A Genetic Algorithm for Optimising Inter-Planetary Trajectories

Stephen Richardson

MSc Information Technology: Knowledge Based Systems

Department of Artificial Intelligence

University of Edinburgh

1995

Abstract

The design of an interplanetary trajectory for a space probe, such as that for the Galileo mission to Jupiter, is a complicated and time consuming process. Initial work by Wilson ([Wilson 93]) suggested that the group of computational models known as genetic algorithms could be applied to this type of problem with some success.

It was the aim of this project to implement a genetic algorithm to evolve the set of course corrections necessary to pilot a simulated space probe through a 2-D solar system. The solution was not only required to take the probe to a specific destination planet but also to do so with an efficient use of fuel.

Several variations on the basic genetic algorithm model were tested in this project with a combination of steady state control strategy and one point crossover being found to be the most effective. Promising results were obtained from this variation on a problem of similar complexity to the Galileo mission. Using three small sub-populations of 25 chromosomes the algorithm is able to evolve an effective trajectory in only 30 generations. Using this trajectory the probe not only reached its target but also appears to do so in a relatively efficient manner.

Although limited in certain ways, the results from this project were enough to give good encouragement that genetic algorithms do have good potential for this type of trajectory optimisation problem.

Acknowledgements

I would first of all like to thank my supervisors, Dr Peter Ross and Dave Corne, for their support and encouragement during the course of this project.

Thanks to all my fellow MSc students for making this year in Edinburgh such an enjoyable one. In particular, for this project, thanks must go to Mike Fairbank for his advice on orbital mechanics and proof reading and to Peter Murphy for his help with the Postscript output routines of my program interface.

Thanks must also go to Dominic Giamapolo, regardless of the fact that I've never met him his Libsx, X library (GNU copyright) has helped enormously in preventing my project being overrun by X-windows programming.

Finally, I would like to dedicate this thesis to my parents, for without their support and encouragement none of this would have been possible.

Notation

GA : Genetic algorithm

p_m = probability of mutation

$\mathbf{r}$ = vector between Sun and a planet

r = orbital radius (magnitude of $\mathbf{r}$)

a = major axis of an elliptical orbit

b = minor axis of an elliptical orbit

p = semilatus rectum of an elliptical orbit

e = eccentricity of an elliptical orbit

T = orbit period

$\frac{d^2x}{dt^2}$ = acceleration in the x direction

$\frac{d^2y}{dt^2}$ = acceleration in the y direction

M_p = mass of probe

M_i = mass of planet i

M_s = mass of the Sun

G = universal constant of gravitation

t = time

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

On October the 18th 1989 the Space Shuttle Atlantis blasted off carrying with it the Galileo spacecraft on the first leg of its six year journey to Jupiter: a journey which has taken it to Venus and back out to shoot past the Earth again on its way out to Jupiter which it is currently approaching at a speed of 25,300 kilometres per hour (relative to the sun).

The probe is due to skim 1000km above the moon of Io on the 7th of December this year and using both the gravitational pull of the moon and its own engines will place itself into an orbit around Jupiter. Galileo will then spend two years completing an eleven orbit track around this massive planet during which it will study Jupiter and its moons while returning a steady stream of data to the Earth.

This complicated path to Jupiter is shown in the figure 1–1, as reproduced from the Galileo mission home page [NASA 95]. It was designed by the members of the Galileo project trajectory design group. Why did the group settle for such a seemingly complicated route? The answer to this question lies in the fact that large gravitational fields exist in the solar system and the group wanted the probe to use the potential of this energy rather than fight against it. By taking a roundabout route the probe is able to use the gravitational attraction of the Earth and Venus to change its velocity and direction and eventually send the probe outwards on an intercept course with Jupiter. This is far more efficient than sending the probe on a more direct route from Earth straight to Jupiter where the probe would have no such gravitational assistance and may not even have enough thrust or fuel to manage the mission.

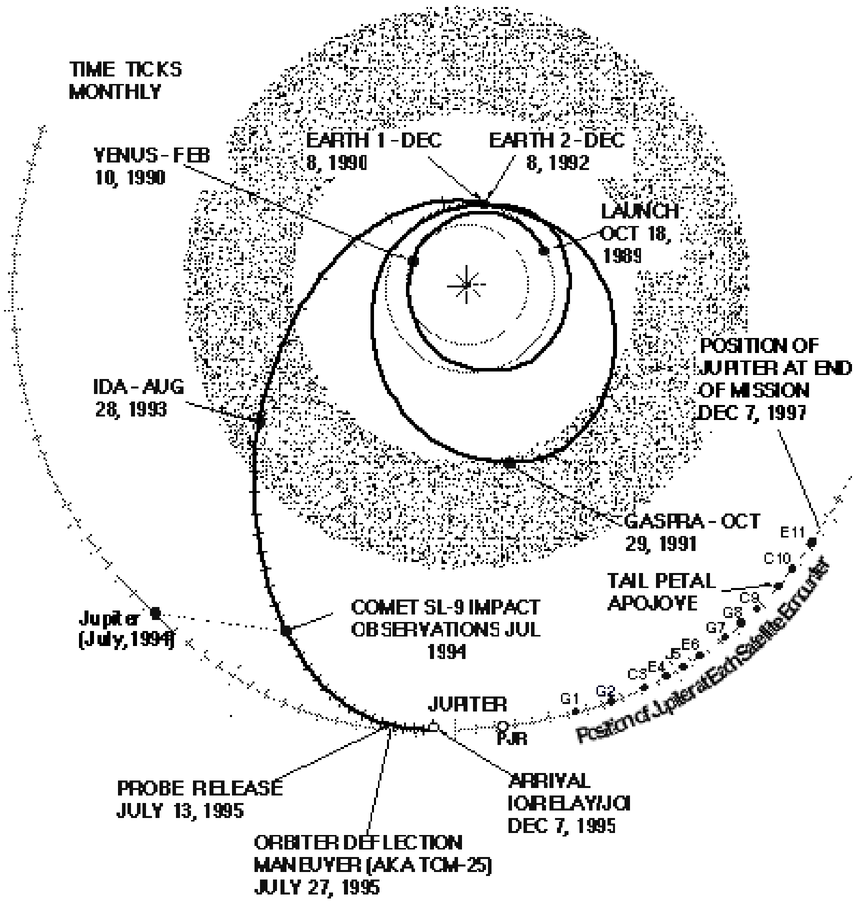

Figure 1–1: Galileo Probe - Jupiter Mission Trajectory Plot

It was the aim of this project to investigate whether genetic algorithms could try and optimise such a complicated path.

1.2 Current Methods

Between the late 1970's when planning for the Galileo mission began in earnest to 1987 when the probe was launched the project underwent many major mission changes each requiring the trajectory to be redesigned. This complicated process was the responsibility of the eleven members of the trajectory design group, arguably some of the hardest working members of the Galileo Project.

Designing a trajectory is currently expensive in terms of both labour and computation, with lots and lots of trajectories to generate and then analyse. Each redesign involves attempting to consider every possible trajectory mode that could maximise the trajectory energy of the probe. In the Galileo mission this effectively involved looking at various combinations of Venus, Earth and Mars flybys until the team eventually settled upon the VEEGA (Venus-Earth Gravity Assist) option. As this quote from the Galileo Messenger newsletter [NASA 95] demonstrates the initial stages of this design process were far from exact,

> Several brainstorming sessions took place attended by Dennis Byrnes, Roger Diehl, Bob Mitchell, and me. After discarding a number of ideas that involved various combinations of Venus, Earth and Mars flybys, the VEEGA option emerged.

Once this first important design decision had been made the group had to optimise a precise trajectory. The full trajectory is then split into two main regions for optimisation. First the interplanetary path between the Earth and Jupiter is studied followed by the tour of orbits of the probe around Jupiter. To aid the group in these tasks four main analytical tools were available, PLATO, STOUR, MOSES and FAST,

- PLATO (Planetary Trajectory Optimisation): A program used for optimising the design of interplanetary trajectories with planetary flybys.
- STOUR (Satellite Tour): A program used to design the satellite tour of a probe round a planet.
- MOSES (Multiple Orbit Satellite Encounter Software): Once a viable satellite tour has been produced by STOUR then MOSES, using a more accurate trajectory model, attempts to minimise the fuel needed to achieve this tour.
- FAST: A tool used in the final design stage, FAST has the most accurate trajectory model and is used as a final check of the fully optimised trajectory.

These programs are used, not only to optimise the basic trajectory decided on by the group in the early design stages but also to ensure that the trajectory meets science, engineering and mission constraints.

1.3 Project Aims

As stated in the last section, designing a trajectory is a complicated task, requiring lots of trajectories to be generated and then analysed. It was the aim of this project to investigate whether the group of computational models known as genetic algorithms could be applied to this type of problem with any success. In particular it was hoped to look at the initial trajectory design stages to determine whether a genetic algorithm (GA) solution to the problem could be useful in determining the best basic route to obtain gravitational assistance from other planets in the system. This could therefore potentially be useful in choosing the initial general trajectory, which could then be optimised using existing exact methods, such as PLATO and STOUR, to produce a final trajectory.

There is a previous precedent for using GAs to solve this type of problem from an article in AI expert [Wilson 93]. Steve Wilson, a partner in the Californian software company Emergent Behaviour, used a GA to try and solve the trajectory problem of a space probe moving through a system of seven planets started with random initial velocities, to reach a target point upon the other side of this simulated region of space. He claimed excellent results were possible with a very small population (25) in relatively few generations (30).

The aim of this project was to investigate a similar problem but to use a simple solar system model instead of randomly initialised planets. It was also intended to make the probe target one of the orbiting planets of this system.

1.4 Work Done and Thesis Outline

To achieve the project aims three main bodies of code had to be written:

- A genetic algorithm to run the problem.

- A 2-d solar system/probe simulation to be used in the evaluation function of the GA.

- A graphical interface to display the resultant trajectories.

These codes were written in the C programming language, with the graphical interface also using the Libsx, X library interface. A brief specification of this graphical interface is presented in chapter 6.

The theory behind the GA written in the course of this project is introduced in Chapter 2. This GA had two different control strategies and two different crossover operators available and various combinations of these were tested upon a simple trajectory problem.

The theory used to simulate the path of the probe through a model 2-d solar system is spread across two chapters. Chapter 3 details the theory used to calculate the positions of the planets as they orbit round a central Sun. This involved a mixture of Kepler's laws and Newton's law of universal gravitation. Chapter 4 then goes on to reproduce the theory used to solve for the motion of the probe through this system. No exact solution was possible for this problem, which had to be solved numerically. This chapter therefore also includes theory on the numerical methods used.

The next chapter, 5, discuses the representation of the problem coded into the GA. This representation was similar to the approach used by Wilson [Wilson 93] although a more concerted effort was made to ensure that the probe was powered by just enough thrust to make the problem interesting.

After this coding was complete the project went on to investigate the performance of the various different GA control strategies and crossover operators on a relatively simple trajectory optimisation problem. Both steady state and generation-based GA control strategies were compared along with one point and uniform crossover operators. The results of these tests are recorded in chapter 7 of this thesis and were found to show steady state control strategy with one point crossover to be the most effective combination in solving this problem.

Finally the resultant GA was tested upon a problem of the same complexity as the Galileo mission. These results, also found in chapter 7, were found to be quite promising with the probe learning to use the gravitational attraction of the Sun to assist in its path. These results were achieved with a relatively small number of seventy five chromosomes, split into three sub-populations and run for thirty generations.

Finally the conclusions of this thesis and suggestions for further work to extend this project can be found in chapter 8, the last chapter.

Chapter 2

An Introduction to Genetic Algorithms

This chapter introduces genetic algorithms (GAs), a series of computational models inspired by natural evolution.

From the study of genetics it is known that the blueprint of any organism is held in long DNA molecules. The complicated double helix structure of DNA uses long strings of four basic building blocks adenine (A), cytosine (C), guanine (G), and thymine (T) to encode all the instruction needed to build that organism. These DNA strands are grouped into chromosomes, each chromosome is built up of genes, which have a position within the chromosome and a value given by the type of base present. All the possible values that can be present are called alleles. The total genetic information held in all of the chromosomes of an organism is known as the genotype.

New organisms produced by reproduction between two parent organisms inherit genetic from both parents; what it does not get from one parent it gets from the other. In this way the new child organism can inherit characteristics from both parent organisms and new characteristics from mixing the parental ones.

Genetic material is also subject to mutation, which is manifested by random changes in the values held at gene level. In this way new gene patterns can be introduced in the child, which would not have been possible from the mixture of genetic material present in the parents.

As stated earlier, in nature the diversity or variety in a population of organisms is a direct consequence of the differences in genetic information between the individuals of which it consists. This variety involves both the structure and behaviour of the organisms. Some members of the population will therefore be better equipped to survive and reproduce in their environment than others. This leads to the concept of the fitness of an organism. Those organisms that are more adapted to living in their particular environment can be said to be fitter and will survive and reproduce at a higher rate than other less fit members of the population. Therefore over a time span of many generations the population will change to contain more individuals with the genetic information that translates into organism structures and behaviours that enable better survival and reproduction rates. This is the concept of survival of the fittest and natural selection as discussed by Charles Darwin. This change in the overall genetic makeup of the population with time is described as the evolution of the population.

In much the same way genetic algorithms attempt to evolve a solution to a specific problem from a population of potential solutions. A potential solution to a problem is usually represented by a fixed length character string known as a chromosome or genome. These strings are evaluated by the GA using a function which returns a fitness value, indicating how well the problem is solved. To create a new generation, the chromosomes in the current population are allowed to "breed" using string recombination operators. A crude evolutionary selection mechanism is used in this process to try to ensure that fitter chromosomes are given a greater chance to reproduce than less fit members of the population. This aims to help preserve and refine good parts of the chromosomes as the generations pass leading to the whole population converging towards a good solution.

The GA also applies mutation to the chromosome strings in order to ensure that gene sequences that are not present in the initial population can be produced later in the run.

This framework for a basic GA can be condensed into a general plan of five main stages,

1. Create an initial population. This is done by randomly assigning values to all the genes of each chromosome.

2. Calculate the fitness of each chromosome using an evaluation function which is representative of the particular problem being solved. This fitness is then used by a fitness function to allocate reproductive opportunities to each chromosome. The size of this allocation is dependent upon the comparison of this fitness value with the average fitness of the whole population.

3. Create a new generation from the current population. This involves selecting pairs of parent chromosomes from the current population to reproduce. This is done using recombination operators to produce children from the parent chromosomes. These children are then inserted into the new generation. The selection of parents is done by a fitness function.

