HYBRIDIZING STATISTICS WITH GENETIC ALGORITHMS

by

TRENTON L. PAMPLIN

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Statistics

MONTANA STATE UNIVERSITY
Bozeman, Montana

April 1995

Math Library Copy

APPROVAL

of a thesis submitted by

TRENTON L. PAMPLIN

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

Date	Jeffrey D. Banfield Chairperson, Graduate Committee

Approved for the Major Department

Date	John Lund Head, Mathematical Sciences

Approved for the College of Graduate Studies

Date	Robert Brown Graduate Dean

STATEMENT OF PERMISSION TO USE

Signature ______________________________

Date ______________________________

ACKNOWLEDGEMENTS

I would like to thank:

My mom and dad for instilling a strong will and desire to succeed and survive any challenge placed before me during my life that I have lived, and yet to live.

My brother, Nathan Pamplin, for being supportive, encouraging, and my best friend since the day he was born.

Joanna Rosińska, for being understanding, accepting, and so caring throughout my graduate school career.

Pete Islieb, for his past generosity and the opportunity to spend several summers working in Bristol Bay, Alaska learning about myself and what can be achieved if I have the right state of mind.

My graduate committee, especially Jeff Banfield and John Paxton, whose time, patience, and expertise helped make this thesis possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

A genetic algorithm (GA) is an adaptive search strategy based on simplified rules of biological population genetics and theories of evolution. The basic concepts of GAs are presented along with their known ability to optimize numerical functions. However, knowledge of these algorithms has been slow to reach the statistical community. In order to show the importance and practical use of GAs to statisticians, a GA is implemented and applied to estimating unknown parameters in linear and nonlinear models. In this problem realm, the GA is demonsrated to be effective by comparing evolved solutions to least squares estimates. The results from simulations indicate the applied GA is a useful tool in fitting linear and nonlinear models.

CHAPTER 1

Introduction

A genetic algorithm is an adaptive search strategy based on simplified rules of biological population genetics and theories of evolution. A genetic algorithm (GA) maintains a population of candidate solutions for a problem, and then uses a biased sampling procedure to select the solutions that seem to work well for the problem. After selecting the "best" candidate solutions, those solutions are combined and/or altered by reproduction operators to produce new solutions for the next generation. The process continues, with each generation providing better solutions, until an acceptable solution is evolved.

Although the foundations of todays GAs were created in the late sixties by John Holland and were successfully applied to a wide variety of problems, it wasn't until the mid 1980's before the algorithms found their way into other disciplines outside the artificial intelligence community. The GAs of today are used to find solutions to complex problems in optimization, machine-learning, programming, and job scheduling. The widespread use and interest is due to the fact that GAs are relatively easy to implement, the objective function does not have to be differentiable, and GAs search a space in parallel for a global optimum which reduces the chance of reporting local extrema. Keeping those benefits in mind, a GA is a useful tool that modern day statisticians should be aware of. There are three main goals of this thesis:

1. Introduce GAs into the statistics literature in order to explain what they are and how they can be used.

2. Demonstrate how GAs can be developed and applied to linear and nonlinear regression problems.

3. Stimulate interest in statisticians to possibly apply GAs to solve problems that are difficult to address with current procedures and further the evolution of both fields.

Chapter 1 describes the terminology associated with a genetic algorithm and looks at the simple genetic algorithm (SGA). Chapter 2 examines a GA applied to simple linear regression and how the applied GA differs from the SGA. Chapter 3 discusses a GA applied to nonlinear regression problems by estimating the relevant parameters of a variogram and identifying an appropriate variogram model. The final chapter contains the conclusions and areas of future work.

CHAPTER 2

Terminology and the Simple Genetic Algorithm

The terminology used in describing the components of a GA, like the name itself, is a blend of terms used in biology and computer science. The first step in any GA is to generate (usually randomly) a population of candidate solutions called chromosomes. Analogous to genetics, chromosomes are made up of genes. In the computer a chromosome is represented by a string of bits (the genes) that usually take on 1's or 0's. Once the initial population has been generated, the next step is to calculate the chromosome evaluated in some objective function .

For example, say we are interested in maximizing the function $F(x, y) = x + y$ with respect to x and y, where x and y can take on integer values between 0 and 15. One representation, would be to let the chromosome be made up of eight bits, four bits for x and four bits for y. An example chromosome c, could be equal to 10110101, implying that $x = 1011$ and $y = 0101$ in binary, or equivalently $x = 11$ and $y = 5$ in decimal. An obvious fitness function, given that representation of a chromosome, could be the function of interest evaluated at the decimal values of x and y for that particular chromosome. Therefore, the objective function evaluated at chromosome c, F_c, equals $11 + 5 = 16$. In this setting, the objective function can be graphed to give us a view of the space the GA is searching. Figure 1 displays the objective function surface over the parametric support.

Once the initial population of chromosomes has been created and each chromosome's fitness calculated, then the selection and reproduction process can take place. The selection process is an important component of a GA. We want to select chromosomes that seem to be doing well relative to the rest of the population so that

