254582	0.429385	0.203590	0.859314	0.804507
0.284500	0.523968	0.798242	0.397551	0.173007	0.918416	0.989691	0.510805	0.733014	0.453381
0.290514	0.134583	0.301861	0.384340	0.222271	0.344583	0.317159	0.161973	0.989535	0.711432
0.296718	0.538706	0.496920	0.431862	0.775845	0.333944	0.120673	0.656290	0.947971	0.761036
0.985976	0.468786	0.145609	0.462493	0.213017	0.006229	0.727617	0.332131	0.670641	0.123872
0.630534	0.316960	0.613253	0.209315	0.934787	0.922193	0.401736	0.769717	0.567837	0.199197
0.175993	0.182825	0.916529	0.893364	0.707554	0.802387	0.983816	0.529284	0.817150	0.681509
0.424207	0.348579	0.596349	0.128743	0.145894	0.680607	0.942837	0.519338	0.232675	0.090759
0.299580	0.871575	0.023815	0.301216	0.984881	0.480828	0.613692	0.099187	0.097441	0.700449
0.459459	0.465264	0.322669	0.065803	0.365782	0.207918	0.633587	0.647313	0.606821	0.448115
0.874071	0.109825	0.870122	0.033933	0.041278	0.981898	0.879533	0.843572	0.188063	0.971251
0.473440	0.541884	0.857741	0.522136	0.737692	0.603735	0.317352	0.362673	0.841555	0.489846
0.022685	0.420130	0.821323	0.714408	0.739929	0.699134	0.955625	0.049749	0.846739	0.137073
0.232070	0.567479	0.688218	0.334008	0.135802	0.441022	0.518249	0.100990	0.303342	0.707354
0.769011	0.202876	0.505591	0.998387	0.401317	0.103360	0.120926	0.572873	0.732483	0.698500
0.853887	0.539156	0.433379	0.793810	0.408115	0.884163	0.642341	0.879242	0.085805	0.089588
0.207729	0.372050	0.531704	0.312128	0.812772	0.837487	0.795672	0.038448	0.959617	0.930790
0.136100	0.309312	0.677688	0.448086	0.888351	0.824815	0.783090	0.334668	0.604000	0.142036
0.091291	0.578756	0.329286	0.806815	0.490228	0.348670	0.431336	0.793689	0.121446	0.486021
0.558048	0.388545	0.496089	0.114308	0.566774	0.583793	0.650219	0.646527	0.624964	0.934714
0.014167	0.859931	0.950871	0.722774	0.546055	0.238762	0.234359	0.397745	0.489007	0.785470
0.901427	0.639709	0.087962	0.309122	0.414169	0.743119	0.989791	0.930265	0.754095	0.900971
0.142965	0.096309	0.544213	0.411477	0.228485	0.652452	0.988904	0.508991	0.868752	0.264418
0.093073	0.835275	0.170925	0.016473	0.804168	0.657471	0.569695	0.737099	0.255040	0.327652
0.551691	0.051585	0.089583	0.868612	0.429453	0.679081	0.641550	0.634968	0.981676	0.095669
0.704587	0.767524	0.513607	0.306281	0.050047	0.486357	0.429447	0.438760	0.327408	0.757587
0.829392	0.057353	0.040242	0.360789	0.058041	0.326622	0.516654	0.129911	0.563185	0.023560
0.654882	0.704720	0.874417	0.705420	0.650260	0.433296	0.863333	0.670524	0.463447	0.615626
0.407103	0.937966	0.020232	0.752477	0.754238	0.784171	0.330989	0.316516	0.382259	0.870688
0.249861	0.669397	0.856262	0.941106	0.040648	0.033951	0.365592	0.093435	0.328648	0.040054
0.107352	0.518675	0.944438	0.066656	0.956978	0.292570	0.506358	0.509329	0.377387	0.636486
0.361915	0.366601	0.143867	0.577547	0.230188	0.706389	0.666596	0.059782	0.803974	0.523925
0.393970	0.225499	0.690727	0.889127	0.379215	0.682502	0.898714	0.745483	0.249844	0.911347
0.025717	0.071855	0.171972	0.765008	0.058515	0.632483	0.308057	0.678913	0.481125	0.543990
0.026504	0.457645	0.090287	0.362251	0.740420	0.885325	0.590530	0.801599	0.492151	0.073677
0.242676	0.599154	0.357096	0.520302	0.687928	0.175288	0.505004	0.838306	0.474615	0.671013
0.999189	0.894856	0.721458	0.668938	0.031512	0.245998	0.280591	0.165375	0.468828	0.661658
0.082440	0.512283	0.105293	0.517037	0.009103	0.763585	0.031852	0.883820	0.391098	0.146792
0.182194	0.021799	0.446293	0.846371	0.881729	0.038231	0.963650	0.153113	0.618801	0.014788
0.683665	0.565256	0.641915	0.223670	0.527586	0.453235	0.938311	0.593619	0.474823	0.175189
0.157785	0.019635	0.230176	0.252924	0.937536	0.484970	0.447492	0.177913	0.098996	0.886822
0.707823	0.552052	0.994542	0.992056	0.768917	0.527042	0.243033	0.969350	0.421413	0.050206
0.003835	0.203470	0.218318	0.716193	0.664930	0.853224	0.858269	0.584694	0.138543	0.126683
0.550421	0.182744	0.273013	0.246512	0.572949	0.433197	0.666903	0.292864	0.846884	0.466250
0.246557	0.158279	0.005029	0.387552	0.385036	0.395611	0.093965	0.116524	0.565906	0.190200
0.688292	0.255408	0.249338	0.775253	0.689845	0.126785	0.110040	0.183619	0.255948	0.871630
0.858614	0.985489	0.839747	0.073944	0.428592	0.497956	0.946113	0.990651	0.608628	0.074800
0.555895	0.752850	0.962663	0.104460	0.109836	0.783799	0.450010	0.189566	0.373711	0.954157
0.446355	0.229030	0.342736	0.254536	0.868198	0.057034	0.622025	0.802400	0.278252	0.506822
0.097335	0.861914	0.422236	0.624552	0.842382	0.796645	0.269783	0.703818	0.135061	0.925749
0.629317	0.693548	0.522338	0.924302	0.891712	0.592446	0.850095	0.762885	0.442740	0.581623
0.169377	0.211730	0.278788	0.779547	0.523791	0.094136	0.312746	0.403906	0.128291	0.800623
0.888426	0.025518	0.360287	0.396590	0.058168	0.928739	0.009040	0.919513	0.552103	0.240464
0.551737	0.648344	0.609957	0.122762	0.165070	0.056785	0.143931	0.500840	0.575658	0.025144
0.988819	0.617066	0.440653	0.599342	0.192121	0.940678	0.099664	0.856758	0.854612	0.395957
0.079634	0.568496	0.999661	0.602128	0.024840	0.548193	0.762789	0.143808	0.055340	0.280582
0.890530	0.298470	0.107515	0.508424	0.793792	0.235914	0.090977	0.765323	0.483171	0.542034
0.264797	0.062728	0.129889	0.619332	0.163756	0.712100	0.009102	0.483060	0.508410	0.167443
0.270214	0.081062	0.406632	0.069611	0.035425	0.448619	0.530142	0.854691	0.455820	0.687359
0.743301	0.869633	0.788513	0.942191	0.525089	0.830941	0.222613	0.493825	0.196026	0.452859