4. Mutation is then applied to the population.

5. return to 2.

This scheme continues until some kind of stopping criteria is met. This can be as simple as stopping after a certain number of generations have been completed or may involve checking the fitness of all the chromosomes and stopping when these values converge.

This plan still leaves a number of questions unanswered. How are the parents chosen?, how does the reproduction process work? and how are the children inserted back into the population? These questions do not have a definite answer, there are many ways in which the five stages of the plan can be implemented leading to many variations based upon this basic GA.

For instance there are many ways to create a new generation from the existing population. In generation-based GAs, the entire current population is replaced by children to form the next generation. While in elitist generation-based GAs the fittest members of the current population are copied intact into the next generation.

In the case of the genetic algorithm written during the course of this project two different control strategies were available one generation-based and the other an elitist generation-based strategy known as a steady state genetic algorithm. These two strategies are explained in section 2.1.

The GA also used a rank-based fitness function to select parents for reproduction. This is discussed in Section 2.2.

Two different types of recombination operator were available to perform crossover. These were one point crossover and uniform crossover. The different ways in which these operators produce children from parent chromosomes are described in section 2.3.

Next the type of mutation operator employed is explained in section 2.4.

Finally, section 2.5 describes and explains the type of multi-population model used.

2.1 Control Strategy

This section contains a discussion of the types of control strategy implemented in the GA used during the course of this project.

2.1.1 Generations-Based GA Strategy

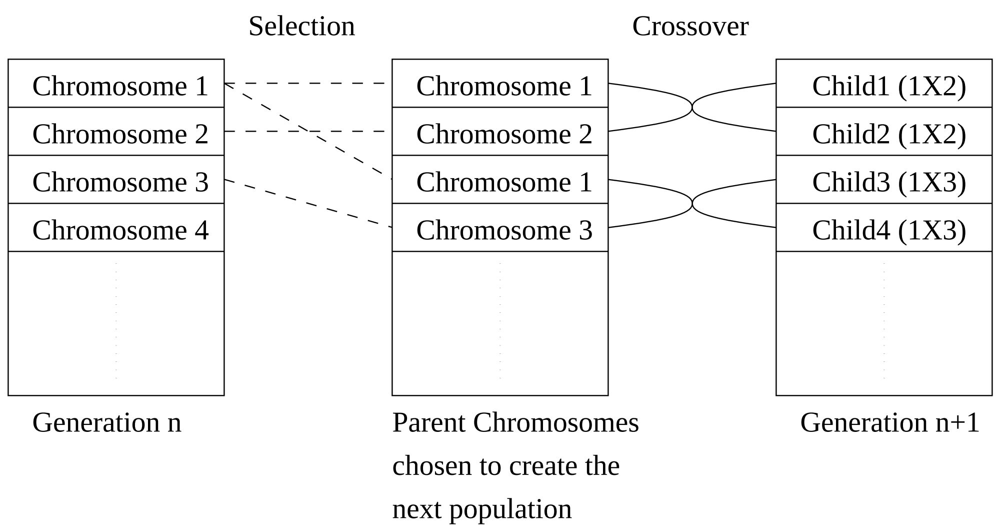

Figure 2–1: Generations-Based GA Strategy

The original genetic algorithm type investigated by Holland [Holland 75], sometimes referred to as the canonical genetic algorithm, used a generations based strategy. Figure 2–1 shows such a strategy. To evolve from one generation to the next, enough parents to produce an entire new generation are selected using the fitness function (see section 2.2). Once this intermediary stage of choosing all the parents is finished crossover is then applied to these parents to create a whole entirely new population of children.

2.1.2 Steady State GA Strategy

In addition to the generations-based strategy described in the previous section, the GA written during the course of this project was also equipped with an elitist generation-based strategy. In particular this strategy was modelled upon the Genitor GA [Whitley & Kauth 89] which was the first example of what Syswerda [Syswerda 89] named a steady state genetic algorithm.

Figure 2–2 shows this strategy,

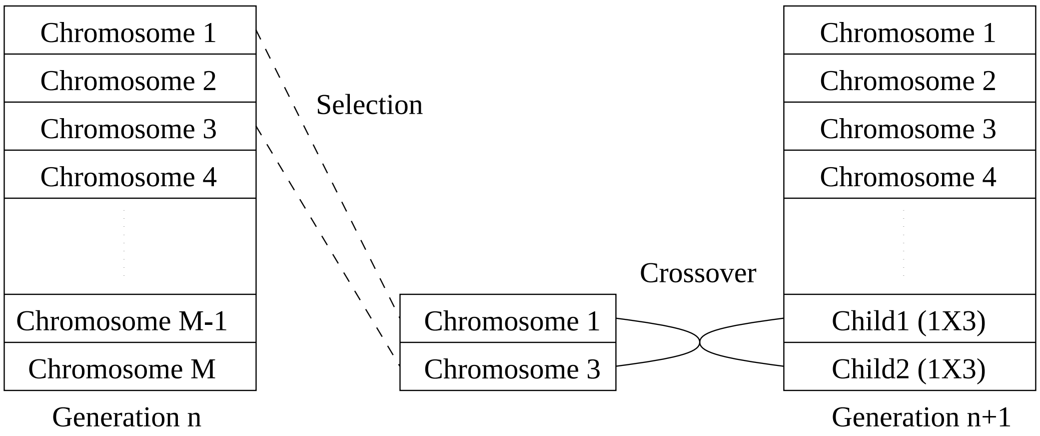

Figure 2–2: Steady State GA Strategy

The main features of this strategy are,

- Reproduction takes place one crossover at a time.
- The two offspring produced by this crossover are used to replace the two least fit individuals of the current population.

In the implementation used in this project this process was carried out M/2 times, where M is the total number of chromosomes present in the population (as shown in figure 2–2). Therefore one generation of this strategy involves only half as many crossovers as the generation-based strategy.

As in the generation-based case mutation is applied to the population after a new generation has been created and the whole process of recombination starts again.

It was hoped that the steady state strategy would result in a more aggressive search by retaining the best chromosomes. Certainly in a problem as difficult as this there seems little point in destroying a good solution almost as soon as it has been found.

2.2 Selection

Selection of parents for reproduction is done by fitness and is a vital component of the genetic algorithm. Parents must be chosen in such a way that the critical genetic information contained in the current population is preserved and hopefully enhanced in later generations.

The fitness function is used to allocate the likelihood of a chromosome being able to reproduce given its fitness value with respect to the average fitness of the whole of the population. Those chromosomes with a fitness higher than the average are given a much greater chance to reproduce than those with a fitness lower than this average.

There are two main methods used in the fitness function: roulette-wheel selection and rank-based selection. In the first chromosomes are chosen with a probability directly proportional to their relative fitness. The name roulette-wheel selection comes from imagining a roulette wheel with each chromosome taking up an area proportional to its fitness value. The method is therefore highly sensitive to the range of fitness values. This can be a problem if the fitness values are high and there is only a small range separating the best and worst solutions. Using roulette-wheel selection on this type of problem leads to the least fit solutions being chosen to reproduce almost as often as the most fit solutions.

The second method, rank-based selection is designed to overcome this problem. Here chromosomes are ranked in order of fitness and are allocated a probability of reproduction with respect to their order in this ranking. The probability of being chosen for reproduction is therefore no longer directly proportional to the relative fitnesses of the chromosomes but to their order in this ranking list.

The GA written for this project used a form of rank-based selection procedure. It divided a total length of 100 into separate sub-sections, one for each of the chromosomes in the population. The length allocated to each chromosome was then used as the probability of that chromosome being picked to be a parent at any one selection process.

The length was divided using formula 2.1. This upper limit gives the number at which the sub-section allocated to chromosome *Rank* finishes. The chromosome ranked first in the population starts it's subsection from zero while the other chromosomes start their subsections from where the previous chromosome in the rankings finishes its section.

$$Upper\ Limit = \left(100 - \frac{100}{Rank+1}\right) + \frac{100Rank}{no.chromos(1+no.chromos)} \tag{2.1}$$

The first term of equation 2.1 splits the length of 100 in the following manner. The first chromosome in the ranking is allocated the area from 0 to the upper limit of 50. The second chromosome is then allocated the area from 50 to the upper limit of 66.6 and so on for the rest of the chromosomes. However doing this does not allocate the full length of 100, as a remainder of $100/(1 + no.chromos)$ is left over. Therefore the second term of equation 2.1 splits up this remainder and adds equal sized chunks to each chromosome's region, by shifting the divisions made by the first term upwards by a small amount. Note that because of the factor, *Rank*, each division is moved a different amount, thus increasing the length of each region by the same amount.

The distribution produced by equation 2.1 is shown in the graph for a typical population of 25 chromosomes. This distribution was quite experimental. The more usual method divides a length of n(n+1)/2 into chunks of size 1,2, ...n, where n is the number of chromosomes in the population. The fittest chromosome

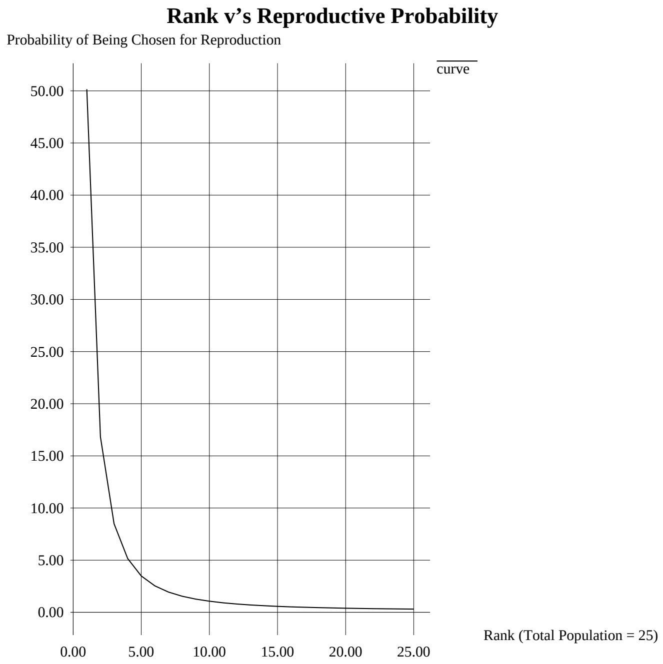

Figure 2–3: Selection Probabilities

is given the chunk of length n, the second fittest receives n-1, and so on to the least fit which receives a length of one. This therefore produces a much more even distribution of reproductive opportunities among the fittest chromosomes.

The distribution shown in figure 2–3 gives the first chromosome in the rankings a large advantage in reproduction. In most traditional GA problems this would be troublesome. The fittest chromosome in the initial population will receive this advantage, even though it may be only a poor solution to the problem. Soon almost the whole population will be derived from this chromosome and the GA may prematurely converge to a sub-optimal answer. For this type of problem the more usual method, described above would be far more effective. However the trajectory optimisation problem is quite different from most other GA problems. Firstly as the initial chromosomes described in section 7.2 show, it is most unlikely for the GA to contain more than a couple of even promising solutions to the problem in its first population. The heavy favour shown to the first two or three chromosomes in the population is therefore not such a bad problem. Also the convergence of the population towards one particular solution and the subsequent loss of genetic diversity is not a massive disadvantage. The form of the chromosomes means that combining different solutions is highly unlikely to produce anything useful. Therefore by converging the population towards one solution saves a lot of wasted computation. This fast convergence does however require high mutation and a multi-population model to help to balance for the loss of genetic diversity.

2.3 Recombination Operators

Recombination operators are used to combine the strings of two parent chromosomes to produce two child chromosomes. There are many ways of doing this, but the two used in this project were one point crossover and uniform crossover.

2.3.1 One Point Crossover

This section considers the one point crossover recombination operator. One point crossover is shown in figure 2–4

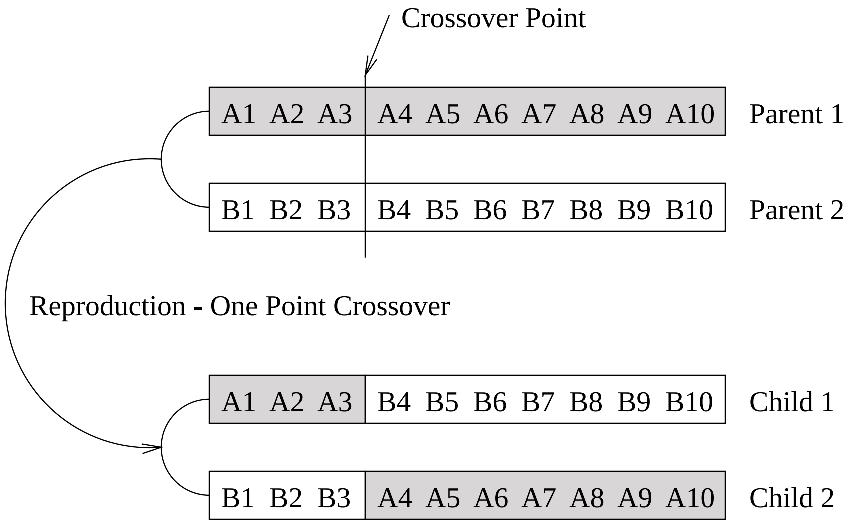

Figure 2–4: One Point Crossover

A random point, the crossover point, is chosen between the start and end of the chromosome. The genes of the first parent chromosome are then copied up to that point followed by the genes of the second parent for the remainder of the string.

To produce the second child the opposite is done, with the second parent being copied up to the crossover point and the first parent copied for the remainder.

2.3.2 Uniform Crossover

Another possible recombination operator is that of uniform crossover, this is shown in figure 2–5.

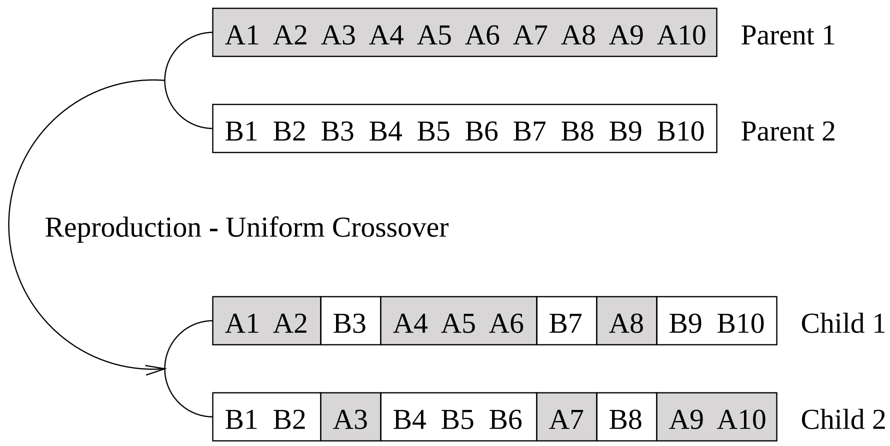

Figure 2–5: Uniform Crossover

Uniform crossover works by randomly choosing from either parent at each position in the chromosome. To create the first child a series of random choices are made on whether to take genes from the first parent or the second parent till the end of the string is reached. The second child is then produced by using the opposite parent for each gene position.