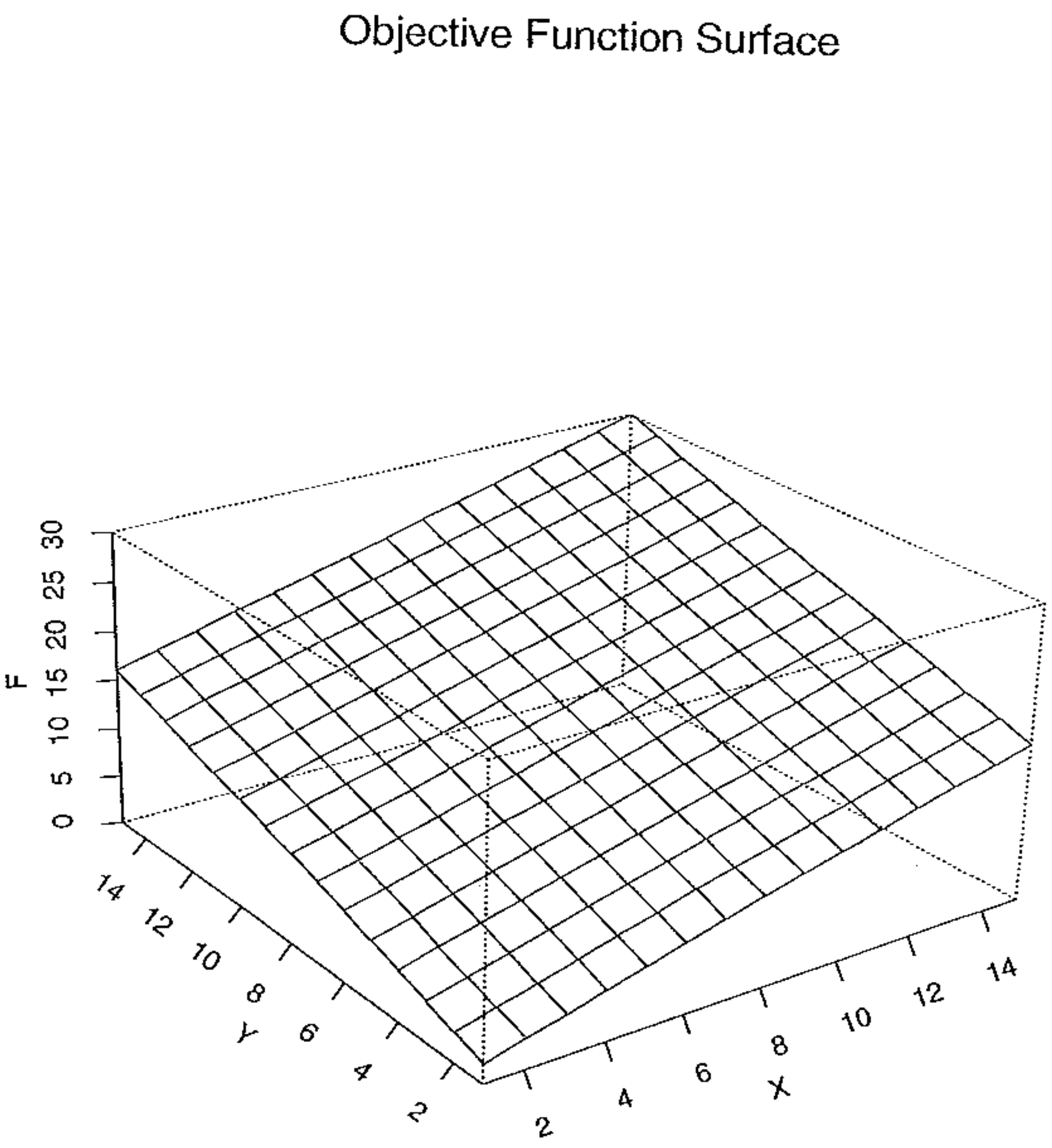


Figure 1: Objective Function Surface for F(x,y) = X + Y

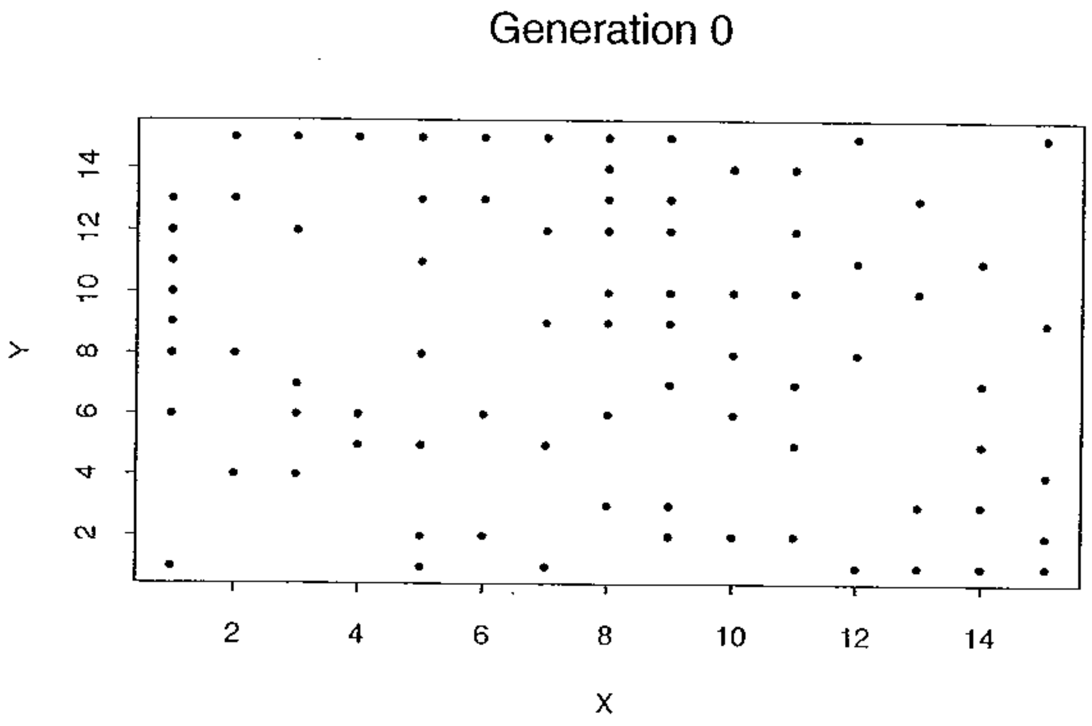
they can pass on their good traits to future chromosomes. Likewise, we do not want to select chromosomes for the reproduction that seem to be performing poorly. A common selection technique is to select chromosomes from the population proportional to their fitness. Thus, chromosomes with larger fitness values get selected a higher precentage of the time. The reason for selecting the chromosomes in the first place is for reproduction purposes, that is, to produce new chromosomes.

In terms of a GA, reproduction is the process of combining one, or more chromosomes to produce new chromosomes. The SGA uses two reproduction operators to perform this task: crossover and mutation. The crossover operator takes two selected chromosomes and then randomly selects a point to "cut" the chromosomes. The "cut" pieces are then recombined with the opposite chromosome from which it originally belonged, thus producing two new chromosomes. For example, if p1 = abcdefgh and p2 = ABCDEFGH, and the crossover point was 3, then the new chromosomes c1 = abcDEFGH and c2 = ABCdefgh would be produced. This type of crossover operator is referred to as a one-point crossover.

The ideal situation for applying the crossover operator is to take the "best" part of each chromosome, yielding new chromosomes with potentially higher F values. Because the crossover point is randomly selected, we do not know what parts of the original chromosomes are good or bad, leading to children with possibly lower fitness values than the original chromosomes. However, over the long run (generation after generation), certain substrings of the chromosomes become prominent in the population. These substrings (schema) identify what values of the genes are good and in what location. The crossover operator plays an important role in evolving the chromosome that optimizes the objective function. However, a GA that only uses a crossover operator is impaired in its ability to find the global optimum. Theoretically, all the chromosomes in a population could have the same values at a particular gene or substring of genes. Thus, any crossover would never change the value of that gene, due to invariance on the chromosomes selected and the choice of the crossover point. Thus, the algorithm may never find the global optimum, the most fit chromosome, simply because it cannot search the space where the global optimum exists.

The mutation operator solves this problem by introducing diversity into the population. It does so by taking a selected chromosome and sweeps down through each gene randomly changing the gene's current value if a probability test is passed. The probabiltiy that mutation occurs (the mutation rate) is usually very low. Intuitively, this should be clear since we want to exploit fit chromosomes and not turn a GA into strictly a random search. However, we do want to explore the search space and maintain some diversity in the population in order to find the global optimum and not get fooled by local extrema.

The reproductive operators are applied to selected chromosomes until N new chromosomes have been produced. These N new chromosomes replace the old ones to form the next generation. The objective function is evaluated for each chromosome in the current population, selection and reproduction take place, and the process starts again. After the final generation, the chromosome with the highest objective function evaluation is reported. Figure 2 and Figure 3 display how the chromosomes search the space shown in Figure 1. The population size was kept constant at 100 chromosomes. These figures illustrate how the majority of the population climb towards the maximum and how some chromosomes keep searching the space for other areas of high fitness. At generation 30, most of the 100 chromosomes are near the maximum at X and Y equal to 15. For this example, the best chromosome evolved would be 11111111.



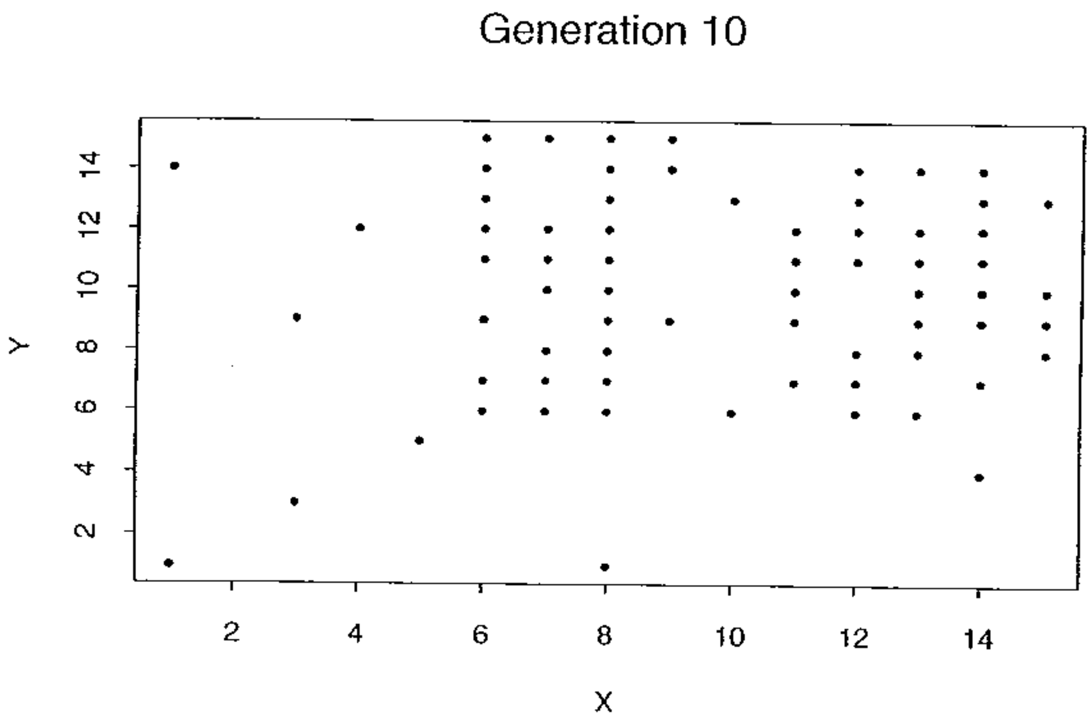
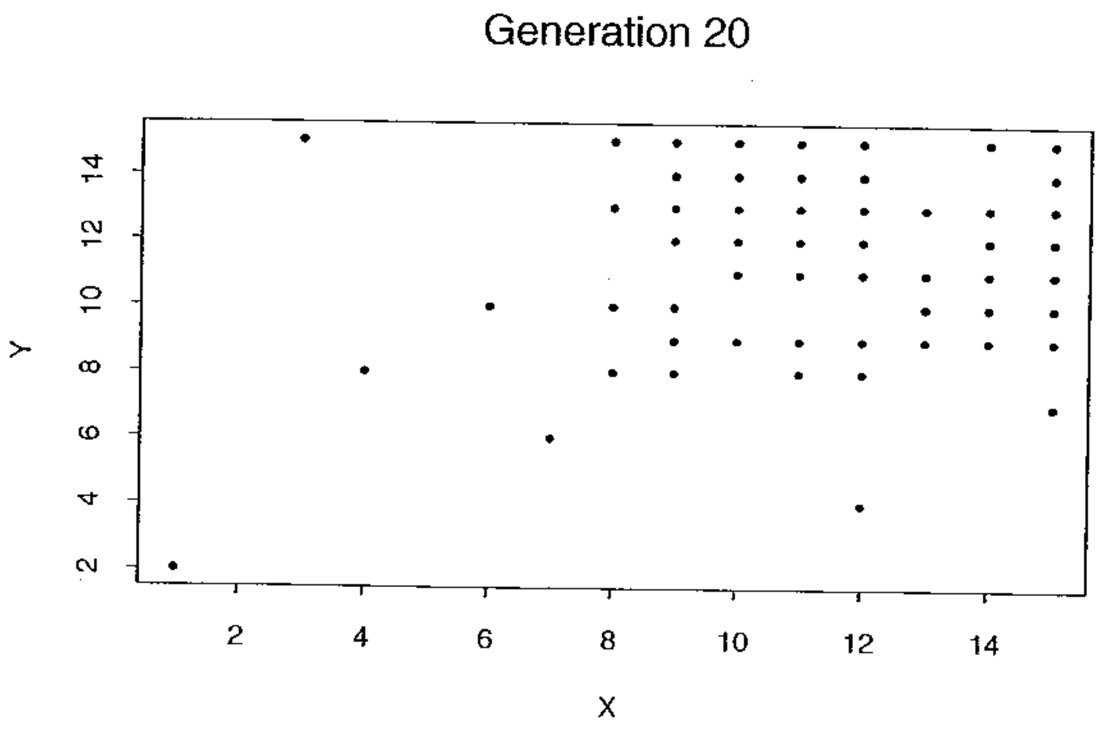