Table A.3: (continued)

0	1	2	3	4	5	6	7	8	9
0.631345	0.422105	0.891795	0.540377	0.903275	0.676195	0.121145	0.604342	0.099009	0.537539
0.093553	0.670542	0.811236	0.376327	0.698451	0.038802	0.951964	0.645464	0.426052	0.534717
0.242013	0.326780	0.150056	0.282372	0.264165	0.642375	0.979187	0.366225	0.613874	0.075971
0.615915	0.306319	0.381900	0.077547	0.457295	0.027592	0.675381	0.505568	0.622617	0.525260
0.301674	0.445629	0.092493	0.812879	0.428246	0.722948	0.186096	0.469399	0.507825	0.561293
0.166248	0.557773	0.875580	0.041877	0.272361	0.454856	0.636500	0.874222	0.766650	0.921046
0.469605	0.338414	0.639423	0.805942	0.072762	0.750510	0.459083	0.777979	0.703011	0.363163
0.472265	0.237386	0.548310	0.467896	0.166980	0.265938	0.288722	0.756885	0.999855	0.670823
0.985512	0.409200	0.683188	0.946456	0.750767	0.045411	0.424284	0.984465	0.737435	0.517154
0.080719	0.947468	0.256253	0.223134	0.711472	0.976574	0.010886	0.389253	0.476535	0.826870
0.995272	0.553667	0.593632	0.719866	0.979523	0.386207	0.696229	0.888592	0.477177	0.014788
0.651834	0.619201	0.569041	0.207668	0.702936	0.053688	0.345662	0.848882	0.585906	0.976633
0.689745	0.080282	0.334952	0.193549	0.020153	0.767781	0.161065	0.532269	0.325748	0.635215
0.993957	0.716843	0.907050	0.182263	0.647846	0.552025	0.161553	0.089871	0.986385	0.560273
0.928731	0.694997	0.973752	0.190006	0.675062	0.991348	0.800124	0.883642	0.182223	0.353090
0.844789	0.648201	0.672083	0.943227	0.022264	0.144626	0.921613	0.993839	0.360715	0.984847
0.013001	0.614191	0.727676	0.912533	0.356001	0.814380	0.980751	0.010752	0.201992	0.660508
0.591228	0.447965	0.934256	0.288716	0.063416	0.595667	0.844974	0.008152	0.293093	0.239274
0.104254	0.218209	0.730272	0.417132	0.612047	0.716792	0.470031	0.880341	0.400428	0.931695
0.472057	0.483090	0.089922	0.266484	0.404613	0.130889	0.878761	0.491160	0.926336	0.815087
0.814057	0.469054	0.406092	0.797857	0.256255	0.250443	0.338471	0.673437	0.844238	0.215553
0.564595	0.994625	0.910353	0.741457	0.894286	0.614522	0.507552	0.646851	0.054774	0.856118
0.384668	0.623658	0.467784	0.635810	0.614834	0.984677	0.333191	0.815833	0.007627	0.928266
0.663802	0.068333	0.231719	0.810286	0.229149	0.953230	0.108376	0.822691	0.186233	0.417829
0.618515	0.247293	0.964467	0.400729	0.137373	0.357755	0.244447	0.489094	0.229639	0.502515
0.013798	0.848133	0.133201	0.690329	0.258527	0.253768	0.554394	0.863866	0.951336	0.101769
0.119902	0.039821	0.993812	0.295175	0.966023	0.064014	0.687409	0.693557	0.190100	0.447954
0.778055	0.919179	0.308827	0.811581	0.921920	0.654910	0.223333	0.239915	0.627226	0.386088
0.724043	0.626226	0.079479	0.952130	0.630269	0.909535	0.121961	0.209514	0.973300	0.982697
0.860256	0.899871	0.214708	0.320375	0.468059	0.430469	0.954030	0.927377	0.725501	0.152631
0.773071	0.260739	0.717673	0.714360	0.615166	0.424778	0.045921	0.060327	0.771604	0.307850
0.526924	0.657470	0.173149	0.201042	0.864532	0.980061	0.991869	0.408490	0.468973	0.990835
0.096927	0.103083	0.422105	0.570581	0.258336	0.718174	0.607568	0.899355	0.653663	0.629638
0.101475	0.074331	0.190040	0.623237	0.170257	0.061657	0.952763	0.763860	0.142266	0.187918
0.688393	0.011589	0.048283	0.503804	0.548063	0.067028	0.242082	0.705028	0.997646	0.160401
0.505951	0.400434	0.661136	0.045256	0.234599	0.431282	0.101830	0.329031	0.513090	0.910189
0.018078	0.471770	0.659590	0.798507	0.748764	0.759261	0.081968	0.437081	0.095665	0.414991
0.009878	0.486627	0.311268	0.533931	0.017084	0.301199	0.696716	0.494823	0.152158	0.566411
0.621690	0.487747	0.299261	0.056505	0.897887	0.441849	0.866779	0.409222	0.664661	0.113159
0.401768	0.510078	0.332947	0.444325	0.362772	0.250985	0.172353	0.122685	0.205191	0.205353
0.675291	0.641217	0.521662	0.862720	0.333844	0.312405	0.129290	0.172867	0.053956	0.211122
0.267387	0.537524	0.905491	0.785229	0.365176	0.902289	0.101139	0.982499	0.315535	0.835526
0.451642	0.534641	0.232391	0.687329	0.497789	0.067006	0.979979	0.309225	0.973284	0.022461
0.974298	0.745941	0.252814	0.988052	0.463585	0.926145	0.743264	0.311240	0.351916	0.691734
0.283278	0.392860	0.016144	0.826695	0.586128	0.546202	0.931313	0.030381	0.290823	0.710196
0.064723	0.698923	0.611985	0.182845	0.997426	0.977924	0.342544	0.116035	0.387966	0.725505
0.784710	0.588072	0.811004	0.143738	0.908957	0.109459	0.979555	0.588073	0.120664	0.872356
0.224624	0.957185	0.128568	0.586303	0.829019	0.975508	0.900664	0.096822	0.365870	0.822635
0.933222	0.401051	0.645490	0.722033	0.027697	0.699030	0.899484	0.011746	0.346019	0.522629
0.975021	0.768933	0.307451	0.909013	0.933594	0.686910	0.545270	0.992892	0.903277	0.294188
0.959732	0.528674	0.005849	0.306952	0.058817	0.484179	0.075380	0.450252	0.865240	0.832628
0.112475	0.379290	0.798688	0.696843	0.871872	0.581004	0.867644	0.525408	0.855945	0.155946
0.540754	0.436502	0.050410	0.667202	0.533487	0.802566	0.887140	0.273546	0.535111	0.184746
0.409957	0.181190	0.191925	0.749236	0.567366	0.018151	0.576112	0.927315	0.730319	0.534728
0.970230	0.608893	0.070840	0.227831	0.909923	0.406163	0.176692	0.433498	0.424422	0.145009
0.104421	0.764635	0.718646	0.339335	0.038743	0.696135	0.129276	0.195852	0.630035	0.546705
0.996626	0.567459	0.389132	0.805746	0.440115	0.320628	0.344396	0.746109	0.772389	0.905080
0.140538	0.252449	0.960016	0.659135	0.093908	0.580718	0.026423	0.602365	0.471608	0.888053
0.927522	0.294730	0.333617	0.573743	0.909232	0.960564	0.433299	0.800540	0.624971	0.364859
0.795723	0.045195	0.431357	0.767623	0.193647	0.291666	0.084265	0.140368	0.994735	0.651105
0.148170	0.086003	0.215989	0.243370	0.523597	0.695595	0.554532	0.437142	0.670985	0.506055
0.459726	0.851787	0.328154	0.272012	0.055678	0.449311	0.762366	0.283156	0.550853	0.796752
0.850575	0.749639	0.249049	0.411391	0.269093	0.824089	0.421943	0.347663	0.335194	0.557664
0.583744	0.899122	0.350242	0.502131	0.387995	0.794426	0.251931	0.861780	0.532245	0.311801

