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.

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

N378
P1915

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.

Apr 21, 1995	Jeffrey D. Banfield
Date	Jeffrey D. Banfield Chairperson, Graduate Committee

Approved for the Major Department

April 21, 1995	John Lund
Date	John Lund Head, Mathematical Sciences

Approved for the College of Graduate Studies

5/4/95	[illegible]
Date	Robert Brown Graduate Dean

STATEMENT OF PERMISSION TO USE

Signature [illegible]

Date 4/21/95

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

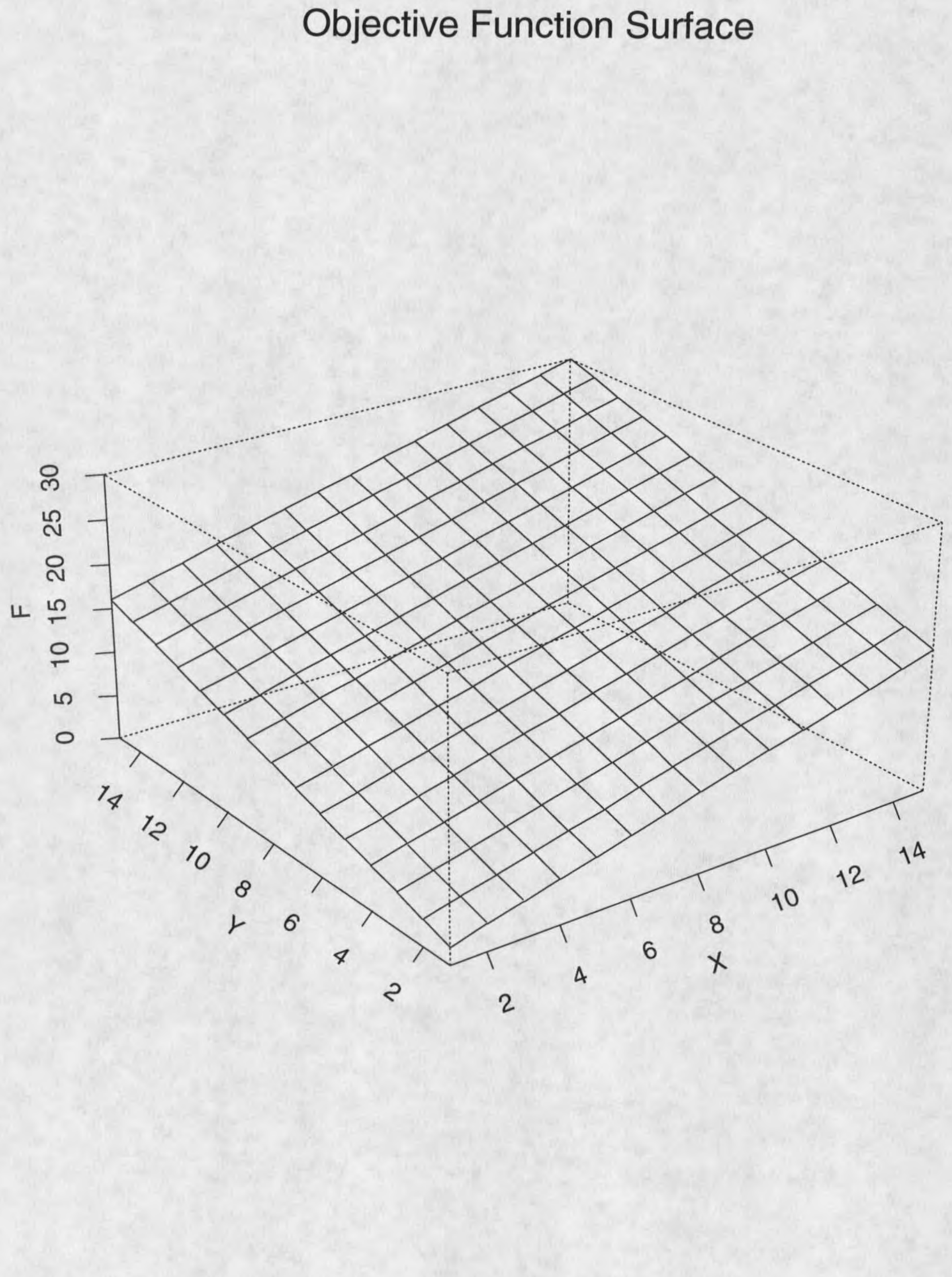


Figure 1: Objective Function Surface for F(x,y) = X + Y

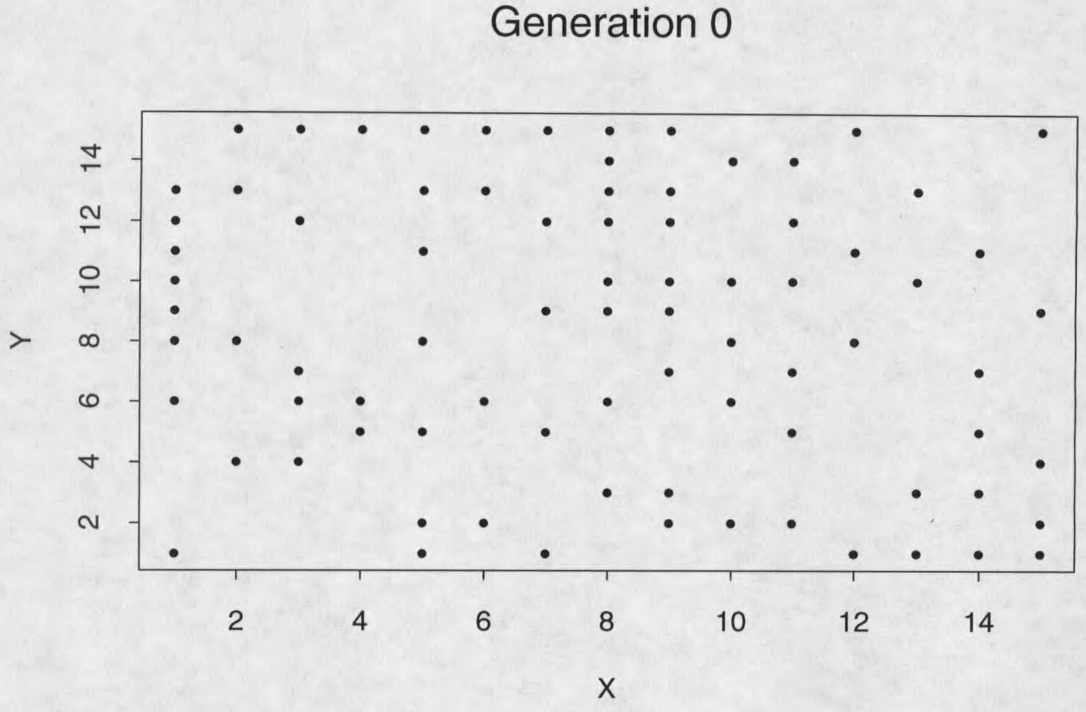
they can pass on their good traits to future chromosomes. Likewise, we do not want to select chromosomes for the reproduction that seem to be performing poorly. A common selection technique is to select chromosomes from the population proportional to their fitness. Thus, chromosomes with larger fitness 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.