Figure 2: Scatter Plots of Chromosomes at Generation 0 and 10



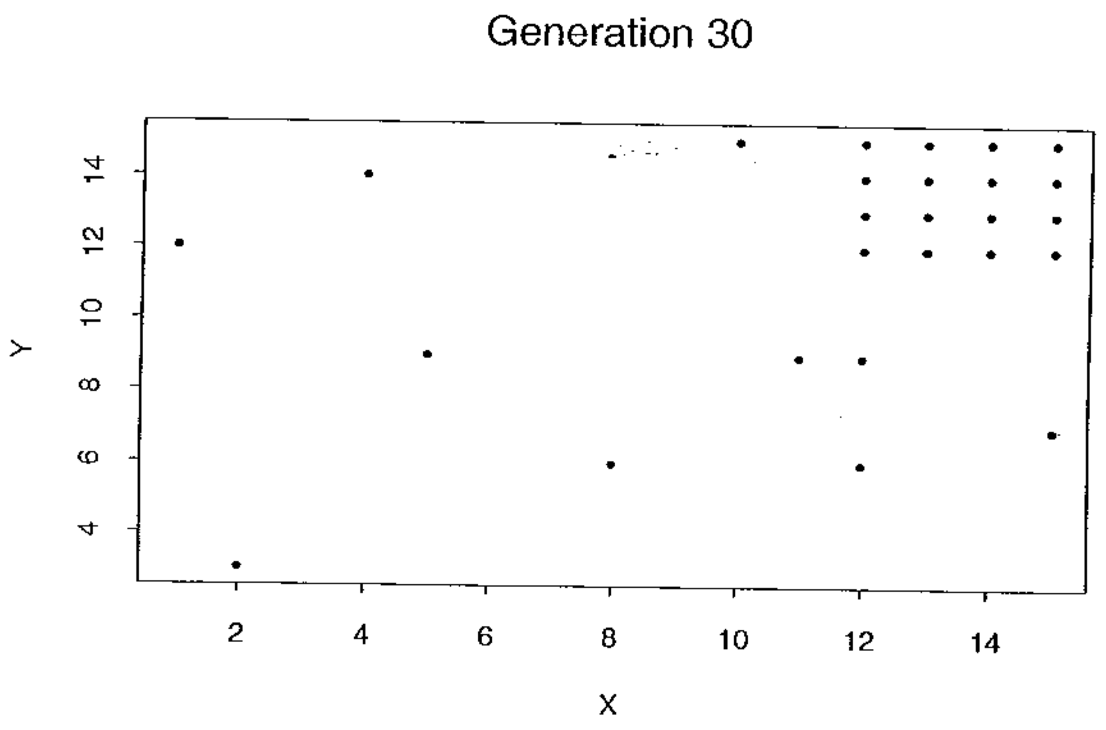


Figure 3: Scatter Plots of Chromosomes at Generation 20 and 30

Although this example was simple, it illustrates the basic ideas behind the SGA. The theory of the SGA is intensively developed in David Goldberg's book, *Genetic Algorithms in Search, Optimization, and Machine Learning.* The SGA has paved the way for a new breed of GAs. GA Researchers are studying different genetic representations, reproduction operators, evolutionary parameter settings and their corresponding effects on the performance of GAs. The GAs implemented and applied to regression problems in the remainder of this thesis are based on the components of the SGA.

CHAPTER 3

A GA Applied to Simple Linear Regression

The Problem Setting

In this chapter a GA is described and implemented to find the coefficients of a line that minimize the sum of the squared residuals. Althougth we know we can solve this problem analytically, the simple linear regression (SLR) setting will be an appropriate place to test the algorithm. The differences between the SGA and GA applied in this chapter will be discussed using a relatively simple problem.

Recall, that SLR implies having one independent variable, X, and one dependent variable, Y. In this setting, the general linear model, Y = X $\beta + \varepsilon$ takes on the following form:

Y is an n $\times$ 1 column vector of the observed responses.

X = [1 X] is a n $\times$ 2 design matrix made up of an n $\times$ 1 column vector of ones and an n $\times$ 1 column vector of the known, levels of the independent variable X.

β is a 2 $\times$ 1 column vector of unknown parameters.

ε is a n $\times$ 1 column vector of random errors.

The goal is to find the values of the unknown parameter vector, β, that minimize the sum of the squared residuals, or the SSE. The SSE is equal to (Y - Xβ)$'$(Y - Xβ). Linear models theory tells us that b = $(X'X)^{-1}X'$y minimizes the SSE and is the Best Linear Unbiased Estimator for β. Therefore, we know that if we find b, then we will have the coefficients for a regression line that has the smallest possible SSE for a particular data set. Now, let's think of this problem in terms of implementing a GA.

Genetic Representation

The first task in applying a GA to any problem is to construct what a chromosome will represent. For the SLR problem, $(a\ ,\ b)$ will represent the general form of a chromosome. Where a is the y-intercept and b is the slope, both of which are real numbers. That is, a chromosome will consist of two genes, the first gene is a proposed y-intercept and the second gene is a possible slope for the regression line. This type of genetic representation is referred to as real number encoded chromosomes. Using real number encoded chromosomes instead of binary ones is the first obvious distinction between this implementation of a GA and the SGA.

Although binary representation is used more by GA practitioners in general, real number representation is becoming more popular for a variety of reasons. Lawerence Davis of TICA Technologies, an authority in the field of GAs, has found that in practice real number encoded GAs have out-performed binary encoded GAs in numerical optimization problems. Numerical representation works effectively on mathematical optimization problems and allows for the use of numerical reproduction operators. In the SLR setting, numerical representation should make intuitive sense, because of the nature of the problem - estimating parameters of a line to minimize the SSE, all of which are real numbers. It should be clear that the choice of chromosome representation directly affects the development of the fitness or objective function and the reproduction process.

The Objective Function

The objective function is the function we wish to optimize. It has to be written in a way that uses the chosen representation of a chromosome from the population as input and then outputs the objective function evaluated at the parameters in that chromosome. In the SLR case, the SSE is to be minimized for a given data set. Realistically, dozens of possible objective functions come to mind that would accomplish this goal given our real number encoded chromosomes. The following objective function was implemented:

$$f(c_i) = \frac{100000}{(SSEc_i + .01)}$$

Where c_i is the ith chromosome and $SSEc_i$ is the sum of the squared residuals found by letting the y-intercept and slope of the model equal the y-intercept and slope in chromosome c_i. Thus a chromosome that yields a regression line with a low SSE, evaluates to a larger fitness or objective value. For the remainder of this discussion, F_i will denote the objective function evaluated at the ith chromosome. It is up to the selection and reproduction process to evolve the chromosome which in this case maximizes the objective function.

Selection Method

The method applied to select chromosomes to be used in the reproduction process is a commonly chosen technique referred to as roulette wheel parent selection. Imagine partitioning a roulette wheel into N slots, one for each chromosome in the population. The size of the slot i is proportional to F_i. The wheel is spun, and whichever slot the ball lands in is the selected chromosome. That is, a chromosome with a large F relative to the rest of the population will on average be selected more often for the opportunity to be used in the reproduction process. However, chromosomes with a relatively lower F still have a nonzero probability of being selected.

Allowing less than average chromosomes to be involved in the reproduction process aides in maintaning the diversity of the population. Without diversity, the objective function may be falsely optimized by a less than perfect chromosome because the place where the true optimum lies was not reached. Finding the balance between exploiting "good" chromosomes (ones with high F values) and exploring the entire search space is a job that is handled not only by the selection procedure, but also by the reproduction process.

Reproduction Process

The SGA used the one point crossover and mutation operator to alter selected binary encoded chromosomes to create new chromosomes for the next generation. Given that we are using real number chromosomes, new reproduction operators need to be defined. Nothing would be gained by applying the one point crossover to selected chromosomes in the SLR setting. Only the slope (the second gene) would have a chance of being altered by the one point crossover due to the fact that there is only one place to crossover. In his book, *Handbook of Genetic Algorithms*, Davis proposes several reproduction operators for real number chromosomes, three of which were applied in this GA.

We would like a crossover operator to combine two selected chromosomes to produce one which is potentially better than the original two. One possible real number crossover is what Davis refers to as the average crossover. The idea behind this operator is to take two selected parent chromosomes and average their corresponding genes to produce one new chromosome. In order to keep the population of chromosomes from converging too quickly and perhaps finding a less than optimal solution, an appropriate crossover rate has to be used and mutation operators need to be applied. One of the applied mutation operators is basically identical to the mutation operator used in the SGA. It changes a gene of a selected chromosome by replacing what is currently there, to a real number randomly selected from the appropriate range for that particular gene.

The third reproduction operator applied is also a type of mutation operator.

The real number creep operator takes a selected chromosome and adds a randomly selected amount from a uniform $(-\delta, \delta)$ distribution to the value in the first gene if a probability test is passed. The operator proceeds in a similar manner through all of the genes in the selected chromosome and thus has the potential to alter every gene. This reproduction operator can be thought of as fine tuning a possible solution to get closer to the global optimum.