Table A.3: (continued)

0	1	2	3	4	5	6	7	8	9
0.405427	0.306251	0.734591	0.360413	0.377628	0.664169	0.881596	0.216386	0.422579	0.615747
0.727885	0.016143	0.688141	0.934637	0.614347	0.483919	0.595090	0.906093	0.161642	0.179262
0.200192	0.084560	0.336650	0.520339	0.205148	0.986682	0.365683	0.539657	0.612622	0.954172
0.715447	0.334341	0.082138	0.205497	0.556568	0.841636	0.417801	0.221029	0.973832	0.943480
0.710679	0.323983	0.890905	0.355568	0.061718	0.005823	0.230240	0.059490	0.695561	0.850077
0.864009	0.996074	0.361767	0.007162	0.677637	0.013424	0.457580	0.767607	0.794257	0.627585
0.060079	0.060128	0.861079	0.799490	0.113307	0.035166	0.942057	0.405767	0.240051	0.112491
0.788377	0.657006	0.599108	0.326229	0.526983	0.838872	0.080086	0.913930	0.836123	0.837873
0.658006	0.046914	0.288766	0.566683	0.035719	0.896638	0.945490	0.996406	0.947074	0.716645
0.129233	0.449276	0.422821	0.508118	0.678453	0.029882	0.924761	0.887448	0.497151	0.637507
0.512324	0.954415	0.084073	0.959532	0.345796	0.646710	0.803381	0.040908	0.061096	0.982459
0.701582	0.089764	0.607404	0.101013	0.384383	0.669439	0.470827	0.148029	0.988293	0.059607
0.023841	0.558123	0.859942	0.074255	0.360799	0.811956	0.620411	0.373304	0.519664	0.671372
0.974710	0.442468	0.275860	0.886573	0.047468	0.966526	0.757079	0.888519	0.277308	0.393538
0.693572	0.459440	0.160879	0.582456	0.319227	0.547068	0.931684	0.389193	0.761811	0.272820
0.514094	0.482658	0.245821	0.061394	0.098630	0.661621	0.115170	0.293242	0.599604	0.955810
0.017172	0.280674	0.744069	0.884582	0.818412	0.933140	0.209998	0.117757	0.178947	0.196689
0.979364	0.787372	0.033796	0.636041	0.872115	0.483296	0.660985	0.091191	0.718492	0.559901
0.850533	0.624449	0.975210	0.237838	0.012688	0.694346	0.790034	0.439374	0.002255	0.021229
0.928320	0.581031	0.648121	0.184507	0.436622	0.617868	0.037696	0.069146	0.978564	0.331592
0.712087	0.813868	0.998718	0.077005	0.944462	0.734874	0.399498	0.490235	0.054516	0.646577
0.313344	0.408952	0.389518	0.442348	0.559488	0.975467	0.283555	0.777171	0.220751	0.511098
0.676349	0.907831	0.924100	0.789651	0.595439	0.155972	0.569926	0.060827	0.133780	0.433171
0.513183	0.203961	0.071863	0.068450	0.870341	0.923748	0.355637	0.037575	0.121418	0.317837
0.696048	0.768080	0.455449	0.262824	0.792629	0.397488	0.071167	0.547474	0.719687	0.572170
0.960507	0.995445	0.360142	0.569166	0.933715	0.583110	0.646992	0.798935	0.836004	0.981139
0.305759	0.315874	0.324486	0.524917	0.029451	0.444600	0.736147	0.789374	0.900468	0.956017
0.302630	0.137429	0.577452	0.593010	0.192196	0.917625	0.664167	0.216052	0.121833	0.471510
0.299199	0.162644	0.420363	0.178363	0.955366	0.295376	0.466476	0.435333	0.456597	0.716334
0.757681	0.491673	0.937494	0.049344	0.220250	0.428425	0.409199	0.641615	0.870404	0.485574
0.341689	0.449950	0.471868	0.644836	0.249464	0.215819	0.230295	0.716918	0.965140	0.092861
0.886914	0.984212	0.922554	0.536491	0.806861	0.473533	0.049203	0.258937	0.217834	0.373249