2.4 Mutation

Mutation is an essential part of any genetic algorithm model. It is unlikely, except for simple problems, that all possible gene sequences can be generated by applying recombination operators to the initial population. Therefore mutation is necessary to add some extra variety to the initial information available to the GA.

In the GA used during this project a relatively simple mutation operator was used. Each gene in the population was allowed to mutate with some probability

p_m. When a particular gene was mutated its value was changed to a randomly generated value. This is an extremely simplistic form of mutation operator and more complicated ones that may have benefited this problem are discussed in the further work section.

For the experiments of this project p_m was set to a value of two percent. This is higher than most GAs which tend to use values around one percent. However the problem representation (see chapter 5) used in this project used a string built up of real numbers, each one in the range 1-100. Therefore unlike problems which use binary strings the initial population was unlikely to contain all of these possible values for a particular string position in the initial population. The higher value of p_m was used to compensate for this and for the loss of genetic diversity caused by the selection method.

2.5 Multiple Population Models

The genetic algorithm produced for this project was equipped with a multiple population model. This enabled it to run several populations separately. Occasionally these sub-populations swap a few chromosomes in a process known as migration.

Each of these sub-populations will result in an independent search of the solution space (independent except for the effect of migration that is). Each search will be somewhat different as each will start from a different initial population thus imposing a different sampling bias on each sub-population. Running the GA with these sub-populations instead of running all the chromosomes in one large population increases the genetic diversity and should therefore improve the search. However if too many chromosomes are allowed to migrate this advantage is lost and if too few chromosomes are allowed to migrate the sub-populations are likely to prematurely converge.

In this particular GA, a probability of 85 percent was used that during one generation the best chromosome from one population would migrate and replace the worst chromosome in another population.

Chapter 3

The Solar System

As stated in the introduction the evaluation function of the genetic algorithm required a code to simulate the motion of the probe as it moves through the galaxy. This task was split into two main parts, solving for the planet positions in their various orbits and then solving for the probe position. This chapter deals with calculating the positions of the planets in their orbits around the sun while the next chapter goes on to deal with solving for the probe position.

3.1 The Elliptic Orbit

This section contains a number of the basic definitions used to describe an elliptic orbit. These definitions are used in the later sections of this chapter to help develop a theory to calculate planet positions and motion.

Figure 3–1 shows a typical elliptic orbit of one planet around the Sun. The shape of the orbital trace is an ellipse with the Sun at one focus. The definition of an ellipse is the locus of a point which moves so that its distance from a fixed point, the focus, bears a constant ratio (which must be less than 1) to its distance from a fixed line known as the directrix. Figure 3–1 shows S (the Sun position) as the focus and KL as the directrix. The position of point P (the planet position) as it moves round the orbit is therefore defined by the relation,

$$SP/PM = e\,(< 1) \tag{3.1}$$

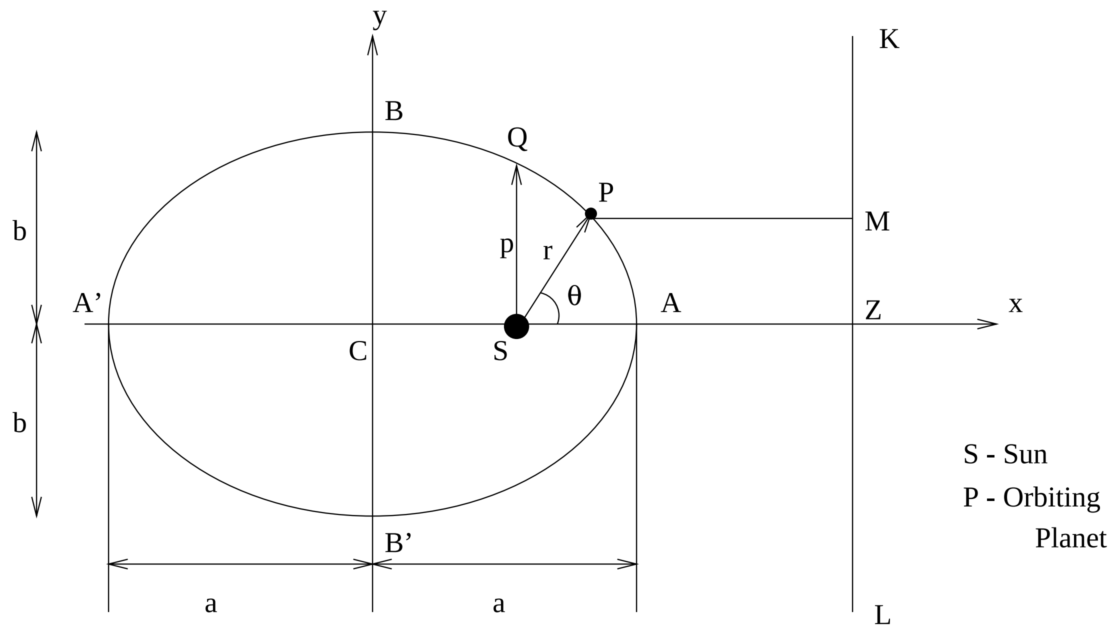

Figure 3–1: An Elliptical Orbit

where e is a constant known as the eccentricity. The eccentricity is also given by the ratio of CS/CA.

Other important definitions used to describe ellipses and elliptical orbits are given below (the uppercase letters correspond to positions on figure 3–1):

The perihelion (point closest to the Sun) = A

The aphelion (point furthest from the Sun) = A'

The major axis:

$$AA' = 2a \tag{3.2}$$

The minor axis:

$$BB' = 2b \tag{3.3}$$

The semilatus rectum:

$$SQ = p = a(1 - e^2) \tag{3.4}$$

The minor axis, b is related to the major axis, a by the following equation which involves the eccentricity, e:

$$b = a\sqrt{1-e^2} \tag{3.5}$$

Using Polar coordinates the magnitude of the vector **r** which connects the Sun and the planet, P is given by the following geometrical relation:

$$r = \frac{p}{1+e\cos\theta} = \frac{a(1-e^2)}{1+e\cos\theta} \tag{3.6}$$

The area of an ellipse is given by,

$$\begin{aligned} Area &= \pi\, a\, b && (3.7) \\ &= \pi a^2\sqrt{1-e^2} && (3.8) \end{aligned}$$

Having defined these basic terms used in the description of an elliptic orbit it is now possible to go on to the mathematical framework which forms the basis of the theory used to calculate the planets position with the passage of time.

3.1.1 Newton's Laws of Motion

Newton's three laws of motion, when first formulated, laid the foundations for the science of dynamics. Used in conjunction with Newton's law of gravitation they allow a near perfect understanding of the motion of the planets round the Sun in our solar system. The three laws are,

First Law: Every body remains in its state of rest or uniform linear motion, unless a force is applied to change that state.

Second Law: The rate of change of momentum of a body is proportional to the total force acting on the body and takes place in the direction in which the force acts. This is represented by the equation,

$$F = ma \tag{3.9}$$

where F is the force acting on the body, m is its mass and a its acceleration. Hence the acceleration of the body is directly proportional (and in the same direction) as F, but inversely proportional to mass m.

Third Law: To every action there is an equal and opposite reaction.

3.1.2 Newton's Law of Gravitation

Newton's law of universal gravitation forms the basis of celestial mechanics, it states that every particle of matter in the universe attracts every other particle of matter by a gravitational force. The force between two such particles varies directly with the product of their masses and is inversely proportional to the distance, r between them,

$$F = G\frac{m_1 m_2}{r^2} \tag{3.10}$$

Where F is the force and m_1 and m_2 are the respective masses of the two particles. G is the constant of universal gravitation (6.668×10^{-11}).

3.1.3 Kepler's Laws

Johannes Kepler (1571-1630) formulated three laws of planetary motion from data provided by an observational study of the planets' positions made by Tycho Brahe (1546-1601). Kepler's three laws are:

- First Law: The orbit of every planet is an ellipse with the sun at one of its foci.
- Second Law: Every planet moves in its orbit such that the line joining it to the sun sweeps over equal areas in equal intervals of time regardless of the line's length.
- Third Law: The square of the period of any planet is directly proportional to the cube of the semi-major axis of its orbit.

These laws, formulated from observed data are consistent with both Newton's laws and the law of Gravitation. This can be seen by the following proof, which is a special case known as the two-body problem [Roy 88]. It is worth noting that in this proof bold letters represent vectors while normal letters represent scalar quantities. For example r is the scalar magnitude of the vector $\mathbf{r}$.

Figure 3–2 shows the force of attraction from the Sun on one planet, P is **Fp** and is directed towards the Sun along the vector $\mathbf{r}$. The force of attraction from the planet, P, on the Sun is **Fs** and is also directed along the vector $\mathbf{r}$, but in the opposite direction.

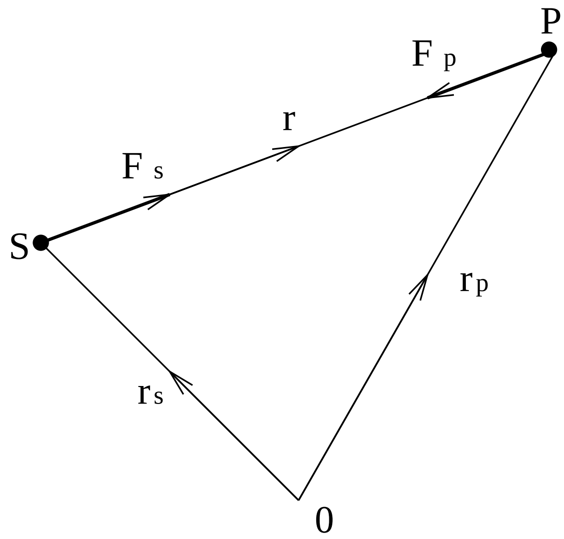

Figure 3–2: The Two-Body Problem: Forces

From Newton's third law,

$$\mathbf{F_p} = -\mathbf{F_s} \tag{3.11}$$

and from Newton's law of universal gravitation, equation 3.10,

$$\mathbf{F_p} = G\frac{m_s m_p}{r^2}\frac{\mathbf{r}}{r} \tag{3.12}$$

Using this law makes the assumption that the planet and the Sun may be treated as point masses. The positions of the Sun and the planet are given with respect to some origin, 0, by the vectors $\mathbf{r_s}$ and $\mathbf{r_p}$. Therefore using Newton's second law the vector equations of motion of the two planets are,

$$m_s \frac{d^2\mathbf{r_s}}{dt^2} = G\frac{m_s m_p}{r^2}\frac{\mathbf{r}}{r} \tag{3.13}$$

$$m_p \frac{d^2\mathbf{r_p}}{dt^2} = -G\frac{m_s m_p}{r^2}\frac{\mathbf{r}}{r} \tag{3.14}$$

these two equations, 3.13 and 3.14 can be rearranged as,

$$\frac{d^2\mathbf{r_s}}{dt^2} = G\, m_p \frac{\mathbf{r}}{r^3} \tag{3.15}$$

$$\frac{d^2\mathbf{r_p}}{dt^2} = G\, m_s \frac{\mathbf{r}}{r^3} \tag{3.16}$$

subtracting equation 3.16 from equation 3.15 gives,

$$\frac{d^2}{dt^2}(\mathbf{r_s} - \mathbf{r_p}) = \mu \frac{\mathbf{r}}{r^3} \tag{3.17}$$

where

$$\mu = G(m_s + m_p) \tag{3.18}$$

noticing from figure 3–2,

$$\mathbf{r_s} - \mathbf{r_p} = -\mathbf{r} \tag{3.19}$$

gives

$$\frac{d^2\mathbf{r}}{dt^2} + \mu \frac{\mathbf{r}}{r^3} = 0 \tag{3.20}$$

Taking the vector product of equation 3.20 with the vector $\mathbf{r}$, noting that $\mathbf{r} \times \mathbf{r} = 0$ gives,

$$\mathbf{r} \times \frac{d^2\mathbf{r}}{dt^2} = 0 \tag{3.21}$$

Integrating this, results in,

$$\mathbf{r} \times \frac{d\mathbf{r}}{dt} = \mathbf{h} \tag{3.22}$$

where **h** is a constant vector introduced as a constant of integration.

This equation is the angular momentum integral. As **h** is a constant vector pointing in the same direction for all possible values of t, the motion of the planet around the sun lies in a plane defined by the direction of **h**.

Figure 3–3 shows this plane with the position of the planet given with polar coordinates with respect to the Sun's position.