The average crossover, mutation, and real number creep operators are the only ones used to create new chromosomes for the next generation in this GA. However, what percentage of the time should each operator be used in creating new chromosomes? In order to answer this question, the following experiment was done. The GA was allowed to run until the best chromosome, the one that yields the smallest SSE, had a fitness within 5 percent of the known maximum fitness (i.e., the fitness function evaluated with the least squares estimates). The response that was recorded was the number of generations it took to evolve an acceptable solution. This was carried out for different proportional uses of the three reproduction operators on the same data set. Table 1 contains nine settings of the reproduction operators. The setting that had the lowest mean number of generations and variance was setting E. Figure 4 displays the distribution of the number of generations, N, in the form of a boxplot for each setting in Table 1. Setting E had a mean of 22.4 generations and a standard deviation of 14.85. Increasing the average crossover rate beyond .40, or decreasing it below .30, caused the number of generations to increase along with the variability. Applying the crossover operator too much pulls the majority of the population towards a chromosome that is better than average, but not necessarily close enough to the global optimum. Not using the average crossover operator enough, turns a GA into a strictly random search which slows down the optimization process. Although other combinations of reproduction operator rates were examined, Figure 4 displays where the best settings were found. Keep in mind, that setting E was found to be the best in this problem realm, these are not necessarily the most robust reproduction operator settings for any function optimization problem.

Table 1: Settings of the Reproduction Operators

Setting	Crossover	Mutation	Creep
A	.40	.05	.55
B	.40	.10	.50
C	.40	.15	.45
D	.30	.05	.65
E	.30	.10	.60
F	.30	.15	.55
G	.20	.05	.75
H	.20	.10	.70
I	.20	.15	.65

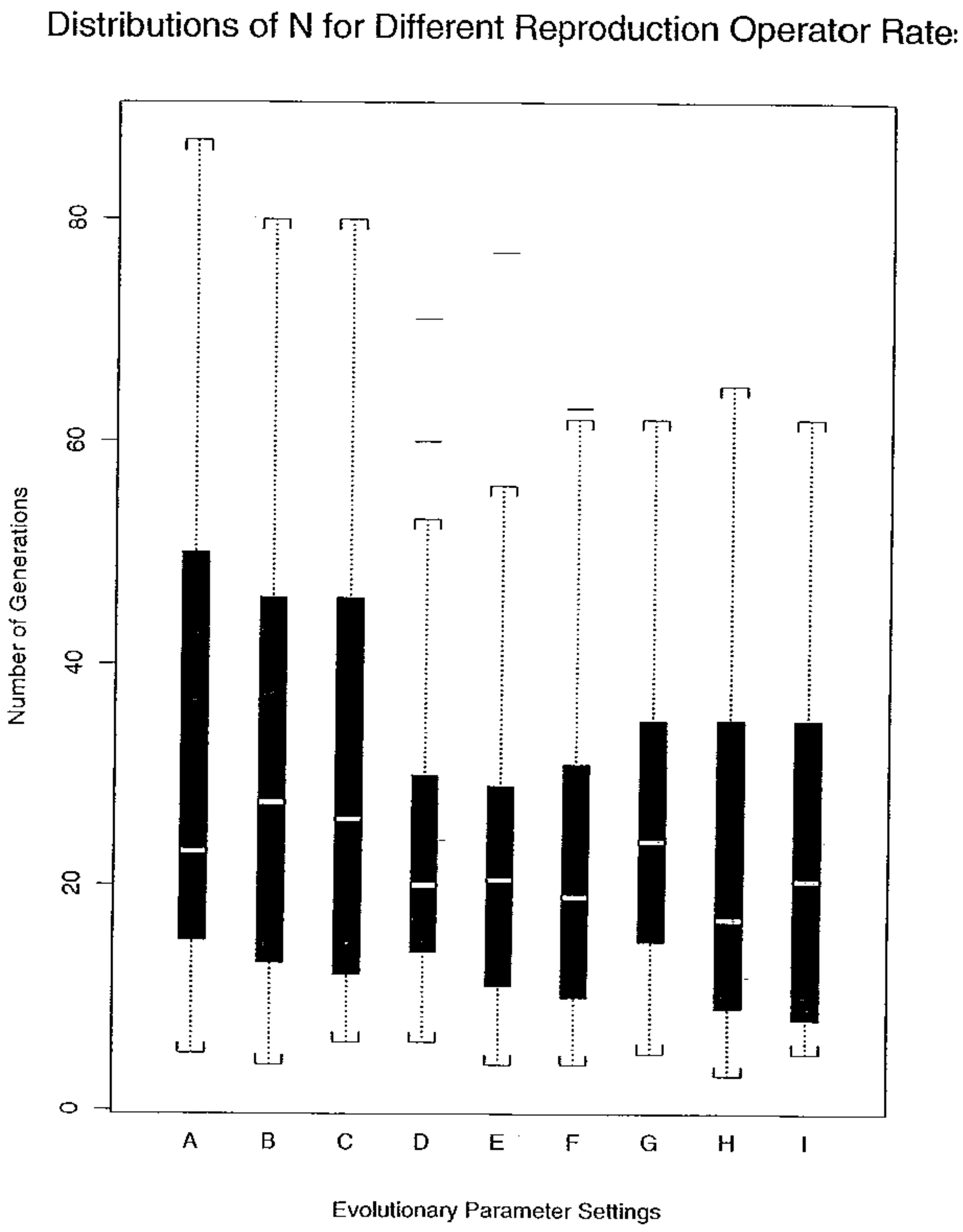


Figure 4: Boxplots

Example 1

Table 2: Example 1 data

X	Y
5	200
10	215
15	227
20	248
25	261
30	279

The GA was applied to the data shown in table 2. The least squares solution for this data set was found to be, $b' = (182.933\ 3.16531)$ and have a $SSE = 18.818$. Therefore, we know the best chromosome should be (182.933 3.16531) with

$$F_{max} = \frac{100000}{(18.819 + .01)} = 5310.956503.$$

Figure 5 displays a scatter plot of the data with the least square regression line. The GA begins by randomly generating 100 chromosomes where a gene's value comes from a uniform(0,200) distribution. It then calculates each chromosome evaluated in the objective function so the selection scheme can begin. Selection takes place and the three different reproduction operators are applied to produce 99 new chromosomes for the next generation. The 100th chromosome for the new population (i.e., the next generation) is the most fit chromosome from the previous generation. Keeping the best chromosome from the previous generation for the next one is a process referred to as elitism. Elitism guarantees that the best solution found thus far will be in the next generation to help the search and not be lost in the selection process. The objective function is evaluated at each chromosome in the new population, the new

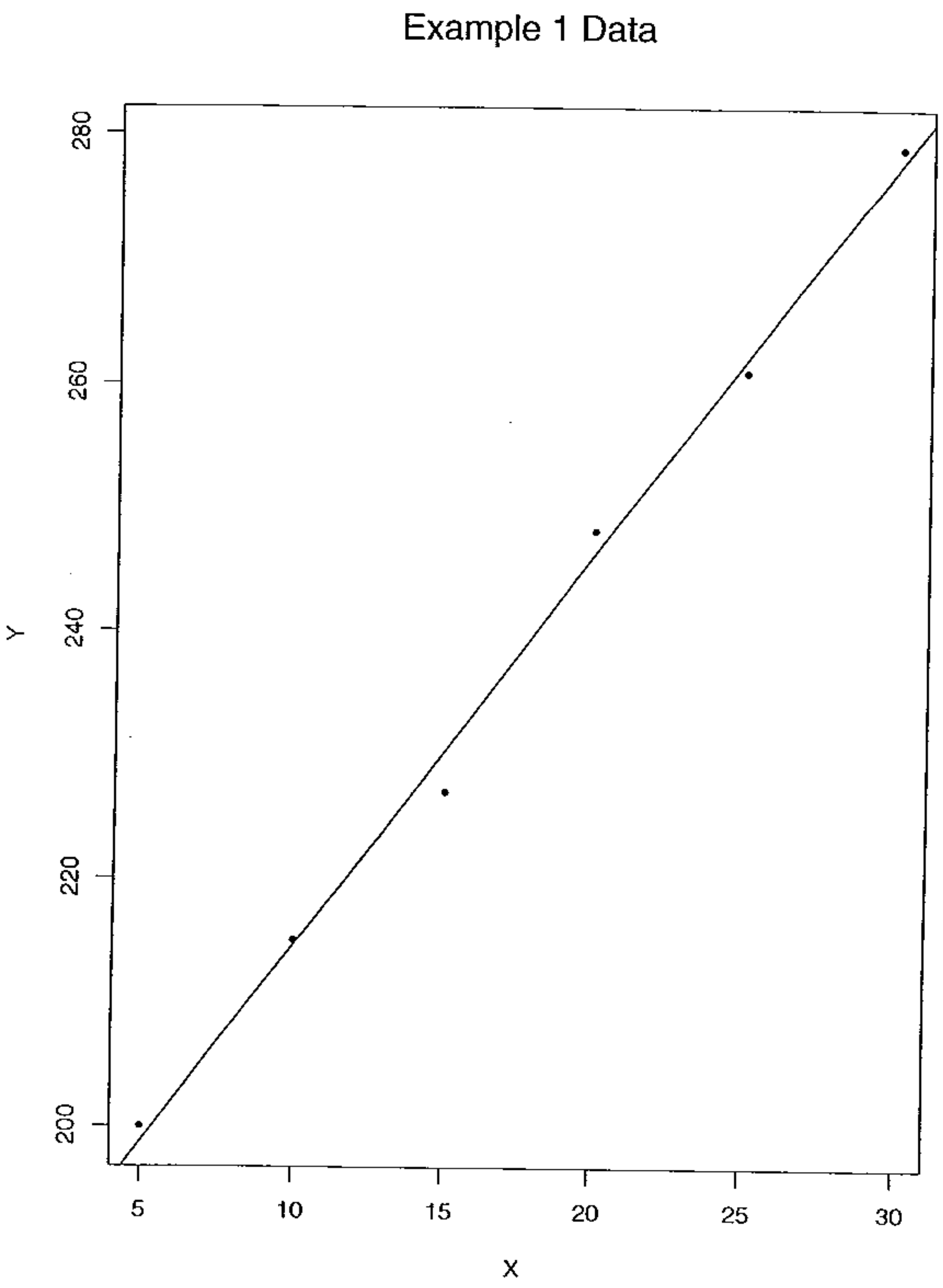


Figure 5: Scatter Plot of Example 1 Data

population becomes the old population, and the process starts over. The algorithm runs for sixty generations and the chromosome with the largest F, the solution, is reported. Due to the stochastic nature of a GA, the solution will vary from run to run. However, the applied GA evolves a solution close to the least squares estimates of the parameter β.