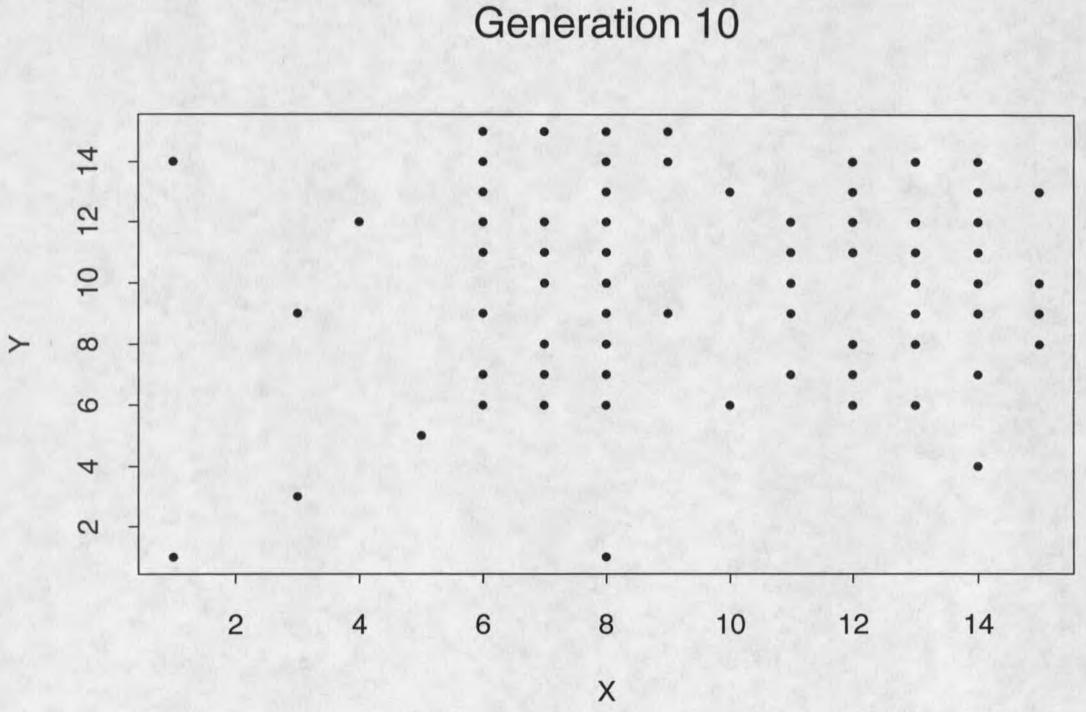
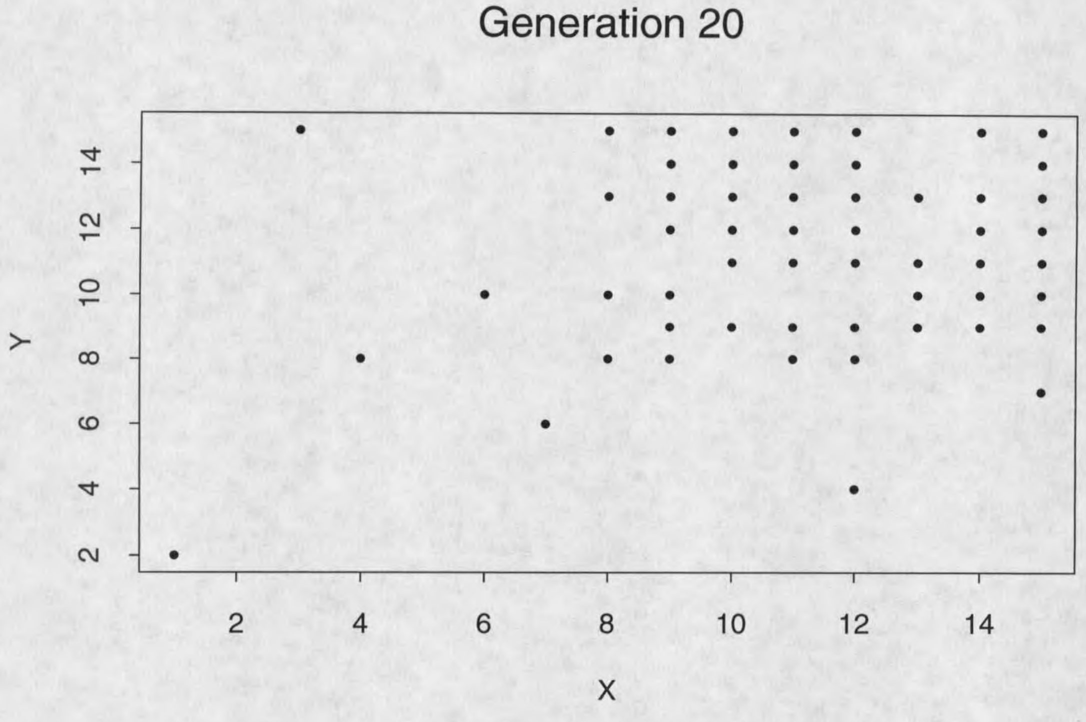