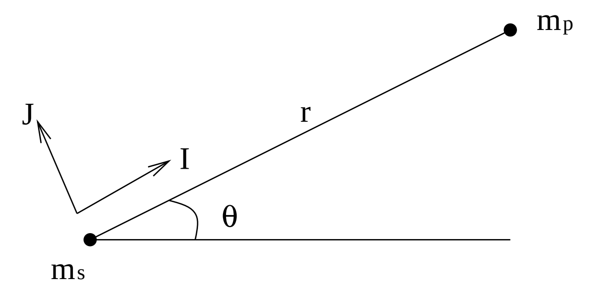

Figure 3–3: The Two-Body Problem: Velocity Components

In this representation $\frac{d\mathbf{r}}{dt}$ is made up of two velocity components, one along the radius vector, **r**, and the other perpendicular to **r**.

$$\frac{d\mathbf{r}}{dt} = \mathbf{I}\frac{dr}{dt} + \mathbf{J}r\frac{d\theta}{dt} \tag{3.23}$$

where **I** and **J** are unit vectors along and perpendicular to the vector, r as shown in figure 3–3. Then by combining equations 3.22 and 3.23 gives,

$$\mathbf{I}r \times \left(\mathbf{I}\frac{dr}{dt} + \mathbf{J}r\frac{d\theta}{dt}\right) = r^2\frac{d\theta}{dt}\mathbf{K} = \mathbf{h} \tag{3.24}$$

as

$$\mathbf{I} \times \mathbf{I} = 0 \tag{3.25}$$

and

$$\mathbf{I} \times \mathbf{J} = \mathbf{K} \tag{3.26}$$

where $\mathbf{K}$ is a unit vector perpendicular to $\mathbf{I}$ and $\mathbf{J}$ and therefore perpendicular to the plane of the orbit. Therefore in this orbital plane we can write,

$$r^2 \frac{d\theta}{dt} = h \tag{3.27}$$

This is the mathematical form of Kepler's second law with the constant h being twice the rate of description of area by the radius vector, $\mathbf{r}$. The equation can be solved by direct integration, see [Roy 88], to give the general solution,

$$r = \frac{h^2/\mu}{1 + (Ah^2/\mu)\cos(\theta - \omega)} \tag{3.28}$$

where A and ω are constants of integration. Comparing terms with equation 3.6 gives,

$$p = h^2/\mu \tag{3.29}$$

$$e = Ah^2/\mu \tag{3.30}$$

As the area of an ellipse is known from equation 3.8 and this area must be described in a time T, where T is the period of the planet in its orbit around the Sun. Then,

$$\frac{2\pi ab}{T} = h \tag{3.31}$$

as h is twice the rate of description of area by the radius vector. Substituting for h from equation 3.29 leaves,

$$\begin{aligned} T &= 2\pi(a^3/\mu)^{1/2} && (3.32) \\ &= 2\pi\left(\frac{a^3}{G(m_s + m_p)}\right)^{1/2} && (3.33) \end{aligned}$$

This equation shows that the period of the orbit of the planet around the sun depends only on the ratio of the two masses μ and the cube of the semi-major axis a.

3.2 Solar System Simulation

In order to provide a realistic environment for the space probe to navigate a solar system simulation was coded, the main features of this being:

- A centrally positioned Sun, which dominates the system in terms of both size and mass.

- A series of other smaller planets in orbit around the Sun, these orbits being ellipses of various sizes and eccentricities with the sun at one focus.

In this representation it was chosen to neglect the effects of gravitational attraction between the various planets by assuming that these effects are much smaller than the gravitational force from the Sun.

This is the case in our Solar System where the largest planet Jupiter is only a thousandth the mass of the Sun. This predominance in mass causes the planets Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune and Pluto to move to a first and high degree of approximation as if they are attracted solely by the Sun.

Therefore this approximation is correct as long as the masses of the planets are significantly smaller than that of the Sun and provided that the planets are not positioned in orbits that bring them so close that the gravitational force between them is no longer insignificant when compared to the attractive force of the Sun. Instead of introducing some complicated checking procedure to ensure that these conditions were met in the simulation the simulated solar system was modelled upon our own solar system using orbital parameters for the Sun, Venus, Earth, Mars, Jupiter and Saturn. These values are recorded in appendix I.

In the interests of efficiency it was decided to calculate the orbits of the planets once at the start of the program, then to store them in a set of arrays which could be accessed later in the program. The theory used to achieve this was based upon Kepler's laws as described in the last section.

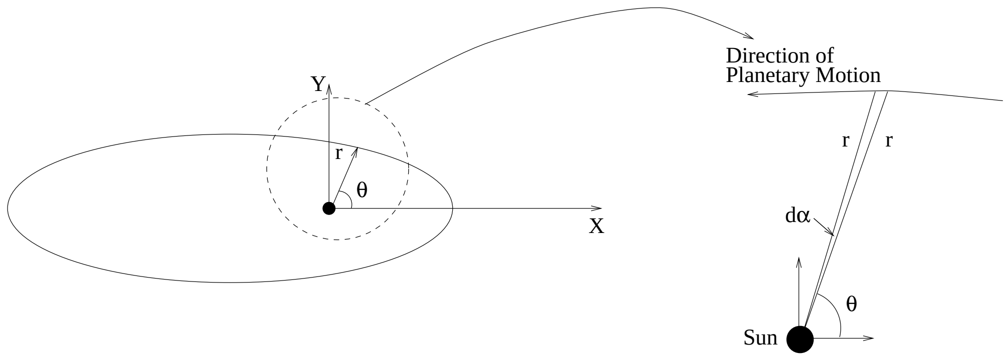

Figure 3–4: Orbital Increment of $d\alpha$

In figure 3–4 a single elliptical orbit is shown along with definitions of the angles involved. As the planet moves round in its orbit the radius vector, $\mathbf{r}$, rotates around the focal point of the ellipse where the Sun is positioned. Kepler's second law dictates that the area swept by this vector as it rotates is constant with respect to time. Therefore by comparing the area swept by $\mathbf{r}$ in a certain θ range, to the total orbital area gives the time taken by the planet to move through this θ range as a percentage of the orbital period. This can be used to build up a set of values of the planet position (given by θ) with time.

First by considering a small change in θ of $d\alpha$ an integral expression for the area swept by $\mathbf{r}$ in a certain θ range can be written as,

$$AreaSwept = \int_0^\theta \frac{1}{2} r^2 . d\alpha \tag{3.34}$$

substituting equation 3.6,

$$AreaSwept = \frac{-a^2(1-e^2)^2}{2} \int_0^\theta \frac{1}{(1+e\cos\alpha)^2} \tag{3.35}$$

This can then be integrated using the standard forms 3.36 and 3.37 from reference [Bronshtein & Semendyayev 85],

$$\int \frac{dx}{(b+c\cos ax)^2} = \frac{c \sin ax}{a(c^2-b^2)(b+c\cos ax)} - \frac{b}{c^2-b^2} \int \frac{dx}{b+c\cos ax} \tag{3.36}$$

$$\int \frac{dx}{b + c\cos ax} = \frac{2}{a\sqrt{b^2 - c^2}} \arctan \frac{(b-c)\tan(ax/2)}{\sqrt{b^2 - c^2}} \tag{3.37}$$

to produce a relation for area swept in terms of θ,

$$AreaSwept = \frac{-a^2(1-e^2)}{2}\left[\frac{e\sin\theta}{(1+e\cos\theta)} - \frac{2}{\sqrt{1-e^2}}\arctan\left(\frac{(1-e)\tan\theta/2}{\sqrt{1-e^2}}\right)\right] \tag{3.38}$$

Using Kepler's second law which states that area swept is constant with time this relation can be used to produce a tabulated set of values of θ versus time.

$$Time = \frac{Area\ Swept}{Total\ Area}.Period \tag{3.39}$$

where $Period$ is given by equation 3.33, $Area$ by equation 3.8 and $Area\ Swept$ by equation 3.38. Therefore putting a particular θ value into equation 3.39 returns the time the planet takes to move from $\theta = 0$ to this position. Hence a table of orbital positions (denoted by θ) versus time can be drawn up for one complete orbit. These values can then be used in conjunction with interpolation, to give the exact position of the planet at any time.

3.3 Method Summary

The following method summarises how the theory of the previous section is used by the code to give the position of a planet given the time elapsed since the start of the simulation.

For each planet the code required five values,

1. The planetary mass.

2. The eccentricity of the orbit.

3. The major axis of the orbit (the value of mean distance from the Sun in appendix I was used for this).

4. A offset of between 0 and 1 indicating how far the planet is round its orbit at time = 0. A value of zero indicates that it is just starting an orbit while a value of 0.5 indicates it is half way round (in terms of time).

5. The planetary radius (this was needed for collision checking).

Having stored these values the algorithm executes the following step to set up a lookup table of positions for each planet orbit:

- Use equation 3.39 to calculate a set of time versus theta values for one complete orbit starting at 0 degrees. This was done for 760 data points in the program, one for every 1/2 degree.

During the simulation a planet's position (as specified by *theta*) is retrieved from this table for a given time, t, using the following steps.

- A factor of (offset $\times$ orbit period) is added to t. This ensures the planet starts at the specified initial position instead of at $\theta = 0$.

- If t is greater than the orbit period, keep subtracting the period until it becomes smaller.

- Take this normalised time and find the two values bracketing it in the time-θ table. Use linear interpolation to find the θ value corresponding to this particular time.

- Use this θ value to calculate r from equation 3.6. These polar coordinates can then be converted to Cartesian x and y coordinates.

Chapter 4

Solving for Probe Position

This chapter deals with the method used to solve for the probe position. It begins by stating the equations of motion of the probe and then goes on to look in general at numerical methods before applying the fourth order Runge-Kutta method to the particular problem relevant to this project. References [Englefield 89] and [William 92] were used for the theory of this section.

4.1 The Equations of Motion

The equations of motion of the probe in two dimensions can be treated separately for both the x and y directions, rearranging Newton's Second law from equation 3.9 results in the two equations 4.1 and 4.2,

$$\frac{d^2x}{dt^2} = \frac{1}{M_p}Fx \tag{4.1}$$

$$\frac{d^2y}{dt^2} = \frac{1}{M_p}Fy \tag{4.2}$$

where $\frac{d^2x}{dt^2}$ and $\frac{d^2y}{dt^2}$ are the probe acceleration in the x and y directions respectively. Fx and Fy are components of the total force upon the probe resolved into x and y components. Figure 4–1 shows the simple situation of the probe and one planet. The total force upon the probe, F, can then be resolved into independent x and y components using equations 4.3 and 4.4.

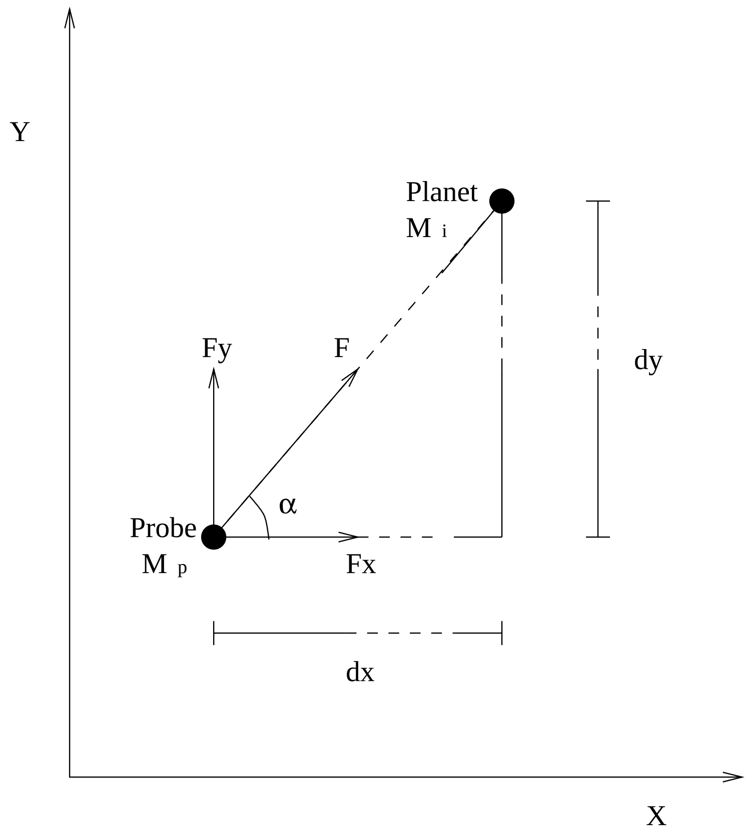

Figure 4–1: Gravitational Attraction on the Probe

$$Fx = F\cos\alpha \tag{4.3}$$

$$Fy = F\sin\alpha \tag{4.4}$$

where the total Force, F is given by Newton's law of Gravitation 3.10,

$$F = \frac{GM_pM_i}{r_{ip}^2} \tag{4.5}$$

In the more complicated situation where more than one planet is present it is necessary to sum the contributions to Fx and Fy from all the planets. This situation is represented by equations 4.6 and 4.7.

$$Fx = \sum_{i=0}^{n} F_i\cos\alpha_i \tag{4.6}$$

$$Fy = \sum_{i=0}^{n} F_i \sin \alpha_i \tag{4.7}$$

Where n is the total number of planets (including the Sun).

For non-trivial solutions of the equations of motion, for example when there are more than two planets exerting a gravitational force upon the probe, it is not possible to construct an exact solution. A numerical method is therefore the only way to obtain a solution for the probe's motion, the choices and theory behind some such methods are reviewed in the next section. This is then followed by a more detailed account of the exact method used in this project to solve for the probe position in the section after that.

4.2 Numerical Methods

This section contains numerical theory as applicable to the solution of first order ordinary differential equations, starting with Euler's equation and then moving onto the higher order methods of improved Euler and finally the fourth-order Runge-Kutta method.

This analysis begins by considering how to solve a first order ordinary differential equation of the general form

$$\frac{dy}{dx} = \dot{y} = f(x, y), \tag{4.8}$$

where no general solution can be found. Suppose that the initial conditions of $y = y_0$ when $x = x_0$ are given. The required solution therefore represents a curve (see figure 4–2) passing through the point $K(x_0, y_0)$ and equation 4.8 shows that its slope at K is $f(x_0, y_0)$ or $\dot{y}_K$ (the derivative of y with respect to x at point K).