0.609381	0.490610	0.616725	0.218546	0.329457	0.005651	0.155649	0.986093	0.368460	0.118881
0.322409	0.085365	0.809570	0.179210	0.819335	0.037125	0.055218	0.421777	0.476132	0.384954
0.461974	0.791525	0.168741	0.028521	0.117789	0.279435	0.939883	0.270331	0.290820	0.709275
0.581696	0.303096	0.408740	0.725682	0.201745	0.876763	0.460486	0.882352	0.302530	0.650589
0.040882	0.140800	0.752043	0.108590	0.636627	0.165968	0.722132	0.742731	0.868302	0.054133
0.809999	0.145605	0.535144	0.257164	0.861489	0.142933	0.222477	0.699554	0.843356	0.478818
0.531052	0.497745	0.967689	0.003848	0.509792	0.428441	0.968980	0.707628	0.604058	0.549815
0.419128	0.918394	0.399669	0.660639	0.929067	0.037409	0.784314	0.718504	0.746415	0.566818
0.957849	0.488259	0.563382	0.024431	0.115181	0.753770	0.335272	0.875135	0.724330	0.097499
0.980368	0.741303	0.972053	0.670912	0.186702	0.000883	0.414395	0.359060	0.146749	0.272726
0.261942	0.754841	0.823478	0.459005	0.859183	0.886658	0.077610	0.086033	0.853690	0.134717
0.612434	0.855471	0.402289	0.482695	0.096865	0.184697	0.849445	0.204401	0.556546	0.853154
0.697871	0.367322	0.193206	0.672232	0.622904	0.283277	0.176614	0.437080	0.675802	0.888151
0.656885	0.199941	0.681321	0.785482	0.350435	0.430696	0.893137	0.656327	0.203671	0.633911
0.428073	0.856804	0.794546	0.549684	0.443304	0.540162	0.559350	0.135024	0.496255	0.113534
0.483443	0.363498	0.317269	0.737296	0.569774	0.396880	0.988758	0.708534	0.650971	0.587243
0.444490	0.484369	0.504567	0.396311	0.301279	0.183231	0.955256	0.054891	0.751921	0.315883
0.967015	0.299285	0.973476	0.004576	0.975517	0.377454	0.786306	0.001606	0.788967	0.383719
0.489964	0.729321	0.106871	0.242706	0.164196	0.847067	0.348118	0.350994	0.094267	0.695087
0.845539	0.948575	0.638537	0.650360	0.331401	0.777970	0.890365	0.221090	0.549152	0.034545
0.160527	0.689143	0.907791	0.093649	0.100312	0.153935	0.295890	0.847823	0.094256	0.080418
0.092894	0.257966	0.311554	0.362047	0.048843	0.395663	0.012745	0.706048	0.464790	0.072090
0.242055	0.449172	0.878374	0.857295	0.138531	0.578607	0.021636	0.144862	0.567105	0.218940
0.518513	0.322039	0.812037	0.823922	0.570767	0.190636	0.832393	0.957449	0.204745	0.242498
0.118505	0.525533	0.071416	0.716092	0.284890	0.478267	0.439441	0.919999	0.793182	0.060381
0.877783	0.999196	0.527080	0.341129	0.385812	0.949557	0.310465	0.117880	0.136490	0.569217
0.253474	0.542816	0.913397	0.176977	0.438779	0.562201	0.477918	0.950698	0.683012	0.234205
0.128983	0.020887	0.482165	0.629886	0.859973	0.129060	0.769754	0.177138	0.393031	0.199487
0.823127	0.855273	0.609724	0.898572	0.041010	0.847456	0.735448	0.024876	0.925955	0.573452
0.250080	0.914524	0.325935	0.542325	0.251819	0.892233	0.719581	0.706212	0.396695	0.633673
0.257325	0.159261	0.631419	0.322381	0.017191	0.410431	0.111107	0.206302	0.232064	0.288059
0.238878	0.128520	0.889097	0.906044	0.106651	0.859229	0.161176	0.277902	0.200660	0.149827