Since the parameter space is in $\Re^2$, we can view the objective function surface over the parameter space. Figure 6 is a plot of the interesting part of the objective function surface, that is, where the optimum lies. Most of the surface is relatively flat and the GA needs to search for the hill by using the mutation operators to "jump" around the surface. Once a "super" chromosome is evolved, the selection process and the average crossover operator pull some of the population towards the hill. The real number creep operator takes chromosomes that are on the hill and has the potential to create chromosomes that will climb towards the top. However, in order to make sure that there is not a larger hill being overlooked, the mutation operators keep some of the chromosomes exploring different regions of the search space. By the last generation, the majority of the chromosomes are concentrated near the top of the hill while others are still exploring other areas of the objective function surface. If the GA was applied again to the data in example 1, then a different solution would be evolved, unlike least squares where we would obviously get the same estimate of β. In the next section, an attempt is made to determine on average how close the GA solutions are to the least squares solution and the variablility of those solutions.

Expectation and Variability

The total variability of a solution found by the applied GA is the variablity in the data, plus the variability in the GA itself. In order to estimate the expected value and variability of the algorithm in the SLR setting a simulation was run. A model with

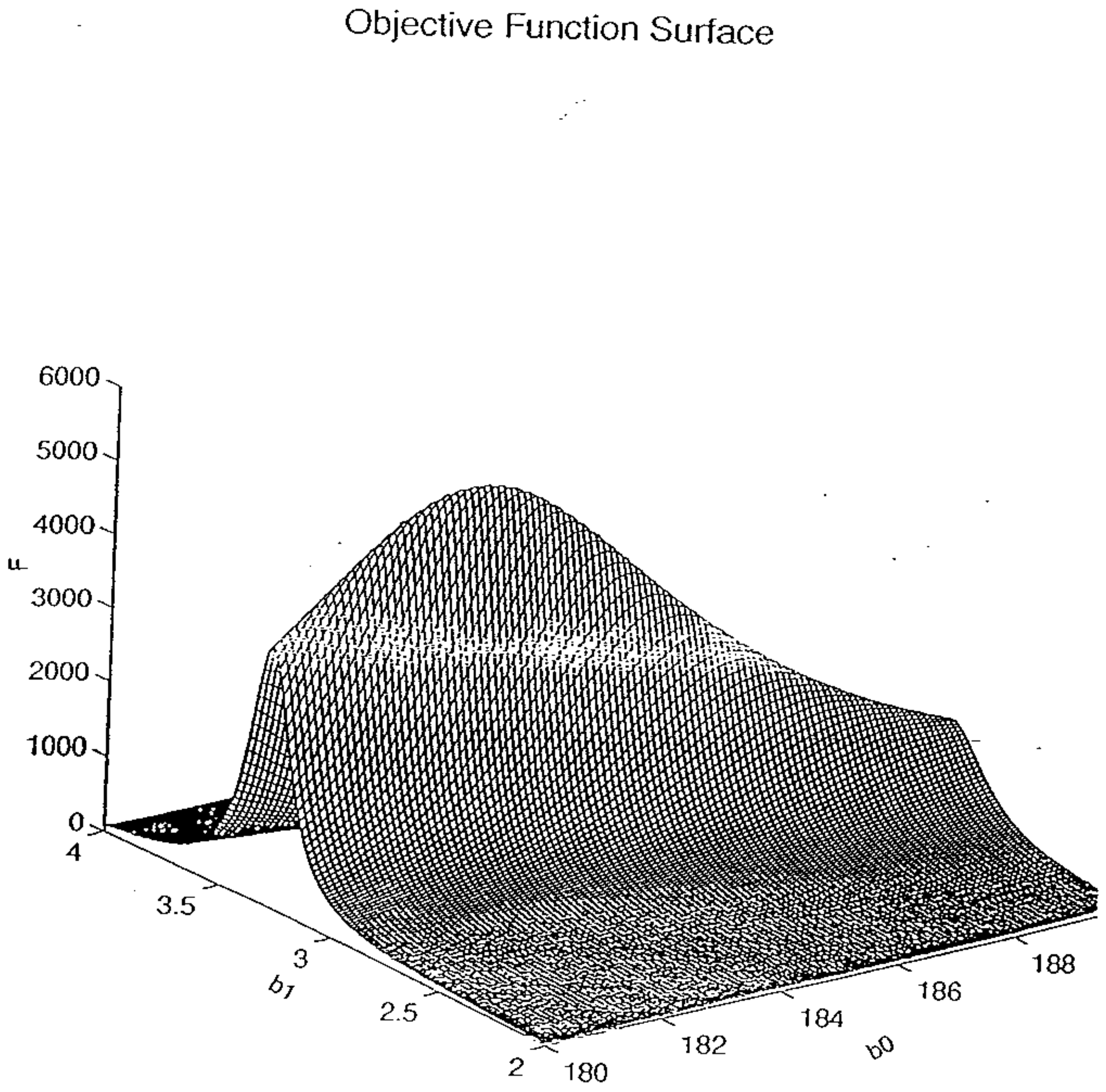


Figure 6: Objective Function Surface for the Data in Example 1

known parameters and error structure was chosen to be Y $= 133X + 33 + \varepsilon$, where ε iid N(0,1) distribution. The explanatory variable, X, was the first ten integers. Three hundred Y's were created from the model with three hundred different random error vectors. The GA was run on the three hundred different data sets for sixty generations and each solution was recorded.

Figure 7 displays a scatter plot of b_0 versus b_1 for three hundred runs. Figure 8 contains histograms of b_0 and b_1. The mean of b_0 and b_1 was found to be 33.1417 and 132.9784, respectively. Thus, the simulation results indicate that the algorithm is evolving toward unbiased estimators of b_0 and b_1, respectively. This seems reasonable when considering the optimal solution we are searching for (the least squares solution) is unbiased for β. Under least squares, the covariance matrix is,

$$\begin{bmatrix} .46667 & -.06667 \\ -.06667 & .01212 \end{bmatrix}$$

The variance of b_0 was equal to .560485, which is slighty more than the variance under least squares. The variance of b_1 was equal to .014464. Although the sample variances are greater than the variances of b under least squares, by looking at Figure 8, the normal distribution assumption still looks valid. This empirical evidence indicates that standard inferences can still be made, using inflated variances. In this case, the variance of b_0 and b_1 were inflated by 20 and 19 percent, respectively. If the inflated variances_are functions of the number of generations allowed for the search (in this case sixty), then asymptotically the GA variances approach the least squares variances.