4.2.1 Euler Formulas

The simplest way to advance this by a step of size $h = (x_1 - x_0)$ is to use Euler's formula, shown as equation 4.9 which approximates the solution curve by using

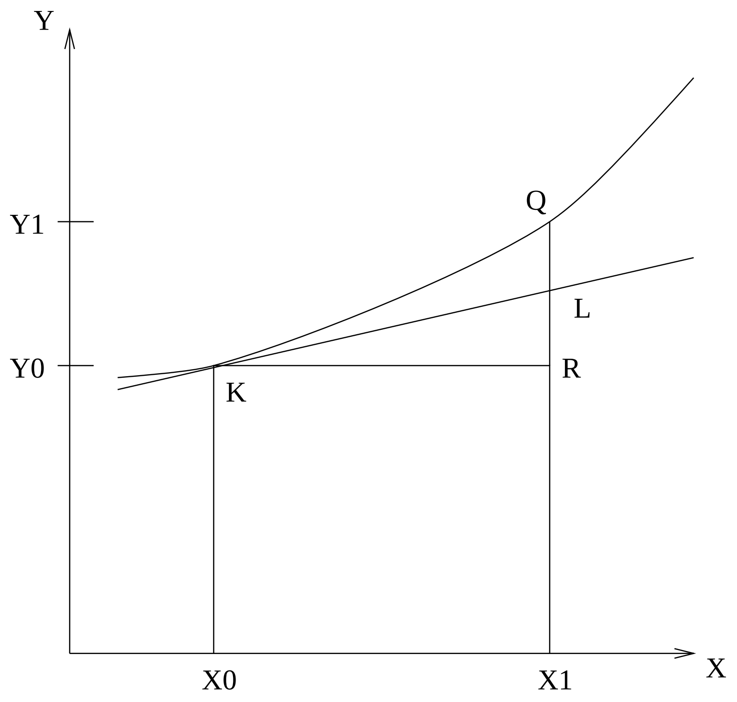

Figure 4–2: Euler's formula approximates QR by LR

the tangent at K and then steps by h in the x direction to give a corresponding y value of,

$$y_1 = y_0 + hf(x_0, y_0) \tag{4.9}$$

However this method is extremely simplistic and as such is prone to error especially when the step size is large relative to the change in gradient of the required solution. With Euler's equation, f(K) is the slope of the line by which the approximate solution passes from K to L, therefore one way to improve the accuracy of the method is to improve the accuracy of this slope. This is possible by using the average of the slope at the start and the end of the step. However as the point Q is unknown it is necessary to use the slope at L (see figure 4–2) as a close approximation of this value. Using this gives the improved Euler formula,

$$\begin{aligned} y_1 &= y_0 + \frac{1}{2}h\left\{f(K) + f(L)\right\} && (4.10) \\ &= y_0 + \frac{1}{2}h\left\{f(x_0, y_0) + f(x_0 + h, y_0 + h\dot{y}_K)\right\} && (4.11) \end{aligned}$$

4.2.2 Fourth-Order Runge-Kutta Formula

The ideas embodied in the creation of the improved Euler formula can be extended further by using an average slope calculated over even more points. The fourth-order Runge-Kutta formula uses four points, one at the start of the step, one at the end and two at the midpoint. This formula is known as fourth-order because the the error incurred in using it is proportional to h^4. This is a considerable improvement over Euler's formula which is a first-order method and involves an error proportional to h and the improved Euler formula, which is second order and introduces an error proportional to h^2 into the solution.

The formula used in the Runge-Kutta method is given by equation 4.13,

$$\begin{aligned} y_1 &= y_0 + \frac{1}{6}h\left\{f(K) + 2f(L) + 2f(M) + f(N)\right\} && (4.12) \\ &= y_0 + \frac{1}{6}h\left\{\dot{y}_K + 2\dot{y}_L + 2\dot{y}_M + \dot{y}_N\right\} && (4.13) \end{aligned}$$

where K is the known point (x_0, y_0) and the points L, M and N are given by,

$$L(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}h\dot{y}_K), \tag{4.14}$$

$$M(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}h\dot{y}_L), \tag{4.15}$$

$$N(x_0 + h, y_0 + h\dot{y}_M). \tag{4.16}$$

As these equations show the y coordinate of each point is calculated using the y derivative calculated at the previous point.

In the above equations (4.13 to 4.16) the weights used with each coordinate and the positions of the coordinates themselves were chosen so as to make the method fourth-order. This is done by comparison with a Taylor series expansion of the exact solution, which is not reproduced in this report, but the results of which are implicit in the form of the above equations.

Unlike equation 4.8 for which the above method applies, the equations of motion of the probe (equations 4.1 and 4.2) are not first-order but second-order differential equations of the general form,

$$\frac{d^2y}{dx^2} = f\left(x, y, \frac{dx}{dy}\right) \tag{4.17}$$

However any second-order differential equation can be split into two first-order equations by introducing an auxiliary variable. Equation 4.17 is therefore equivalent to the following first-order equations,

$$\frac{dp}{dx} = f\left(x, y, p\right), \frac{dy}{dx} = p. \tag{4.18}$$

To solve these two conditions requires a further boundary condition of $y = y_0$ and $p = p_0$ when $x = x_0$. Hence the initial known point K now becomes a three dimensional coordinate, $K'(x_0, y_0, p_0)$ when used to solve for the auxiliary variable p.

The method of equations 4.13 to 4.16 can be extended to simultaneously solve the two first-order equations in 4.18 using the following formulae,

$$y_1 = y_0 + \frac{1}{6}h\left\{\dot{y}_K + 2\dot{y}_L + 2\dot{y}_M + \dot{y}_N\right\} \tag{4.19}$$

$$p_1 = p_0 + \frac{1}{6}h\left\{f(K') + 2f(L') + 2f(M') + f(N')\right\} \tag{4.20}$$

where K' is the known point (x_0, y_0, p_0) and the points L', M' and N' are given by,

$$L'(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}h\dot{y}_K, p_0 + \frac{1}{2}h\dot{p}_K) \tag{4.21}$$

$$M'(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}h\dot{y}_L, p_0 + \frac{1}{2}h\dot{p}_L) \tag{4.22}$$

$$N'(x_0 + h, y_0 + h\dot{y}_M, p_0 + h\dot{p}_M) \tag{4.23}$$

where,

$$p_K = f(K'),\ p_L = f(L')\ and\ p_M = f(M') \tag{4.24}$$

and $\dot{y}_K$ is given by the third coordinate of K' (i.e. p_0), $\dot{y}_L$, $\dot{y}_M$ and $\dot{y}_N$ are given by the third coordinates of L', M' and N' respectively.

The complicated code written to implement this second-order differential equation numerical solution was validated upon a test problem equation for which an exact solution had been found. The results of this validation are shown in appendix A and were found to show the code to work as intended.

4.3 Solving the Probe Equations of Motion

In order to solve for the probe position it was decided to use the fourth order Runge-Kutta method, as described in the last section. The full equations of motion of the probe are given by:

$$\frac{d^2x}{dt^2} = \sum_{i=0}^{n} \frac{GM_i}{r_{ip}^2} \cos\alpha_i \tag{4.25}$$

$$\frac{d^2y}{dt^2} = \sum_{i=0}^{n} \frac{GM_i}{r_{ip}^2} \sin\alpha_i \tag{4.26}$$

These equations detail the movement of the probe in both they x and y directions. Splitting equation 4.25 so that it can be solved by the Runge-Kutta method gives,

$$\frac{dx}{dt} = p \tag{4.27}$$

and

$$\frac{dp}{dt} = \sum_{i=0}^{n} \frac{GM_i}{r_{ip}^2} \cos\alpha_i \tag{4.28}$$

While splitting 4.26 in the same manner produces,

$$\frac{dy}{dt} = p \tag{4.29}$$

and

$$\frac{dp}{dt} = \sum_{i=0}^{n} \frac{GM_i}{r_{ip}^2} \sin\alpha_i \tag{4.30}$$

Simultaneously solving these sets of equations would give the motion of an unpowered probe being pulled through the solar system by the gravitational attraction of the planets and the Sun. Therefore to solve for the motion of the powered probe the code adds the acceleration caused by the engine thrusts to the right hand sides of equations 4.28 and 4.30 at the appropriate time steps of the numerical solution.

Chapter 5

The Problem Representation and Evaluation Function

Usually there are only two main components of a genetic algorithm that are problem dependent, the problem representation and the evaluation function.

Defining the problem representation usually involves encoding potential solutions to the problem to be solved as a fixed-length character string. Finding an effective representation is a key issue in any genetic algorithm problem because the algorithm only directly interacts with this coded representation of the problem. If this representation is poor in terms of expressive power or inaccurate to the real problem then the results of the genetic algorithm will suffer accordingly.

As well as deciding how to represent the problem it is also necessary to devise an evaluation function that given a chromosome returns a fitness value indicating how well the solution 'held' in that chromosome solves the given problem. The goal of the genetic algorithm then involves optimising this evaluation function. Most evaluation functions involve some sort of simulation whereby the problem is run using the information from a chromosome, the outcome of this simulation is then checked and a numerical fitness value is assigned to the chromosome. The simulation can be as simple as a basic arithmetical expression or in the case of this project as complicated as modelling the motion of a space probe through a 2-D solar system.

5.1 Problem Representation

The chromosome representation used is shown in figure 5–1 and is the same as that used in the original paper by Wilson [Wilson 93]. Each chromosome is made up of a series of "course corrections". The probe thrusts with a given force and direction and then waits a specified amount of time before making the next thrust. Hence each correction is made up of three numbers, a number of time units since the last course correction, an X direction thrust unit and a Y direction thrust unit. Integer values were used for these three values. Time units were in the range 0,100 while thrust units were restricted to the range -50,50. The actual significance of each unit is described in sections 5.1.1 and 5.1.2. Arbitrarily it was chosen to have 30 such course corrections leading to a chromosome length of ninety real numbers.

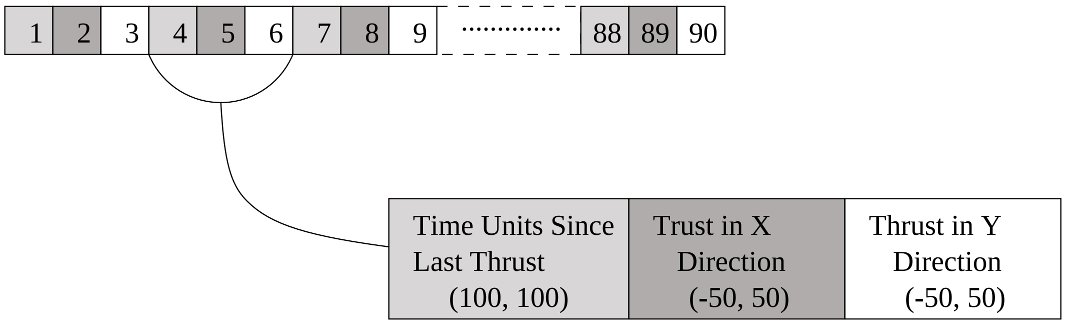

Figure 5–1: Genome Representation

This representation leads to the following total search space,

$$\begin{aligned} Total\ Search\ Space &= (No.\,of\,possible\,values\,for\,each\,gene)^{No.\,of\,genes} && (5.1)\\ &= 100^{90} = 10^{180} && (5.2) \end{aligned}$$

5.1.1 Time Units

The above representation leaves open the question of how much real time (i.e. time in the simulation) each time unit in the chromosome should represent. In this project it was decided to scale the time units so that if each time unit in the chromosome were at its maximum of 100 the last thrust would occur just before the end time of the simulation.

This means that the total length of time for which the simulation is run is important not only for giving the probe enough time to reach the target but also for the precise times when it fires its engines.

This method is far from perfect, but in the short time frame of this project there was not enough time to develop something more elegant.

5.1.2 Thrust Units

A similar problem to that of the representation of time was encountered with the thrust units; how much force would each thrust unit in the chromosome representation be assigned?

Before going on to look at this problem it is necessary to note that the thrust units used in the chromosome were not taken to be pure thrust but were taken to be a normalised value of thrust divided by probe mass. This had two advantages, firstly that the mass of the probe need not be specified and secondly these values could then be added directly onto the right hand sides of equations 4.1 and 4.2 at the appropriate simulation time step.

However it was still necessary to define some factor with which to multiply and convert the chromosome values from the range -50, 50 to a useful value. If this factor is assigned too high a value the problem becomes trivial, the presence of the solar system is irrelevant as the probe possess far greater power in its engines than the gravitational pull of the planets. In this case the probe can thrust straight towards its target and travel more or less straight there.

On the other hand, if the value assigned is too weak the probe will not be strong enough to resist the gravitational pull of the sun and will be sucked in to its gravitational well to certain destruction. This is a direct consequence of the fact that the probe was started with zero initial velocity.

Clearly some balance needs to be found between these two extreme sets of values. This was done through a combination of testing various values and by considering how the magnitude of the engine thrust compared with the gravitational pull of the planets, in particular the Sun.

Tests were carried out for various thrust scaling factors, the results of a selected few of these can be seen in appendices B and C. The starting positions of the probe and planets are shown in figure B–1 and can be seen to be a reasonably complicated problem. Running these tests were therefore quite time consuming but it was hoped they would lead to a fair value of the thrust factor. The first test began with a factor of 0.1, for which the probe proved to be far too highly over powered, the factor was then reduced in a further series of tests. Upon reaching a scaling factor of 0.03, the probe started to having to slingshot round the Sun in order to make it to the destination, this can be seen in figure B–2. However the trajectory is still very straight indicating that the gravitational pull of the planets is having little effect compared to the probes engines. The scaling factor was reduced further and the results of a scaling factor of 0.01 can be seen in figure B–3, here the path of the space probe is far more curved, the probe also travels far slower and therefore takes longer to reach the target. However the trajectory still shows sharp direction changes when the engines were fired. The thrust scaling factor was therefore reduced still further to its final value of 0.005, the results of this can be seen in figure C–18. In this case the probe needs to make three orbits of the sun before it can place itself in an orbit around the Sun that will take it out to the target destination. It was decided that this thrust scaling factor provided the best balance between the two effects discussed above and this was therefore the value used for the rest of the project. This trajectory is discussed in further detail in the results section.

The thrust scaling factor arrived at during the test runs of the code, of 0.005 was further investigated by comparing the thrust this would give the probe with the gravitational attraction of the Sun. Figure 5–2 shows the force/unit mass on the probe from the sun's gravitational pull at various distances. These values are compared to the maximum thrust available to the probe and a thrust of one unit. A maximum thrust of 50 thrust units in the chromosome representation can be seen to be scaled by 0.005, to equal the Sun's gravitational pull at a distance of approximately 2.3×10^{10} metres. The thrust represented by 1 thrust unit in the chromosome representation is equal to the Sun's gravitational pull at a distance of

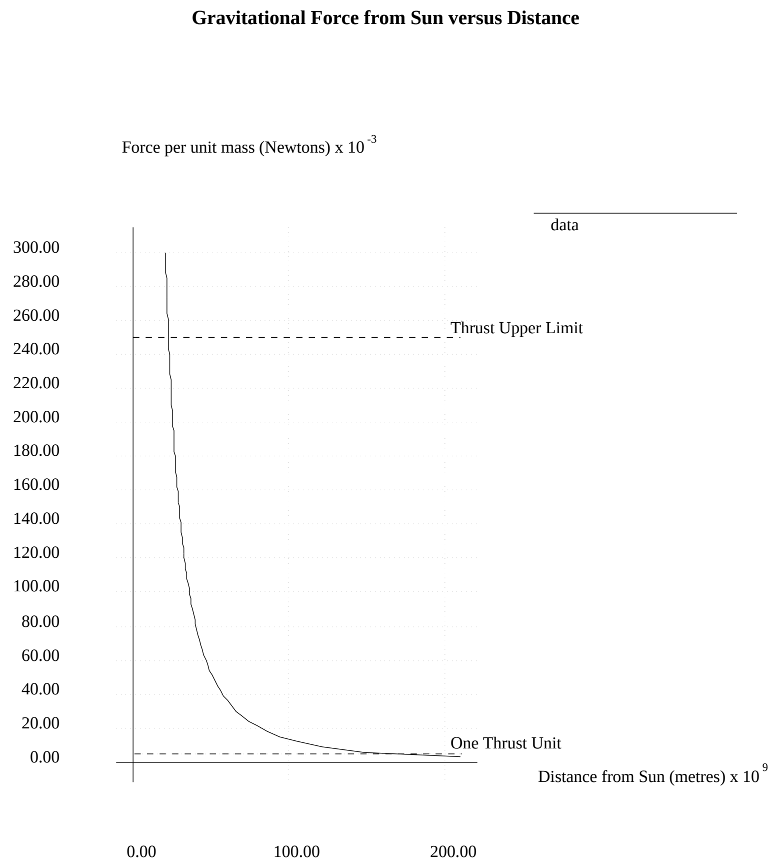

Figure 5–2: Comparative Magnitude of Probe Thrust

approximately 1.7×10^{11} metres (about 12 percent larger than the average distance of the Earth to the Sun). These figures again confirm what the earlier experiments showed, namely that the probe is neither massively under or over powered when using the thrust scaling factor of 0.005. Running the GA with this thrust scaling factor should therefore result in a problem of challenging difficulty to the GA.