Table A.3: (continued)

0	1	2	3	4	5	6	7	8	9
0.171384	0.961016	0.859439	0.483687	0.563272	0.276112	0.589489	0.012313	0.772821	0.540008
0.531956	0.213113	0.112020	0.288620	0.159094	0.645827	0.388986	0.681848	0.914347	0.709733
0.439639	0.334922	0.783926	0.642008	0.525200	0.782780	0.393217	0.061996	0.134603	0.924890
0.411407	0.702652	0.457654	0.591560	0.263934	0.627259	0.770967	0.168903	0.963118	0.818218
0.276840	0.075146	0.082653	0.214342	0.424564	0.683249	0.580465	0.451439	0.601506	0.505387
0.036687	0.259499	0.479558	0.796973	0.968792	0.116372	0.038547	0.732866	0.558140	0.638909
0.086021	0.625854	0.451275	0.511710	0.655325	0.121458	0.555820	0.158218	0.103349	0.842276
0.533730	0.917176	0.426800	0.147286	0.248638	0.536260	0.221240	0.409223	0.527976	0.609446
0.534874	0.303003	0.529229	0.239730	0.570837	0.300065	0.705729	0.036300	0.966571	0.244019
0.883518	0.325164	0.001734	0.233261	0.037172	0.316891	0.003728	0.437769	0.213288	0.637510
0.438356	0.851709	0.541250	0.941801	0.272426	0.770802	0.689578	0.718152	0.884297	0.636505
0.866547	0.865293	0.360182	0.426644	0.613061	0.956903	0.509133	0.269145	0.505364	0.612032
0.152817	0.719809	0.481727	0.348700	0.459176	0.821532	0.987665	0.929348	0.126494	0.430681
0.583454	0.649865	0.612546	0.427604	0.546595	0.760309	0.557181	0.354779	0.668990	0.361081
0.271128	0.754789	0.193489	0.211155	0.731810	0.345141	0.334002	0.892713	0.554313	0.098897
0.177535	0.446041	0.643412	0.438902	0.221861	0.206851	0.238774	0.590025	0.514942	0.329672
0.842003	0.469912	0.288726	0.853074	0.648826	0.104842	0.207551	0.878936	0.042822	0.920759
0.427976	0.316678	0.797406	0.183787	0.643639	0.495042	0.425683	0.671495	0.763978	0.386800
0.049157	0.594613	0.664055	0.416033	0.753417	0.355424	0.186249	0.265354	0.438821	0.237305
0.170217	0.141757	0.938198	0.548477	0.095393	0.166316	0.696722	0.258196	0.063566	0.516847
0.934554	0.636400	0.327770	0.150772	0.179240	0.580970	0.673751	0.616739	0.944449	0.912122
0.091609	0.535426	0.145933	0.102510	0.997646	0.323586	0.510715	0.010706	0.568444	0.459188
0.629589	0.824951	0.291135	0.214110	0.789670	0.063102	0.938097	0.052635	0.985769	0.085191
0.370503	0.362090	0.727628	0.312502	0.222806	0.146422	0.994473	0.708192	0.167801	0.969054
0.151025	0.124348	0.950131	0.695523	0.256064	0.761013	0.465729	0.409464	0.703311	0.844947
0.930018	0.578412	0.056721	0.739901	0.958695	0.633608	0.550897	0.588483	0.376473	0.999542
0.142057	0.968174	0.624815	0.083674	0.676545	0.093983	0.067269	0.820356	0.167927	0.725700
0.629474	0.438148	0.294389	0.517030	0.372693	0.538709	0.951166	0.573828	0.905185	0.235916
0.533160	0.070459	0.452491	0.042823	0.436925	0.459684	0.642012	0.248115	0.241850	0.973430
0.494365	0.238246	0.488131	0.206875	0.541000	0.312069	0.930432	0.974762	0.045919	0.910905
0.333107	0.292540	0.948394	0.148929	0.273742	0.915950	0.228493	0.560286	0.643066	0.724542
0.424119	0.571084	0.136582	0.877145	0.866543	0.217510	0.114032	0.465913	0.196354	0.488053