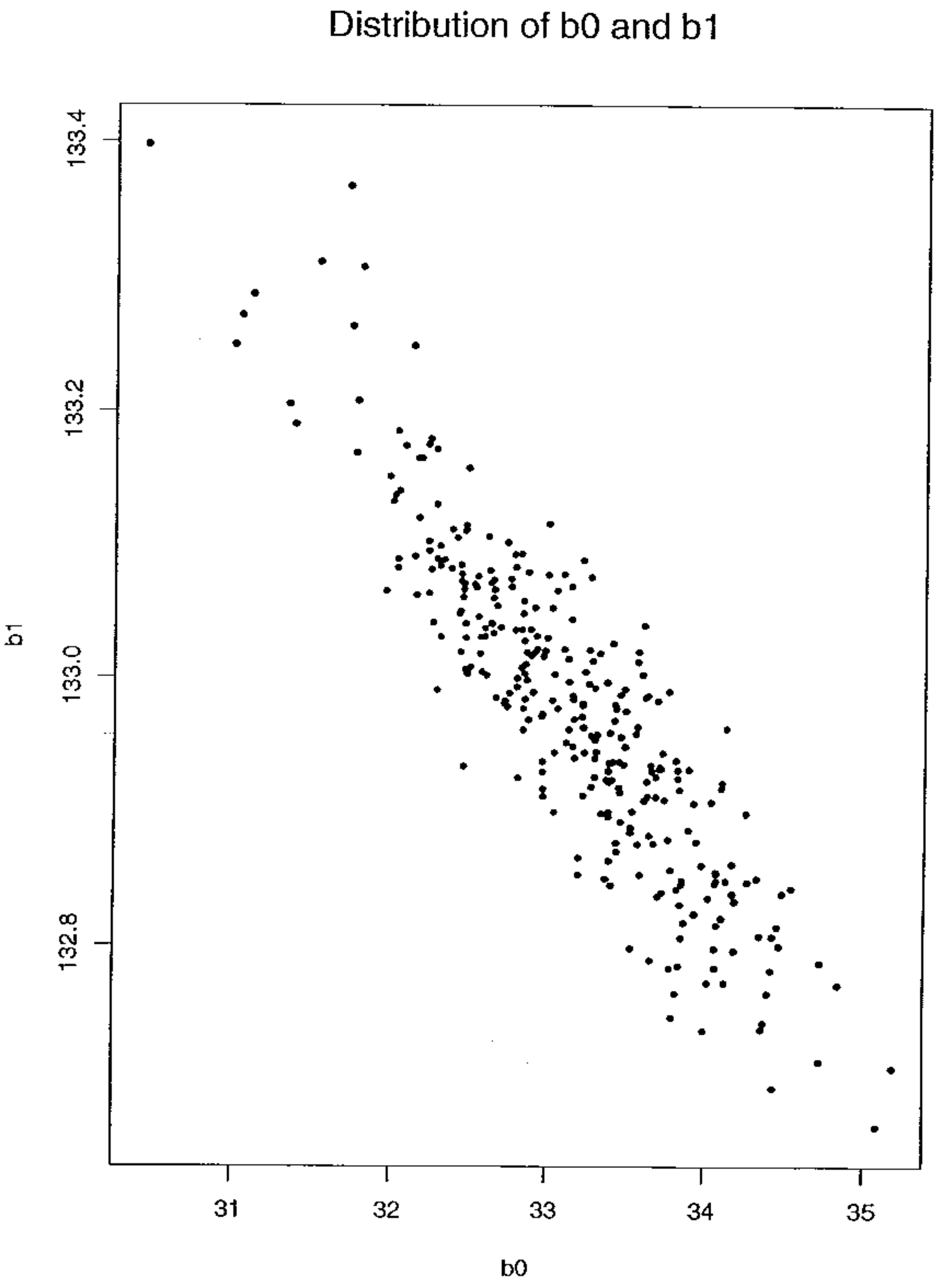
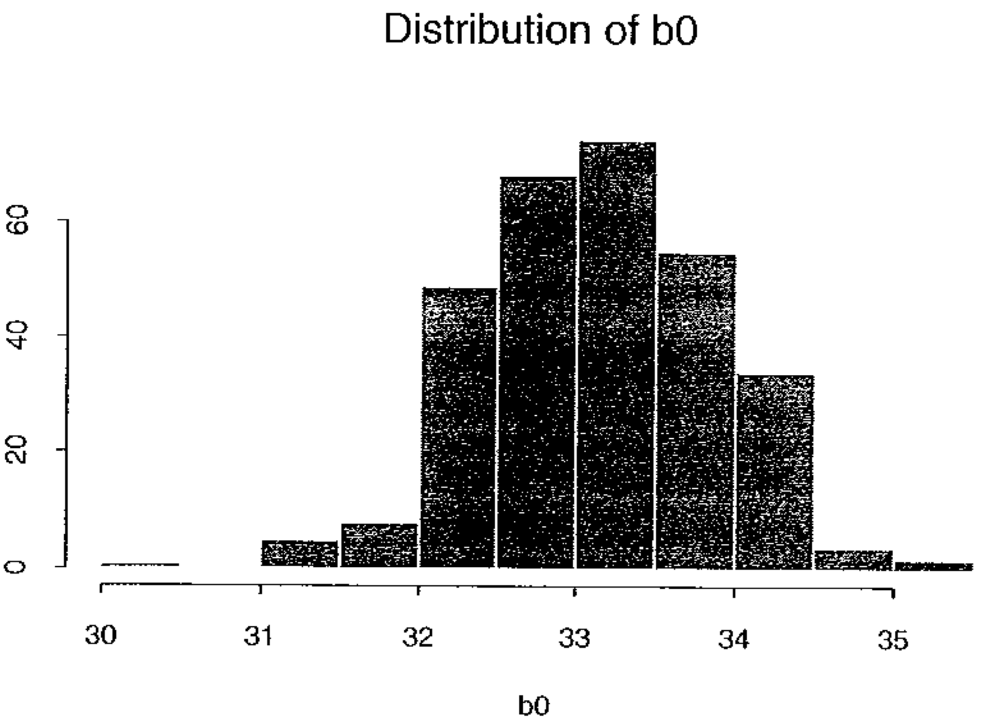


Figure 7: The Distributions of b_0 and b_1 From a Simulation



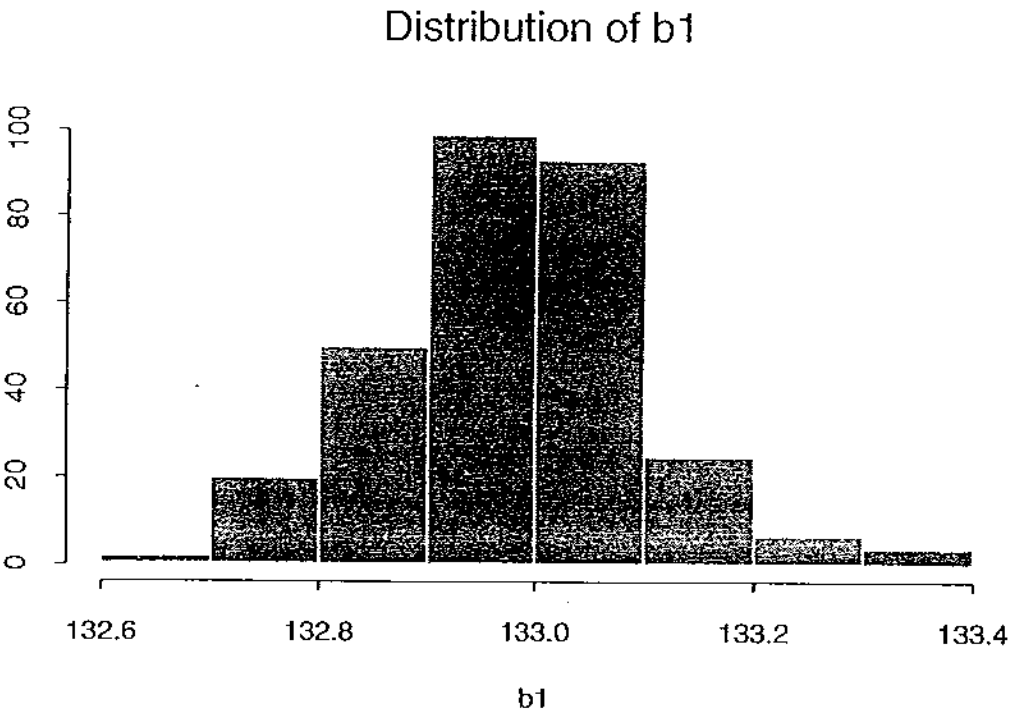


Figure 8: Histograms of b_0 and b_1 From a Simulation

In order to verify that the distributional and variances results were not artifacts of the previous model selected, Y = 133X + 33, another simulation was run. Models were selected in a way to create a solid support in $\Re^2$ and 100 different data sets were created from each model, again by adding random N(0,1) errors. The distributions of b_0 and b_1 generated by the GA for each model were approximately normal with means close to the true model parameters. The variances of b_0 and b_1 were inflated by an average of 15 and 20 percent of their corresponding variances under least squares. The results from the simulation indicate that the GA will perform effectively, regardless of the underlying SLR model. In other words, the performance of the GA is independent of the selected SLR model.

CHAPTER 4

A GA Applied To Nonlinear Regression

The Problem Setting

In this chapter, a GA is applied to finding the unknown parameters in a nonlinear model such that the SSE is minimized for a given data set. The spherical model, which is a commonly used model to fit variograms, was selected as a nonlinear model to test the algorithm. This model is difficult for many techniques to fit to data because it is nondifferentiable at particular points. Gradient based searches would have to be modified to numerically approximate the derivatives in an attempt to fit the model. Assuming no nugget effect and equally spaced data, the spherical model reduces to,

$$\gamma(h) = c_s(1.5 \times h/a_s - .5 \times (h/a_s)^3)$$

Where h is the lag, c_s is the sill, and a_s is the range. For relevant variogram information, refer to Noel Cressie's book, *Statistics for Spatial Data* 1991.

Changes in Genetic Algorithm

A nice characteristic of a GA is it can be changed to solve a different optimization problem without much effort. We will still be using real number encoded chromosomes with two genes, one for each parameter in the model. However, gene one will be equal to a possible sill and gene two will represent the range. The objective function has the same form as in the SLR regression setting, except SSE_c is now the SSE found by using the spherical model. With the representation and objective function defined, the selection and reproduction process can take place. Roulette wheel parent selection will again be the chosen selection technique. Average crossover, mutation, and the real number creep will be the operators applied in the reproduction process. Now that the main components of the GA are defined, lets apply the GA to the data in example 2.

Example 2

Assuming a spherical model with known parameters, we can see if the GA is evolving acceptable solutions. The data in example 2 came from a spherical model with $c_S = 9$ and $a_s = 10$, plus some white noise.

Table 3: Example 2 data

h	$\gamma(h)$
1	0.8392
2	2.8045
3	4.9857
4	3.7586
5	5.6109
6	6.5367
7	7.1301
8	8.1980
9	7.1317
10	8.9134

On average, the GA evolves the chromosome (8.32 10.12), which is extremely close to the parameter estimates using a Gauss-Newton based procedure. This preliminary test indicates that the GA is working rather well. However, the data in example 2 was not generated in a fashion that we actually believe to be driving a

one dimensional time series process. A more legitimate simulation was done in the following manner. Assuming second order stationarity and the spherical model with known parameters, a known covariance matrix was developed.
A one dimensional time series having that covariance structure was randomly generated. From the time series, the estimates of $\gamma(h)$ were found in their usual manner.