5.2 The Evaluation Function

In his paper Wilson [Wilson 93] chose to base his evaluation function upon three criteria, the distance from target, amount of fuel used and length of time taken to reach the target. He considered distance from the target to be the most important factor and therefore used the square of the distance from target to severely punish individuals that missed the target by large amounts. The exact function used by Wilson is shown in equation 5.3.

$$Fitness = -1 \times (\sqrt{target\ distance}) + remaining\ fuel - time\ elapsed \qquad (5.3)$$

For reasons, mainly of simplicity this function was not used in this project. Firstly it was decided not to make the time taken to reach the target as a criteria, this purely made the evaluation function and results of the genetic algorithm easier to analyse and understand.

Secondly the target distance was not squared but divided by 1×10^7. As described in chapter 3 the solar system simulation used a rough model of part of our own solar system. The distances involved were therefore very large and squaring the distance would have made the fuel criteria totally irrelevant. This rescaled distance was therefore used to provide a suitable balance between fuel and target distance. In this way each unit of fuel used was equivalent to 10000 km of target distance. This distance may seem large until the dimensions of our solar system are remembered. It is in fact one sixth the radius of Jupiter, the largest planet and one and a half times the radius of the Earth.

The evaluation function used during the course of this project is shown in equation 5.4.

$$Fitness = (target\ distance(in\, metres)/1 \times 10^7) + fuel\ used \qquad (5.4)$$

This fitness value was also supplemented by a collision check which recorded if the probe passed within a zone five planetary radii wide around any of the planets in the system. Any solution failing this criteria was heavily penalised by having 1000000 added to its fitness. The reasons for doing this were twofold, firstly to discard any solutions where the probe is destroyed by crashing into a planet (including the target planet) and secondly because the numerical method used had a fixed step size it is beneficial to avoid the areas very close to a planet where the gravitational attraction is very high and there is a chance the numerical solution will explode. A five radii limit was imposed to add a safety factor to both these concerns.

Equation 5.4 leads to the biggest difference in approach when compared to Wilson's: here a higher fitness represents a worse solution than a lower fitness value. This is opposite to the approach of most GAs where a high fitness represents a better solution to the problem than a lower fitness. The fitness value calculated by equation 5.4 could have been standardised to work in this traditional way by taking its reciprocal but it was decided to leave it as it stands. This was done as the fitness clearly displays the effects of changes in the distance to the target and fuel used as these parameters are directly proportional to the fitness value. This also gives a theoretical (albeit impossible due to the collision criteria) best fitness of zero.

The only draw back of this scheme is remembering when comparing results that a lower value of fitness represents a fitter chromosome compared to another higher value. Considering that a higher value represents a larger distance from the target and/or more fuel should make this intuitively easier.

Chapter 6

Graphical Interface

As stated in the introduction a graphical interface was required to view the results of each trajectory. This chapter contains a brief specification of the interface which was required to read in a file of coordinates containing the motion of the planets and the probe over the time limit of the evaluation function.

A sreenshot of the interface in action is displayed in figure 6–1.

The interface contains five buttons and two slide bars, the functions of each being explained below:

- The button marked "Animate!" begins the animation from the initial positions of the planets and the probe. If no other buttons are pressed this animation continues until the final positions of all the bodies are reached. Upon reaching the end of the animation a trace is drawn for each body showing its movement during the animation.

- The button marked "Stop Anim" stops the animation at its current position and draws in traces to illustrate the motion of the various bodies up to their current positions. The animation can be restarted at any time using the "Animate!" button.

- The button marked "Reset Anim." resets the planet and probe positions to their initial positions.

- The "Timeout Val" scrollbar in the top right hand corner of the display controls the speed of the animation.

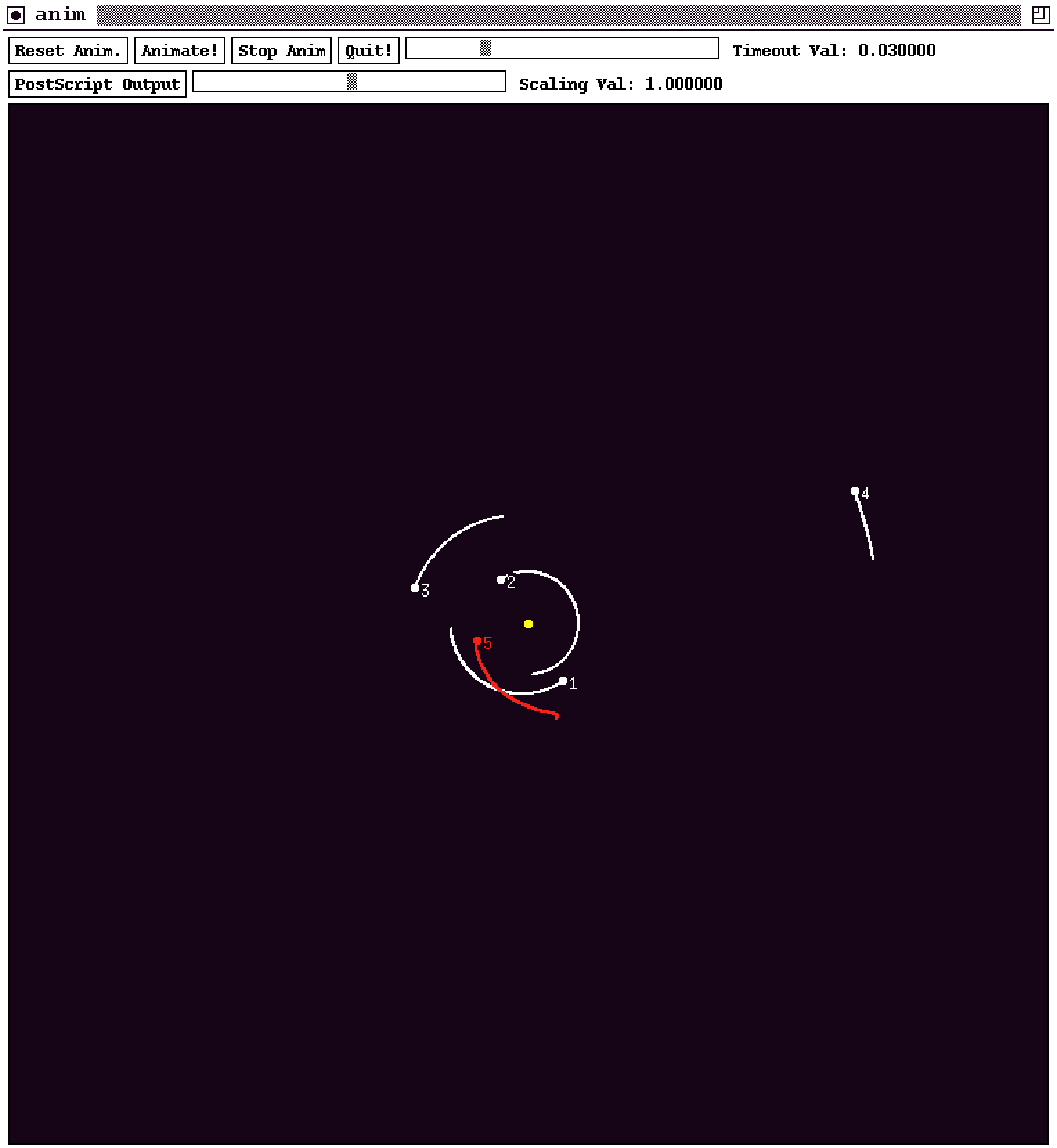

Figure 6–1: Screen Shot

- The "Scaling Val" scroll bar controls the scale of the drawing area. Adjusting this causes the screen to either zoom into or away from the Sun which is in the centre of the drawing area. Thus when the probe moves into the centre of the solar system it is possible to zoom in and closely observe its movements, similarly when it moves to the edge of the solar system it is possible to zoom out for a clear view.

- Finally the button marked "Postscript Output" dumps the current planet positions as shown in the current screen, along with traces of their movement up to that point into a postscript file. This was essential as screen dumps such as figure 6–1 were too messy and time consuming to display the results well.

Chapter 7

Experiments

The results from this project can be split into two main categories:

- Firstly a series of tests were carried out on a simple problem to investigate the performances of the different control strategies and recombination operators available to the GA written.

- Secondly the GA was run with a more complex problem, quite similar to the Galileo mission.

These two areas of research are split into sections 7.1 and 7.2.

7.1 Comparative Experiments on a Simple Problem

In this section the results of comparisons between different GA strategies and operators are displayed and analysed. Tests were made using three different GA implementations:

- Steady state control strategy with one point crossover.

- Generation-based control strategy with one point crossover.

- Steady state control strategy with uniform crossover.

It was hoped to both determine which of these combinations was best at solving the trajectory problem, as well as hopefully, learning something about the way in which the GA solves the problem.

To achieve these aims, runs of the three GA variations were carried out on a relatively simple problem. A solar system was set up containing the Sun and three other planets, Earth, Venus and Mars. The initial positions of these planets is shown in figure 7–1. The probe was started at the opposite side of this micro solar system from Mars which was chosen as its target.

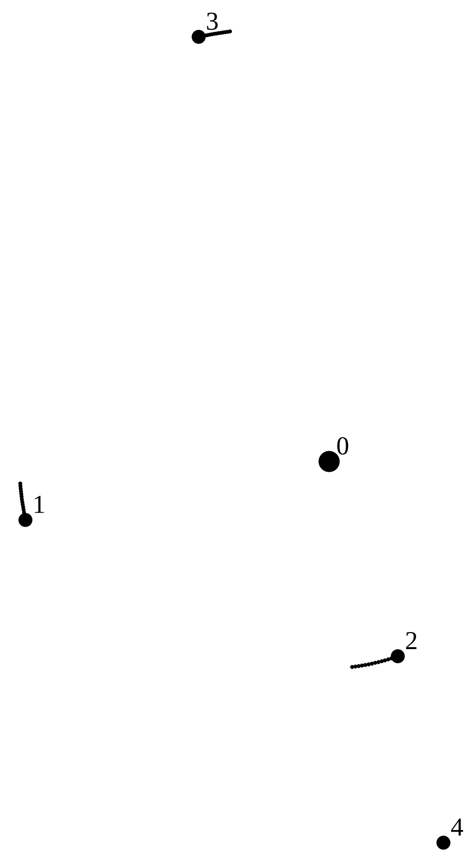

Figure 7–1: Initial Positions of Probe and Planets

Key (for figure 7–1)

0: Sun	3: Mars
1: Earth	4: The Probe
2: Venus	

Although quite simple, this problem is still sufficiently complicated, with its five bodies (including the probe) for there to be no exact solution possible. It was therefore considered to be a reasonable test of the GAs ability to find useful solutions to this type of problem.

As these tests had to be relatively fast the GA was setup with only three populations of twenty five chromosomes each. Tests with this size of population were then run for 30 generations and took between 3 and 7 hours on the departments Apollo Hewlett Packard workstations, depending on the amount of other background activity upon the machines.

Before going on to the actual analysis of results from these tests, the next two sections 7.1.1 and 7.1.2 are used to define the statistical terms and methods used in this analysis. Reference [Welkowitz *et al.* 82] was used for this section.

7.1.1 Descriptive Statistics

The aim of descriptive statistics is to summarise and describe the properties of a set of data in a concise and convenient way. Methods available to do this include calculating the frequency distribution, mean and standard deviation of a set of data values.

Graphing the data set in the form of a frequency distribution curve allows the information held by the data set to be displayed in a much clearer manner than if the data was just presented in a table. To do this, every possible value that can be present in the data set is written down and the frequency with which each particular result occurs in the data set is recorded. This distribution is then graphed to easily display this information.

This distribution of results can be further described by two numbers, the mean and the standard deviation. The mean gives information upon the location around which the frequency distribution is centred. It is calculated for a set of data values by adding up all their individual values and dividing by the total number of data points,

$$\bar{X} = \frac{\sum X}{N} \tag{7.1}$$

where,
$\bar{X}$ = mean,

$\sum X$ = sum of individual values,
N = number of data points.

However this only gives information on the central point of the set of data points, it gives no information on how widely the points of the data set vary about this centre. A commonly used measure of this variability is the standard deviation,

$$S = \sqrt{\frac{\sum \left(X - \bar{X}\right)^2}{N - 1}} \tag{7.2}$$

while the variance is given by squaring this value,

$$S^2 = \frac{\sum \left(X - \bar{X}\right)^2}{N - 1} \tag{7.3}$$

7.1.2 Inferential Statistics

The branch of statistics known as inferential statistics is used to draw conclusions about what is happening in a specific population, given only information from a small sample of this whole population. The term population, as used in statistics, refers to all the possible outcomes of a particular case being studied which is usually a very large (theoretically infinite in many cases) number. This population of results can also be represented by a theoretical population frequency distribution, although this distribution can never be calculated for a real problem, theoretical distributions like the normal distribution can used as an approximation in certain cases.

As each run of a genetic algorithm on a specific problem produces a different result the number of potential sets of results is effectively infinite. This difference in results is a consequence of the GA using a different set of random numbers on each run, leading to different initial populations and differences in the application of crossover, mutation and migration in each case. By recording the results of more and more runs a distribution of results can be built up which should tend towards the theoretical population distribution. However to exactly define the performance of the GA on a particular problem would, as stated earlier, require recording the results of an infinite number of runs on that problem. This would define the exact population frequency distribution, for the GA on that specific problem. Clearly this is impossible, as it will only ever be possible to obtain a small sample from the whole population. It is therefore necessary to use inferential statistics to try to draw any conclusions about the complete GA performance.