0.445495	0.438300	0.442825	0.431182	0.615865	0.700818	0.708666	0.322760	0.180178	0.028708
0.378424	0.429677	0.821744	0.225744	0.822011	0.569767	0.073882	0.648265	0.640402	0.497207
0.174077	0.003762	0.861039	0.540895	0.824439	0.413896	0.159867	0.486249	0.672249	0.216649
0.831323	0.502029	0.833025	0.245588	0.009843	0.326374	0.667481	0.167935	0.170438	0.276119
0.504068	0.480131	0.199594	0.436782	0.891259	0.609163	0.905472	0.726979	0.077177	0.203230
0.844618	0.206939	0.182000	0.122079	0.896709	0.779853	0.002169	0.780245	0.827265	0.752453
0.212314	0.608709	0.123780	0.526141	0.837964	0.051739	0.654757	0.815821	0.096658	0.488346
0.266955	0.038328	0.861155	0.957409	0.079418	0.976379	0.716483	0.464866	0.236979	0.986211
0.125012	0.829373	0.684750	0.151502	0.326691	0.272612	0.578755	0.122670	0.746145	0.462961
0.061988	0.412643	0.309466	0.058919	0.520557	0.352025	0.922436	0.679499	0.330289	0.308287
0.796383	0.353961	0.974482	0.234327	0.577983	0.422547	0.704116	0.955737	0.110330	0.797240
0.990310	0.547386	0.969593	0.545171	0.004920	0.987619	0.599168	0.589627	0.030690	0.563571
0.158340	0.621566	0.983784	0.211275	0.869603	0.814693	0.338994	0.089309	0.520341	0.159968
0.150446	0.913745	0.732441	0.754785	0.140885	0.255021	0.510921	0.134156	0.998661	0.759751
0.888924	0.497088	0.520902	0.609813	0.781098	0.127534	0.894297	0.904814	0.218976	0.157308
0.748002	0.163443	0.343788	0.403590	0.062084	0.331845	0.444968	0.211808	0.625381	0.091326
0.726758	0.874847	0.576949	0.645606	0.129749	0.188545	0.844736	0.708416	0.433179	0.724270
0.323456	0.964403	0.856676	0.437076	0.480084	0.928592	0.927020	0.362215	0.306538	0.234663
0.986926	0.052713	0.857350	0.546212	0.183429	0.035077	0.702486	0.356781	0.225104	0.473788
0.193653	0.417125	0.315335	0.381541	0.668317	0.308746	0.784282	0.451049	0.020770	0.337536
0.716920	0.844065	0.873444	0.499503	0.814894	0.432549	0.077569	0.458097	0.154845	0.660243
0.762960	0.921015	0.143288	0.115507	0.476191	0.098363	0.180674	0.231540	0.186146	0.377153
0.905771	0.835980	0.940703	0.757115	0.832909	0.943279	0.932683	0.717615	0.557594	0.425284
0.747265	0.389933	0.722857	0.606542	0.696729	0.058601	0.475456	0.546400	0.576468	0.051954
0.086461	0.774813	0.669195	0.857438	0.543230	0.945009	0.680320	0.359088	0.734169	0.681025
0.061215	0.905245	0.962181	0.416264	0.703189	0.213014	0.319335	0.413731	0.494201	0.427682
0.237330	0.698890	0.596615	0.050665	0.439495	0.213363	0.611100	0.682654	0.290869	0.601568
0.445516	0.573117	0.249628	0.968754	0.414721	0.356875	0.912983	0.283504	0.431068	0.229268
0.532619	0.779368	0.279963	0.360191	0.077533	0.507209	0.133075	0.005887	0.480963	0.435680
0.241403	0.418915	0.269276	0.389631	0.758616	0.341605	0.553651	0.377973	0.276084	0.089823
0.321415	0.308466	0.303020	0.621144	0.410675	0.484521	0.566483	0.593401	0.431318	0.121101
0.267919	0.264675	0.668073	0.282321	0.491419	0.323686	0.989246	0.571434	0.729592	0.257808