Example 3

The $\widehat{\gamma}(h)$'s in Table 4 were calculated from randomly generated observations with a known covariance structure and a spherical model with $c_s = 5$ and $a_s = 10$. The parameter estimates using a Gauss-Newton technique to minimize the SSE for this data set were found to be $c_s = 4.642$ and $a_s = 20.654$. On average the GA evolved estimates of $c_s = 4.634$ and $a_s = 20.691$. Similar results were obtained by applying the GA to other data sets from other spherical models. The GA can be used as an alternative optimization tool to fit nonlinear models instead of a Gauss-Newton based technique.

Why would you use a GA instead of a Gauss-Newton based algorithm to fit a nonlinear model? Because, sometimes, a Gauss-Newton based algorithm fails to converge or yields incorrect parameter estimates. To use a Gauss-Newton algorithm, good starting points for the estimates of the parameters need to be supplied. If the starting point is not close enough to the global optimum, then invalid parameter estimates may be found or possibly none will be found due to divergence. Admittedly, the GA needs to have an allowable range supplied for each parameter. However, that range can be extremely large relative to ranges or starting points needed in other search algorithms and still evolve acceptable solutions.

An additional ability of a GA over other optimization strategies is that it has the capability to not only evolve good parameter estimates, but also evolve the "best"

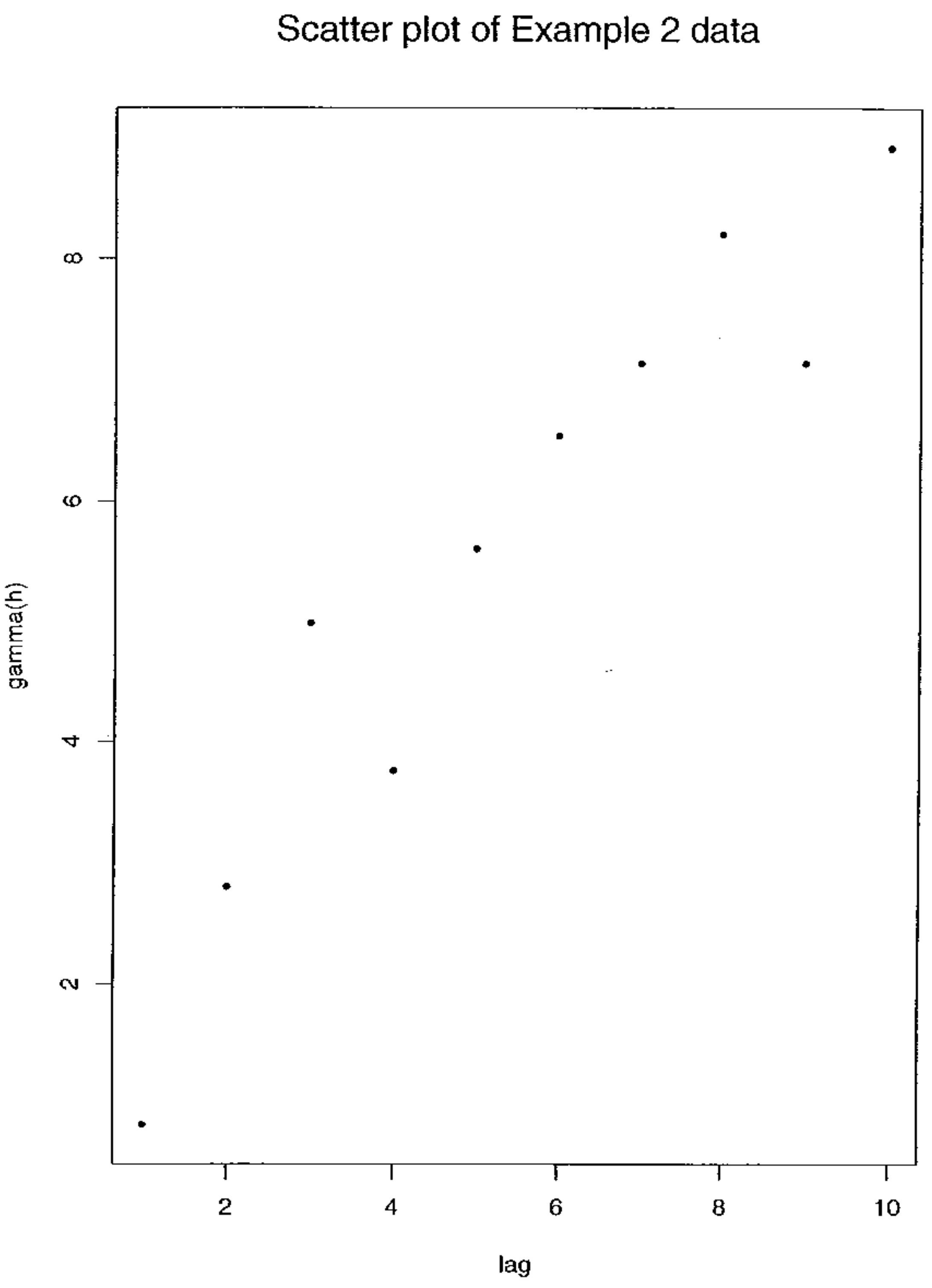


Figure 9: Scatter Plot of Example 2 Data

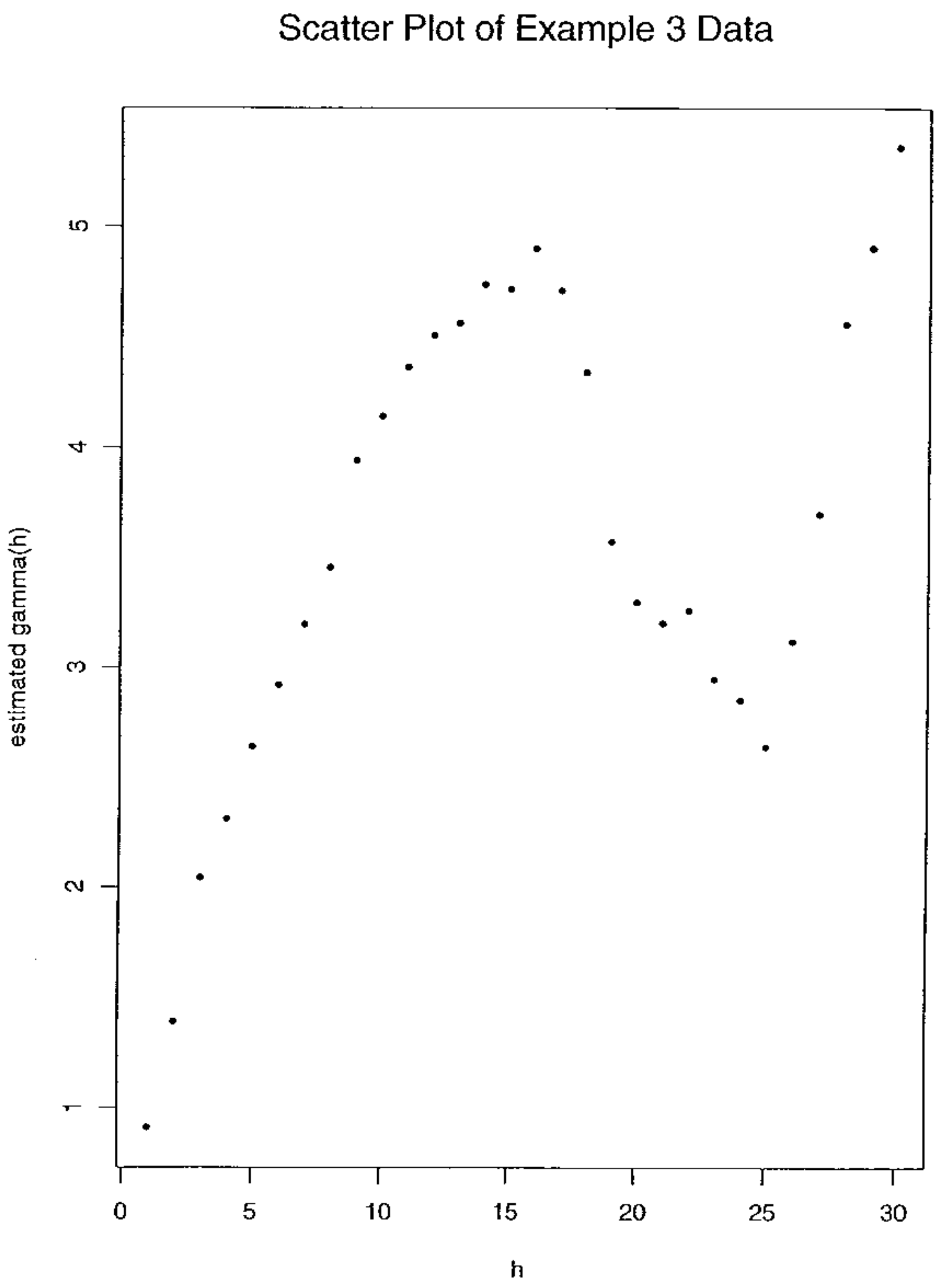


Figure 10: Scatter Plot of Example 3 Data

model, where "best" could be related to any desired criteria, such as minimizing the SSE. Example 4 discusses this interesting variation on this potential use of a GA.