Having only a small sample from the population of possible results introduces problems with sampling error, caused by those cases which happened by chance to be included in the sample being unrepresentative of the population as a whole. The sample must contain enough results to be representative of the theoretical population containing all the possible results. This problem is at its greatest when only a very small sample is used. It is of relevance to this project as it could be possible that the GA 'gets lucky' and all the initial populations of these few runs contain very good solutions to the problem. This will lead to an inaccurately high estimate of the GA performance on the problem. On the other hand the GA may be 'unlucky' and receive only very poor solutions to the problem in the initial populations leading to a low estimate of performance. When comparing the GA variations this type of effect could then, for instance, make it easy to confuse a GA variation that is poor at solving a particular problem but which receives good solutions to the problem in its initial populations as being a better method than a variation which is very effective at solving the problem, but which receives only very poor solutions in its initial populations.

The only way to reduce sample error is by taking larger numbers of results, making the chances of all the runs being 'lucky' or 'unlucky' very small and ensuring a spread of results that are more representative of the population. This ensures a much fairer assessment of the GAs performance which will be much closer to the theoretical population distribution.

Deciding which of the GA variations performed best upon the test problem requires taking the sample data sets obtained from running the variations two at a time and comparing them. This comparison is done using a t-test which takes into account the sampling error when trying to prove that one GA variation performed better on the problem than the other.

As has been stated earlier the sample mean of the results from the GA variations is almost never exactly equal to the population mean because of sampling error. Therefore the populations in question (a hypothetically infinite number of runs of variation 1 and a hypothetically infinite number of runs of variation 2) may well have equal means even though the sample means are different. More information is needed to resolve exactly whether the difference in means produced by the results from the two GA variations is enough to be a reliable indication that the population means are different (and that therefore one GA variant performs better on the problem than the other) or whether the difference was due solely to sample error (and both GA variations perform equivalently well).

The aim of a t-test is to test the hypothesis that the two data sets come from the same normal distribution and that any difference in their means is due solely to sample error. Failure to prove this hypothesis to a reasonable degree of certainty leads to the conclusion that the two samples of results come from populations with different means and that therefore one variation is performing better on the problem than the other. As with any inferential statistic, this hypothesis can not be proved with absolute certainty as we are only dealing with samples of the whole population. The t-test therefore aims to prove the hypothesis to within some probability, called the criterion of significance, α which is based upon an estimation of the amount of sample error present in the results.

An example of the particular type of distribution, known as the normal distribution, is shown in figure 7–2.

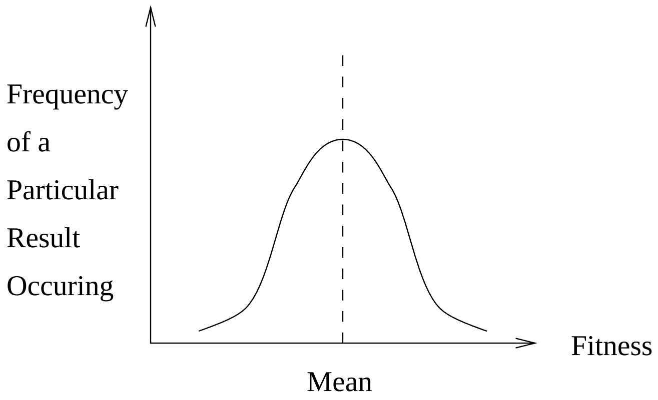

Figure 7–2: Normal Distribution

The normal distribution is symmetrically centred around the mean value and is defined by the following formula involving its mean and standard deviation (σ

is used to denote population standard deviation as opposed to sample standard deviation).

$$f(X) = \frac{1}{\sigma\sqrt{2\pi}} \exp -(X - \bar{X})^2/2\sigma^2 \tag{7.4}$$

The normal distribution is a theoretical distribution, which is useful in statistical analysis as many variables can be assumed to have a normal distribution. In this case the results from the GA can be approximately modelled by a normal distribution. Each run of the GA will produce different results depending on the initial randomly created population and the effects of random numbers in the crossover and mutation operators. As stated earlier some runs will be lucky and will begin with good solutions to the problem in the initial population, while others will contain only very poor solutions and are likely to evolve poor answers. However most runs will lie in between these two extremes and it is not unreasonable to assume that a distribution like the normal distribution will result.

The t-test is performed by calculating a t value for the difference in means between the two sets of results. This is calculated by the following formula,

$$t = \frac{\bar{X_1} - \bar{X_2}}{\sqrt{(N_1 - 1)S_1^2 + (N_2 - 1)S_2^2}} \sqrt{\frac{N_1 N_2 (N_1 + N_2 - 2)}{N_1 + N2}} \tag{7.5}$$

where,
S_1^2 = population variance estimate of sample 1
S_2^2 = population variance estimate of sample 2
N_1 = number of data points in sample 1
N_2 = number of data points in sample 2

The two values of the criterion of significance, α and degree of freedom, df of the two data sets, where

$$df = N_1 + N_2 - 2 \tag{7.6}$$

are used to lookup a corresponding critical t value from tables.

If the value of t obtained from equation 7.5 is greater in absolute value than the critical value obtained from the table then it can be stated with probability (100 - α) that the two data sets are samples from populations with different means and that the GA variation with the higher mean can be seen to be better for solving the problem.

On the other hand if the value of t obtained is lower in absolute value than the critical value obtained from the table then it can be stated with probability (100 - α) that the two data sets come from the same population and that the difference in means is due to sample error. In this case neither of the GA variations can be said to have performed better on the problem.

In theory the t test is bound by the following two assumptions,

1. Both sets of data points are sampled from normal distributions.

2. The variances of these normal distributions are equal.

In practice, however these two assumptions are quite flexible [Welkowitz *et al.* 82]. Even if the assumption of normal distribution is not met the t-test still gives fairly accurate results.

Also the second assumption can be practically ignored, provided the two sets of sampled data points are equal in size which was the case for the data analysed during this project.

7.1.3 Comparison of Steady State and Generation-Based

Comparing the steady state and generation-based strategies was not straight forward. As the two strategies apply different numbers of crossovers and evaluations between generations it is necessary to compare solutions produced at different generations in order to produce a fair comparison.

Section 2.1 states that the steady state method applies half as many crossovers than the generation-based, it therefore seems reasonable to compare the steady state solution to a generation-based solution produced in half the number of generations. However the results produced over the same number of generations are also produced here for comparison.

Ten runs of each strategy were made, with the fitness of the best chromosome from each run being recorded in table 7–1. It is worth reiterating here, that unlike most GAs the aim of this problem was to reduce the fitness value of the chromosomes, thereby getting closer to the target and/or using less fuel. Hence in the following comparisons the lower fitness values represent better solutions to the problem. The results in this table show the solutions produced by the steady state control strategy after 30 generations, to have generally lower fitnesses than the generation-based results at both 15 and 30 generations.

Table 7–1: Comparison of Generation-Based and Steady State Control Strategy

Run	Generation -Based Generation 30	Generation -Based Generation 15	Steady State Generation 30
1	1.407×10^3	1.589×10^3	1.122×10^3
2	1.508×10^3	1.574×10^3	1.370×10^3
3	1.330×10^3	1.433×10^3	1.241×10^3
4	1.371×10^3	1.439×10^3	1.153×10^3
5	1.343×10^3	1.424×10^3	1.052×10^3
6	1.554×10^3	1.405×10^3	1.101×10^3
7	1.499×10^3	1.504×10^3	1.277×10^3
8	1.293×10^3	1.347×10^3	1.130×10^3
9	1.299×10^3	1.377×10^3	1.249×10^3
10	1.170×10^3	1.369×10^3	1.040×10^3

The mean and standard deviations of these three sets of results are shown in table 7–2.

These values show the steady state results to be, on average lower than the comparative results from the generation-based strategy. However before concluding that the steady state control strategy is better at solving this trajectory optimisation problem it is necessary to test this hypothesis using a t-test.

Using the values held in table 7–1 with equation 7.5 allows the calculation of a t value for the comparison of the steady state results (at generation 30) and the generation-based (at generation 15),

Table 7–2: Comparison of Mean and Standard Deviation for Different Control Strategies

	Generation -Based Generation 30	Generation -Based Generation 15	Steady State Generation 30
Mean	1377.4	1446.1	1173.5
Standard Deviation	117.22	83.71	106.66

$$t = \frac{1446.1 - 1173.5}{\sqrt{(10-1)83.71^2 + (10-1)106.66^2}} \sqrt{\frac{100(10+10-2)}{10+10}} \tag{7.7}$$

$$= 6.3578 \tag{7.8}$$

This value of t was produced by a degree of freedom value of,

$$df = 10 + 10 - 2 = 18 \tag{7.9}$$

The line corresponding to this df from the critical t table, of reference [Welkowitz *et al.* 82] is reproduced in table 7–3.

Table 7–3: Tabulated t-test data

α	0.20	0.10	0.05	0.02	0.01	0.001
Critical t ($df = 18$)	1.330	1.734	2.101	2.552	2.878	3.992

This table shows that a 0.001 criterion of significance requires a t value greater than 3.992 in absolute value to justify rejection of the hypothesis that the two samples of results come from populations with the same mean. Since the obtained t value of 6.3578 is greater than this critical value, it is not likely that the obtained difference in means is due to sample error. In fact using $\alpha = 0.001$ implies a 99.9 percent probability that this is not the case. Therefore it is possible to reject this hypothesis and instead conclude that the two sets of results are random samples from populations with different means, and more specifically that using a steady state control strategy performs better in solving the test problem.

This difference in output can also be observed in the graphs of appendices D and E. Figures D–1 to D–10 show the convergence of the fitness of the best chromosome in each of the three sub-populations over 30 generations, for the ten runs with the steady state control strategy, while figures E–1 to E–10 show the corresponding information for the generation-based control strategy cases.

Comparison of these figures show the steady state control strategy to execute a far more aggressive search. The curves for this method show a large drop in fitness (and therefore increase in solution accuracy) in the first handful of generations followed by a further, much slower optimisation for the rest of the generations. This convergence of fitness is monotonically decreasing; the GA always improves on the best solution between generations. This is consistent with what is expected from a steady state control strategy, which by keeping the best solutions and only replacing the worst solutions in the population, rapidly converges the population towards one particular type of solution. Once all the chromosomes have converged the rapid decrease in fitness slows as the rest of the improvement in solution is provided by mutation. This method can be quite susceptible to sample error but as is the case with these results can also result in an aggressive search that is in practise very effective.

This search can be compared to the curves for the generation-based strategy, which show an altogether more erratic convergence of fitness. This strategy does not maintain the best solutions in the population as each chromosome is replaced by children in the next generation. Therefore it is possible for the fitness of the best chromosome to increase (i.e. the accuracy of the best solution worsens). This manages to occur even despite the large probability of reproduction allocated to the best chromosome in each population. Usually the main advantage in using this method is that much more genetic information is maintained in the population and the method should therefore not be as prone to sample error and premature convergence as the steady state method. However in this specific case this diversity seems not to be as effective as in other more traditional GA problems.

This point may be illustrated by the simple example of a GA being used to solve the travelling salesman problem (TSP). In this problem it is possible for two chromosomes to contain two very good solutions to different parts of the salesman's tour. If these two solutions are combined in a single chromosome they will produce a very good solution to the whole tour. The generation-based strategy provides a better environment for this combination to happen, as it is likely to maintain both chromosomes in its population. On the other hand the steady state strategy is less likely to keep the two chromosomes for long enough to combine them to form a better solution. It is more likely to focus upon the best chromosome in the population and any genetic information contained in the other chromosomes has a much greater chance of being lost from the population. It is therefore likely in the TSP problem that the steady state strategy will converge prematurely to a sub-optimal answer compared to that produced by the genetically more diverse population of the generation-based strategy. However this is clearly not the case in the trajectory design problem. The results seem to suggest that crossover between different solutions is much less effective in producing good new solutions. For example it is very unlikely that two good but completely different solutions will combine to produce an even better solution, they are far more likely to be incompatible and produce a much worse solution. Hence the large amount of diversity in the generations-based strategy seems to hamper the search more than help it, leading to a less effective search than steady state which aggressively pursues one particular form of solution from the start.

7.1.4 Comparison of One Point and Uniform Crossover

After testing control strategies it was decided to test a uniform crossover operator against a one point crossover operator. Both tests were carried out with the steady state control strategy that produced the best results in the comparison of control strategies. As both sets of test cases used this same control strategy there was no added complication of having to compare solutions from different generations. Ten runs of each case were made and the fitness of the best chromosomes in each population are shown in table 7–4. These results show the fitnesses of the chromosomes produced by one point crossover to be, on the whole, lower and therefore better than those produced by uniform crossover.

The mean and standard deviation of these two sets of results are shown in table 7–5.

Table 7–4: Comparison of One Point and Uniform Crossover (both Steady State Control Strategy)

Run	Uniform Crossover Generation 30	One Point Crossover Generation 30
1	1.300×10^3	1.122×10^3
2	1.318×10^3	1.370×10^3
3	1.479×10^3	1.241×10^3
4	1.518×10^3	1.153×10^3
5	1.344×10^3	1.052×10^3
6	1.387×10^3	1.101×10^3
7	1.431×10^3	1.277×10^3
8	1.276×10^3	1.130×10^3
9	1.459×10^3	1.249×10^3
10	1.237×10^3	1.040×10^3

These values confirm the lower average values of fitness of the chromosomes produced using one point crossover. As before a t-test is used to check that this difference in results is not likely to be caused by sample error.

Using the value of table 7–4 in equation 7.5 allows a t value to be computed for the comparison of results produced by uniform crossover with those produced by one point crossover (both after 30 generations),

Table 7–5: Comparison of Mean and Standard Deviation for Different Crossover Operators

	Uniform Crossover Generation 30	One Point Crossover Generation 30
Mean	1374.9	1173.5
Standard Deviation	94.45	106.66

$$t = \frac{1374.9 - 1173.5}{\sqrt{(10-1)94.45^2 + (10-1)106.66^2}}\sqrt{\frac{100(10+10-2)}{10+10}} \tag{7.10}$$
$$= 4.4704 \tag{7.11}$$

This value is again greater than the critical t value for a 0.001 criterion of significance (see table 7–3). It is therefore possible to conclude with a 99.9 percent certainty that the two sets of results are sampled from populations with different means, and more specifically that using one point crossover provides generally better results than uniform crossover when solving this test problem.