Table A.4: Seed Values for Random Number Generator for String Length 8

seed	0	1	2	3	4
1	1401261800	357293397	1460808642	416840239	1520355484
2	819509505	1937462122	907931091	2025883708	996352677
3	745227174	753467483	761707792	769948101	778188410
4	279744999	2125664504	1824100361	1522536218	1220972075
5	764981652	423229905	81478158	1887210059	1545458312
6	1553392701	812858422	72324143	1479273512	738739233
7	1777857586	86853431	543332924	999812417	1456291910
8	1000760707	193217700	1533158341	725615334	2065555975
9	1595495168	1863541261	2131587354	252149799	520195892
10	1729267513	1445914050	1162560587	879207124	595853661
11	388833662	1912938451	1289559592	666180733	42801874
12	65001183	1276315408	340145985	1551460210	615290787
13	692100908	916525577	1140950246	1365374915	1589799584
14	151739893	302619662	453499431	604379200	755258969
15	821417866	1551361583	133821652	863765369	1593709086
16	2132083195	1586070588	1040057981	494045374	2095516415
17	1169064472	692045253	215026034	1885490463	1408471244
18	226550897	2043112218	1712189891	1381267564	1050345237
19	1709302358	212252235	862685760	1513119285	16069162
20	413647575	931702824	1449758073	1967813322	338384923
21	1712695236	2136734017	413289150	837327931	1261366712
22	367941293	157359846	2094262047	1883680600	1673099153
23	1806814690	1638427687	1470040684	1301653681	1133266678
24	641447283	1926504148	1064077365	201650582	1486707447
25	44530736	833480829	1622430922	263897367	1052847460
26	1715908777	831538034	2094650939	1210280196	325909453
27	564323886	723887043	883450200	1043013357	1202576514
28	705836495	46027840	1533702833	873894178	214085523
29	591407964	305500025	19592086	1881167795	1595259856
30	1957736037	1125852350	293968663	1609568624	777684937
31	192624442	11843871	1978546948	1797766377	1616985806
32	1596350443	1783919724	1971489005	11574638	199143919

seed	5	6	7	8	9
1	476387081	1579902326	535933923	1639449168	595480765
2	2114305294	1084774263	55243232	1173195849	143664818
3	786428719	794669028	802909337	811149646	819389955
4	919407932	617843789	316279646	14715503	1860635008
5	1203706565	861954818	520203071	178451324	1984183225
6	2145688602	1405154323	664620044	2071569413	1331035134
7	1912771403	221767248	678246741	1134726234	1591205727
8	1258012968	450469961	1790410602	982867595	175324588
9	788241985	1056288078	1324334171	1592380264	1860426357
10	312500198	29146735	1893276920	1609923457	1326569994
11	1566906663	943527804	320148945	1844253734	1220874875
12	1826605012	890435589	2101749814	1165580391	229410968
13	1814224253	2038648922	115589943	340014612	564439281
14	906138738	1057018507	1207898276	1358778045	1509657814
15	176169155	906112872	1636056589	218516658	948460375
16	1549503808	1003491201	457478594	2058949635	1512937028
17	931452025	454432806	2124897235	1647878016	1170858797
18	719422910	388500583	57578256	1874139577	1543217250
19	666502687	1316936212	1967369737	470319614	1120753139
20	856440172	1374495421	1892550670	263122271	781177520
21	1685405493	2109444274	385999407	810038188	1234076969
22	1462517706	1251936259	1041354812	830773365	620191918
23	964879675	796492672	628105669	459718666	291331663
24	624280664	1909337529	1046910746	184483963	1469540828
25	1841797553	483263998	1272214091	2061164184	702630629
26	1589022358	704651615	1967764520	1083393777	199023034
27	1362139671	1521702828	1681265985	1840829142	2000392299
28	1701760516	1041951861	382143206	1869818199	1210009544
29	1309351917	1023443978	737536039	451628100	165720161
30	2093284898	1261401211	429517524	1745117485	913233798
31	1436205235	1255424664	1074644093	893863522	713082951
32	386713200	574282481	761851762	949421043	1136990324

Appendix B

Tables of Select Numerical Data

All tables show the string length across rows and population size down columns.