Figure 2: Scatter Plots of Chromosomes at Generation 0 and 10



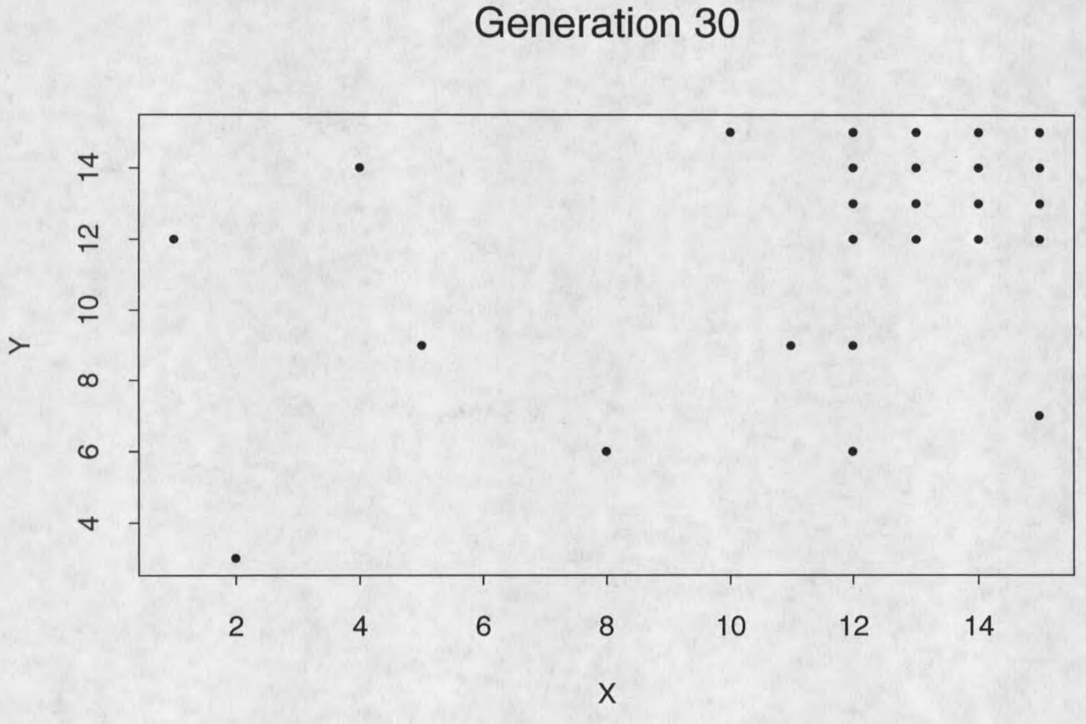


Figure 3: Scatter Plots of Chromosomes at Generation 20 and 30

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, $\mathbf{Y} = \mathrm{X}\,\beta + \varepsilon$ 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 $(\mathbf{Y} - \mathrm{X}\beta)'(\mathbf{Y} - \mathrm{X}\beta)$. Linear models theory tells us that $\mathbf{b} = (X'X)^{-1}X'\mathbf{y}$ 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