Example 4

Suppose we are trying to decide wether to fit a spherical model or an exponential model to a particular data set of $\hat{\gamma}(h)$'s such that the SSE is minimized. The exponential model used to fit variograms, assuming equally spaced data and no nugget effect becomes,

$$\gamma(h) = c_e(1 - exp(-h/a_e))$$

Where h is the lag, c_e is the sill, and a_e is the range.

Chromosome representation thus far has been only to let genes take on specific parameters in a given model. For this problem, a chromosome was defined to be, (m s r). Where m will identify the desired model and take on any real number between 0 and 1, s and r are the sill and range for that model. If m is less than or equal to .5, then the spherical model will be used to obtain the fitness for that chromosome. Or, if m is greater than .5, then the exponential model will be used. The objective function was changed to facilitate the use of the new chromosome. It still takes a chromosome for input, but now outputs the F associated with evaluating the chromosome's sill and range in the model specified by the first gene. The last change in the GA was in the initialization process. The initial population of chromosomes is randomly generated having the new (m s r) structure.

It may seem that the reproduction operators need to be changed to handle the new chromosome structure. However, the point of allowing m to take on any real number between 0 and 1 was so the reproduction operators do not need to be changed. The average crossover, mutation, and the real number creep operators will work with the newly defined chromosome. The new GA was applied to finding the best model and parameters for that model that minimize the SSE for the data shown in Table 5.

The data in Table 5 is from evaluating a spherical model with a sill = 9 and a range = 10, at the first ten integers. A spherical model was fit to the data using a Gauss-Newton technique which yielded a sill = 9.028845 and a range = 10.10603 with a $SSE = .3622863$. A sill = 13.63256 and range = 8.525267 was found by fitting an exponential model to the data again using a Gauss-Newton technique. The new GA was applied to the data and evolved the chromosome (.109 9.016399 10.07732) with a $F = 268.3853$ ($SSE = .3726125$). In other words, the best chromosome states to use the spherical model (m less than .5) with a sill = 9.016399 and a range = 10.07732 to minimize the SSE. The majority of the chromosomes in the population at the final generation (the GA was allowed to run for 50 generations) had a number less than .5 for the first gene. This implied that the spherical model was the better choice. Although, there were still chromosomes exploring parameter settings for the exponential model.

Of course, it would be possible to fit the spherical model and exponential models separately and compare the resulting SSE. However, the purpose of the example was to illustrate another way of using a GA. Also, this example led to potentially beneficial new ways of applying a GA to difficult problems which will be described in the next chapter.

Table 4: Example 3 data

h	$\hat{\gamma}(h)$
1	0.9084
2	1.3839
3	2.0408
4	2.3124
5	2.6386
6	2.9182
7	3.1931
8	3.4518
9	3.9404
10	4.1423
11	4.3676
12	4.5118
13	4.5671
14	4.7415
15	4.7198
16	4.9023
17	4.7143
18	4.3426
19	3.5684
20	3.2947
21	3.1985
22	3.2555
23	2.9462
24	2.8491
25	2.6386
26	3.1159
27	3.6943
28	4.5624
29	4.9074
30	5.3574

Table 5: Example 4 data

h	$\gamma(h)$
1	1.3455
2	2.6640
3	3.9285
4	5.1120
5	6.1875
6	7.1280
7	7.9065
8	8.4960
9	8.8695
10	9.000

CHAPTER 5

Conclusions

Future Work

Example 4 illustrated the idea of using a GA to select a model and the parameters for that model that minimize the SSE for a particular data set. That idea led to an inherently harder problem of trying to fit a mixture model to a data set. For example, say we are trying to determine wether to fit a spherical, exponential, or linear combination of the two models to a specific data set. What is commonly done in practice is to fit one of the models and ignore the mixture model due to the complexity in finding the appropriate parameters. However, the GA applied in this thesis could be modified to solve this problem. For example, the model would become,

$$\gamma(h) = \alpha Spherical(s1, r1) + (1 - \alpha)Exponential(s2, r2)$$

Where α is a real number between 0 and 1. A chromosome would take the form (α $s1$ $r1$ $s2$ $r2$). The GA would evolve the best α, sill and range for the spherical and exponential models to minimize the SSE. If α were close to zero, then fit just the exponential model with sill = $s2$ and range = $r2$. On the other hand, if α were close to one, then fit the spherical model with the evolved sill and range. Otherwise, use the evolved linear combinations of the two models.

There were two GAs written for this thesis, one in C and the other in S-Plus. The S-Plus code is slow and can be improved. One future goal is to write a GA library for S-Plus. The researcher will be able to use different settings of the evolutionary parameters, different reproduction operators, and specific objective functions, and run a GA from within the S-Plus environment.

Conclusions

A GA is based on a relatively basic idea, that of using simplified theories of evolution to search a space. The key is to encode possible solutions to a problem in the form of chromosomes. Select the solutions that solve the problem better than the others and allow reproduction operators to change them to form potentially better solutions. GAs search a space in parallel for a global optimum. Using the selection and reproduction processes a GA avoids local extrema and steers toward the global optimum whereas many other search strategies can be easily fooled by local extrema. GAs do not need to have the luxury of continuous spaces and/or existence of derivatives to be effective. However, this is not to say that they could not take advantage of further information about the search space if it is available. This thesis discussed the components of a GA that was used to fit linear and nonlinear models. The applied GA can be summarized as follows:

Real number encoded chromosomes

Generational replacement with elitism

Roulette wheel parent selection

Average crossover

Mutation

Real number creep

The GA successfully evolved parameters for linear and nonlinear models to minimize the SSE. In the realm of fitting variograms, the GA always converged on a solution that was acceptable at minimizing the SSE, while a modified Gauss-Newton technique occasionally failed to converge. The idea of converging on near optimal solutions when other search strategies fail is one obvious advantage of using a GA over other search strategies. A GA is an extremely versatile search strategy. A GA found the parameters for a nonlinear model to minimize the SSE and without much effort, was changed to evolve a model, and the parameters for that model, that minimize the SSE.

Although minimizing the SSE was the criterion used for selecting the "best" model, other criteria could have been successfully implemented. For example, the sum of the absolute residuals could have been used if robustness to outliers was a concern. Any quantitative criteria could have been used and written to be the objective or fitness function for the GA. The GA's simplicity, versatility, and power make it an excellent search strategy that statisticians can and should take advantage of.

BIBLIOGRAPHY

[1] T. Back, U. Hammel. Evolution Strategies Applied to Perturbed Objective Functions. *Proc. of the 1st IEEE Conference on Evolutionary Computation.* (pp. 40-5), IEEE Service Center, Piscataway, NJ, June, 1994.

[2] N. Cressie. *Statistics for Spatial Data.* John Wiley and Sons, New York, NY, 1991.

[3] L. Davis. *Handbook of Genetic Algorithms.* Van Nostrand, Reinhold, NY, 1991.

[4] K. A. De Jong. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems.* PhD Thesis, University of Michigan, 1975.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley Publishing Company, Reading, MA, 1989.

[6] D. E. Goldberg, J. Richardson. Genetic Algorithms With Sharing Multimodal Function Optimization. *Proc. of the 2nd International Conf. on Genetic Algorithms* (pp. 41-9), Lawrence Erlbaum Assoc., Hillsdale, NJ, 1987.

[7] J. J. Grefenstette. Optimization of Control Paramters for Genetic Algorithms. *IEEE Transactions on Systems, Man and Cybernetics.* SMC-16(1):122-128, 1986.

[8] J. H. Holland. *Adaptation in Natural and Artificial Systems.* Univ. of Michigan Press, Ann Arbor, 1975

[9] E. H. Isaaks, R. M. Srivastava. *An Introduction to Applied Geostatistics.* Oxford Univ. Press, New York NY, 1989.

[10] Z. Michalewicz *Genetic Algorithms + Data Structures = Evolution Programs.* Springer Verlag, 1992.

[11] R. H. Myers *Classical and Modern Regression with Applications.* Duxbury Press, Belmont CA, 1990.

APPENDIX

Help Guide

This appendix contains a brief description on how to use the GA within the S-Plus environment. The GA is written in one function called "GA". This function contains the population initialization, selection, and reproduction processes. A call to the GA function takes on the following form:

$$answer < -GA(N, T, h, gh)$$

Where N is the desired number of chromosomes in the population, T is the maximum number of generations until termination, h is a vector of length n containing the lags, and gh is a vector of length n of $\hat{\gamma}(h)$'s. The GA function returns the best chromosome found after T generations of searching. Thus, *answer* would contain the sill and the range that minimize the SSE for fitting say, the spherical model, to the data stored in h and gh.

The GA function calls another function called CalcFit, for calculate fitness. The CalcFit function uses the population of chromosomes for input and returns a vector containing the value of each chromosome in the population evaluated in the objective function. Recall, the objective function used in this thesis was,

$$f(c_i) = \frac{100000}{(SSEc_i + .01)}$$

Where c_i is the ith chromosome in the population and $SSEc_i$ is the SSE found by fitting the spherical model with the sill and range in chromosome c_i to the data (h and gh). Thus, CalcFit returns a length n vector containing the fitness values. Currently, CalcFit fits the spherical model, but it can be altered to be any desired model.

Say we wanted to find the sill and range for the spherical model that minimize the SSE for the data in example 2 using the GA function. Once in S-Plus create the vector h to be the first ten integers and gh to be 10 * 1 vector of the $\gamma(h)$'s shown in Table 3. Then type,

$$answer < -GA(100, 50, h, gh)$$

which will run the GA with a population of 100 chromosomes for 50 generations and store the best solution found during the search in the 1 * 2 vector answer. For this example, *answer* would contain the range and sill that minimize the SSE.