Table B.1: Average Deviation Values

	16	32	64	128	256	512
3	1.095667	0.773651	0.222848	0.135635	0.100438	0.095611
4	2.317238	1.655882	1.197987	0.725938	0.589234	0.432046
5	3.416066	2.812707	2.061991	1.605647	1.149561	0.823731
6	4.051885	3.481512	3.338597	2.615676	1.888999	1.469145
7	4.346070	4.276920	3.982309	3.284968	2.622981	1.745336

Table B.2: Adjusted Average Deviation Values

	16	32	64	128	256	512
3	0.362822	0.220546	0.148979	0.100831	0.068289	0.049605
4	0.461496	0.289939	0.207595	0.157416	0.119498	0.085713
5	0.685098	0.535180	0.394036	0.277180	0.191855	0.141895
6	0.762199	0.656226	0.525618	0.397897	0.279177	0.211979
7	0.741824	0.720218	0.780004	0.800035	0.598644	0.456905

Table B.3: Predicted Adjusted Average Deviation Values

	16	32	64	128	256	512
3	0.243124	0.179344	0.132295	0.097589	0.071988	0.053103
4	0.445891	0.328918	0.242630	0.178980	0.132027	0.097391
5	0.713739	0.526499	0.388379	0.286493	0.211335	0.155894
6	1.048269	0.773269	0.570412	0.420772	0.310388	0.228962
7	1.450818	1.070215	0.789458	0.582354	0.429581	0.316886

Table B.4: Average Number of Fixed Points

	16	32	64	128	256	512
3	3	3	3	3	3	3
4	4	4	4	4	3	4
5	5	5	5	5	5	5
6	11	10	8	9	7	8
7	13	12	8	7	6	6

Table B.5: Proportion Slope

	16	32	64	128	256	512
3	-21.87241954	-24.39347936	-29.13942191	-29.48016412	-55.01747976	-101.45891
4	-15.05171113	-26.93668	-10.39770647	-27.47876176	-56.11791786	-77.3715
5	-29.48249577	-19.69548254	-13.17794742	-28.5640015	-29.10349676	-68.12440857
6	-31.04437368	-14.72126591	-21.19365696	-4.213182823	-20.30721438	-37.44679538
7	-36.76626347	-35.40181752	-6.944031672	-4.236903313	-8.218160724	-4.091274803

Table B.6: Average Shifts

	16	32	64	128	256	512
3	4.910	2.730	3.670	2.530	1.100	0.820
4	15.520	8.210	5.450	2.610	1.500	0.500
5	31.880	19.410	10.140	5.830	3.580	2.540
6	58.640	35.010	24.050	19.050	13.780	8.670
7	78.860	51.860	26.840	15.450	11.950	9.020

Table B.7: Equation Shifts

	16	32	64	128	256	512
3	5.886659	3.690364	2.313500	1.450340	0.909223	0.569994
4	14.523264	9.104677	5.707749	3.578205	2.243187	1.406261
5	29.260015	18.343190	11.499400	7.209008	4.519348	2.833193
6	51.859995	32.511185	20.381358	12.777134	8.010023	5.021507
7	84.136648	52.745514	33.066319	20.729373	12.995306	8.146796

Table B.8: Revised Proportion Slope

	16	32	64	128	256	512
	-21.87241954	-24.39347936	-29.13942191	-29.48016412	-55.01747976	-101.45891
	-15.05171113	-26.93668	-10.39770647	-27.47876176	-56.11791786	-77.3715
	-29.48249577	-19.69548254	-13.17794742	-28.5640015	-29.10349676	-68.12440857
	-31.04437368	-14.72126591	-21.19365696	-4.213182823	-20.30721438	-37.44679538
	-36.76626347	-35.40181752	-6.944031672	-4.236903313	-8.218160724	-4.091274803
	-45.48942168	-23.87133789	-20.46581409	-51.14416364	-4.717744440	-18.0910035

VITA

Paul Scott Hethmon was born January 7, 1964 in Carbondale, Illinois. He graduated from Union City High School, Union City, Tennessee in 1982 and received a Bachelor of Arts degree in Political Science from the University of Tennessee in June, 1986. He is married to Marci Hethmon and has one son, Zachariah Scott. In August 1991 he entered the Computer Science program at the University of Tennessee and was awarded the Master of Science degree in Computer Science in December 1993.