This difference in output can also be observed in the graphs of appendices D and E. Figures D–1 to D–10 show the convergence of the fitness of the best chromosome in each of the three sub-populations over 30 generations for the ten runs of the one point crossover cases, while figures F–1 to F–10 shows the corresponding information for the uniform crossover cases.

Using uniform crossover can be seen to make a marked difference upon the convergence of these fitness values, which are no longer monotonically decreasing. Even though a steady state control strategy is used it is still possible for the fitness to increase between generations i.e. it is still possible for the GA to destroy the best solutions. To understand how this is able to happen it is necessary to consider the information graphed in the figures of appendix G and H. Figures G–1 to G–10 show the changes, with generation, of the standard deviation of the fitnesses of the best five chromosomes in each of the sub-populations for each of the ten runs made using one point crossover. Figures H–1 to H–10 show the same information for the ten runs made using uniform crossover. These standard deviation values can be used as an indicator of how similar the first five chromosomes of each population are to each other.

Although the graphs are quite messy it is possible to state in general terms that both cases start with approximately the same standard deviation between the best five fitnesses. However as the number of generations progresses this value drops for the one point crossover cases while staying quite high for the uniform crossover cases. This decrease for the one point crossover cases matches what we know the about the steady state control method, namely that the population quickly converges. However in the cases with uniform crossover, this crossover operator is too disruptive to allow such a convergence in the population and the fitness values remain quite diverse. Therefore despite the fact that the best solutions are kept in the population, these chromosomes are still subject to mutation, which can easily destroy a good solution. When this happens in the uniform crossover case there is no backup of the best solution in the rest of the population and the fitness increases to the next best solution. In the one point crossover case if this happens the first five chromosomes are all very similar, if not the same exactly the same and it is highly unlikely that they will all be destroyed (or worsened) by mutation so the fitness of the best solution is not allowed to increase.

7.1.5 Analysis of Best Solution

Before going on to look at the results of a more complicated problem it is worth taking a quick look at the best solutions produced to this simple problem. As an example the results of one of the steady state, one point crossover runs are reproduced below.

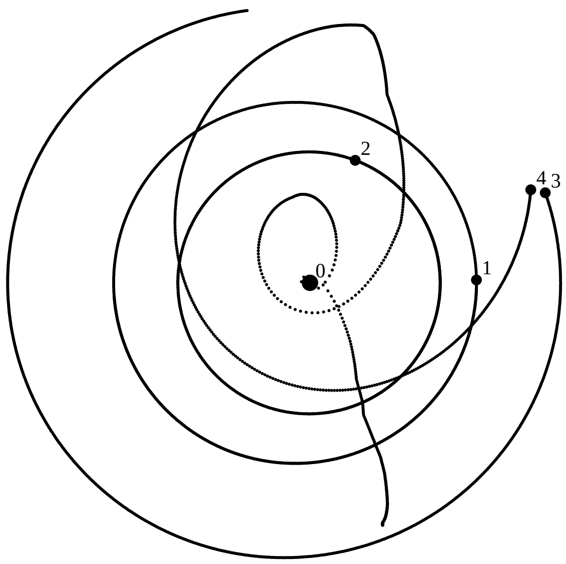

Figure 7–3: Generation 5 - Best Chromosome

Figure 7–3 shows the best solution produced by the GA at the fifth generation. The probe can be seen to make three orbits of the Sun of ascending sizes before meeting up with the target planet. It is quite impressive that the probe manages to reach its target destination this early in the solution. It does however take a rather roundabout route and uses just over half (1544 units out of a 3000 total) of its fuel in doing so.

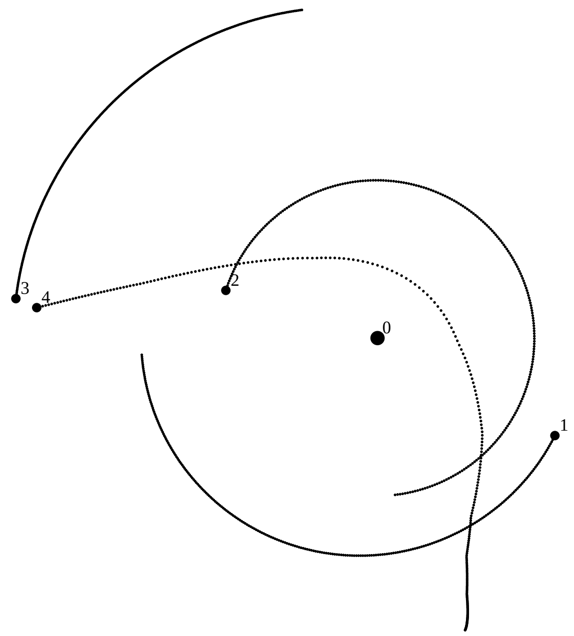

Figure 7–4: Generation 11 - Best Chromosome

Figure 7–4 shows the best solution produced by the GA after eleven generations. The GA has now found a far faster route to its target, by using one fly by of the Sun the probe is able to speed itself towards the target. This solution also saves on fuel using only 1265 units. After this generation the trajectory changes very little, although by the final generation of thirty the fuel requirement has been further reduced to 1016 units, although these gains mainly came from reducing the magnitude of thrusts made after the target planet had been passed.

These results show initial encouragement for the GA method, it copes very well with this problem, finding the target easily and reducing the fuel requirement effectively. However this was quite a simple problem and the more realistic tests of the next section are likely to give the method a more accurate evaluation.

7.2 Results from a Realistic Problem

After completing testing of the various GA variations it was decided to test the best of those variations, the GA using a steady state control strategy and one point crossover, upon a more realistic type of problem similar in terms of complexity to the Galileo mission. The results from this problem are shown in appendix C–1. To solve this problem three sub-populations of twenty five chromosomes each were run for thirty generations.

A solar system was set up with a Sun and five planets, Earth, Venus, Mars, Jupiter and Saturn. The starting positions of the planets and probe are shown in appendix C, figure C–1. The probe was started near the Earth and a target of the furthest planet from the Sun, Saturn was chosen. The initial position of Saturn was on the probe's side of the Sun, so the problem set before the GA was a difficult one. Could it learn to use the potential of the Sun's gravitational force to swing the probe back round 180 degrees and send it on its way to Saturn?

The difficulty of this solution was demonstrated by the initial randomly created population. Looking at one of the sub-populations of twenty five chromosomes, only eleven of these avoided crashing straight into the Sun. As the probe is started quite close to the Sun a good series of initial thrusts appears to be required to prevent this early catastrophe. This failure of so many of the random solutions is also good validation for the thrust calibration factor of 0.005. If it had been a lot higher most of these random solutions would have been able to resist the attraction of the Sun to head off in lots of random directions. But with this particular setting the GA was certainly facing a most difficult problem.

The trajectories of the surviving eleven solutions are shown in figures C–2 to C–12. Even these solutions are not particularly encouraging, eight of them merely manage to avoid the Sun, which changes their direction only slightly as they are sped out through the other side of the solar system and away from Saturn. Only three of the initial randomly created solutions show any encouragement at all and even then they all miss Saturn by a large amount. Both the sixteenth and twenty first chromosomes (solutions shown in figures C–7 and C–9) make a single orbit close round the Sun. Although in both cases, after this orbit the probe heads out the opposite side of the solar system from Saturn these results are slightly encouraging. If the probe can thrust differently during the short orbit around the Sun perhaps it can put itself into a larger elliptical orbit which will eventually take it out towards Saturn. Also the fifteenth chromosome of this initial population approaches the Sun and is sharply turned through almost 360 degrees to shoot out of the same side of the solar system that it started. It is therefore the only solution from the total of twenty five that manages to pass back through the same side of the solar system as Saturn is positioned even though it still misses its target by a sizeable distance. This trajectory is shown in figure C–6, although because of the scale the sharp turn of the probe around the Sun is not very clear.

The convergence of the fitness of best chromosome in each of the three sub-populations is traced from this not exactly promising starting position to generation 30 in figure C–13.

By generation 11 (see figure C–14) the best solution in the population is still very similar to the solution of chromosome fifteen in the initial population. The probe still turns sharply round the Sun, but now it leaves the Sun at a slightly better angle and gets slightly closer to Saturn, although it is still a very long way from reaching its target.

However by generation 19 the GA makes a break through, the best solution makes two and a half orbits of the Sun before shooting off out in the general direction of Saturn, to which it gets reasonably close. This trajectory is shown in figures C–15 and C–16. During the course of the remaining 11 generations this trajectory is optimised until at generation 30 the GA has succeed in its task and is getting very close to Saturn (without actually hitting the planet and being destroyed). This final optimised trajectory is shown in figures C–17 and C–18.

These results show the method to work well, with the probe reaching its target in what appears to be quite a difficult problem. The GA is seen to evolve solutions vastly superior to those created in the initial population and what is more does so in a relatively small number of generations. In 30 generations the GA will have only evaluated a mere 1215 chromosomes (75 in the initial population followed by 38 for each generation after that), a tiny number when compared to the total search space of 100^{90}, a number so large that it can not be calculated or stored on an ordinary pocket calculator. This suggests the GA has made a very efficient search.

Although it is possible to conclude that this trajectory is very successful in terms of reaching the target it is not so easy to make any conclusions about how efficient a route it took. Unfortunately no other results or methods were available during the course of this project for an accurate comparison to be made. However it is possible to state a few general conclusions on the efficiency of this solution. Firstly the final trajectory involves the probe closely orbiting the Sun, as the Sun is the largest mass in the system it also possess the greatest potential amount of gravitational energy that the probe can use. It is therefore highly likely that the optimal solution to this problem also uses the Sun in much the same way as the trajectory produced by the GA does. Also the trajectory produced by the GA can be compared reasonably favourably (albeit in very general terms) with that of the Galileo mission, shown in figure 1–1. Both trajectories spend a couple of orbits of the Sun picking up velocity before heading out through the solar system to their respective targets.

Chapter 8

Discussion

8.1 Conclusions

A main aim of this project was to test different variations of the basic genetic algorithm model upon a trajectory optimisation problem. The results of these comparisons showed the aggressive search of a steady state control strategy with one point crossover to be the most effective variation tested.

The results of this project show promise for the use of genetic algorithms in the optimisation of interplanetary trajectories. The GA method was able to find solutions to all the problems tested during the course of this project that not only reached their targets but also appeared to do so in an efficient manner. A very encouraging feature of these results was the fact that the probe learned to use the immense potential of the Sun's gravitational energy to save on fuel and increase the efficiency of its route.

These solutions were particularly impressive considering the small numbers of evaluations needed to produce them. All the problems were run with three sub-populations of 25 chromosomes each for 30 generations. The representation used during this project contained 100^{90} possible chromosomes, during 30 generations the steady state control method evaluates only 1215 chromosomes, a minute fraction of the total search space. These figures certainly indicate the GA to be making a very effective search of the total search space.

Despite these excellent results certain limitations of this project must also be considered. In particular all the tests used a zero initial velocity for the probe, whereas a real space probe would leave its launch planet with approximately the same velocity as that planet; the probe should therefore start the simulation in an orbit around the Sun. Giving the probe this extra initial energy would make the current problem easier but would also open up the possibility of using the planets for gravitational assistance as the probe would not be pulled straight towards the Sun at the start of the simulation. Using the planets for gravitational assistance has two main advantages: firstly the probe will not have to pass so close to the Sun and will be exposed to much less damaging radiation and secondly unlike the Sun the planets are moving and could therefore be used to 'pull' the probe across the solar system in their wake. This would potentially be a very efficient way to get from one side of the solar system to the other, but as the planets have a much smaller mass than the Sun it will be a difficult effect to harness. However the results of this project show the GA method to be quite effective in solving trajectory type problems and there is no reason to believe that the method will not overcome this extra difficulty. Certainly with more generations and a bigger populations there should be no reason why this more interesting problem can not be solved.

8.2 Further Work

The implementation part of this project was quite time consuming, requiring the production and testing of three large bodies of code. There was therefore little time to test as many cases as wanted. In particular there was no time to test the effectiveness of different selection processes or mutation operators:

- The selection process used by the genetic algorithm was quite an experimental formulation. To obtain an accurate assessment of how well this method worked it would be necessary to compare results with a more traditional method. However as time was limited and the existing method worked well on all the test problems this comparison was never made.

- The steady state method is highly reliant upon mutation to both add genetic diversity and prevent premature convergence of the population. It would therefore certainly be worth while investigating other mutation operators such as step mutation. Step mutation randomly picks a gene to mutate but instead of replacing it with a totally random number it alters the current value by some small amount. This could be applied in conjunction with the existing mutation operator to hopefully 'fine tune' the thrust values of a trajectory as it evolves.

Another improvement to the results of this project would be to add an initial velocity to the probe. It would then be possible to simulate the launch of the probe from a planet by starting the probe next to the planet with an escape velocity in some direction away from the planet. Also this initial velocity would make it easier for the probe to use the gravitational attraction of the planets instead of the Sun.

In addition to trying to improve the existing method it would also be very informative to compare it directly with existing methods, such as hill climbing or simulated annealing. Only then could the true worth of genetic algorithm solutions to this problem be determined. Also if more data was available on missions such as Galileo and Voyager, the results of the algorithm could be directly compared with the trajectories used in these missions.

This leads to another possible expansion of the current work, the evaluation function could be altered to see how the GA method copes with more complex mission constraints. The most important restriction would be to prevent the probe from passing within a certain distance of the sun. This would make the solution much more realistic and would also test the GA with the increased difficulty of having to use the planets for gravitational assistance instead of the Sun.

8.3 Summary

This thesis demonstrates the potential of genetic algorithms in providing solutions to the design of inter-planetary trajectories. Although there are still some further difficulties to be resolved if this goal is ever to be fully achieved, the results produced certainly demonstrate that the method maybe useful.

In particular current methods for resolving the question of which planets of the system to use for gravitational assistance to produce the most effective trajectory are far from exact. It is certainly conceivable that a GA equipped with a moderately accurate solar system simulation could be used to obtain this information, which could be used as the basis of a trajectory which could then be finished with existing optimisation methods.

Bibliography

[Bronshtein & Semendyayev 85] I.N. Bronshtein and K.A. Semendyayev. *Handbook of Mathematics.* Van Nostrand Reinhold Company, 1985.

[Englefield 89] M.J. Englefield. *Mathematical Methods for Engineering and Science Students.* Edward Arnold, 1989.

[Holland 75] J. Holland. *Adaption in Natural and Artificial Systems.* University of Michigan Press, 1975.

[NASA 95] NASA. *Galileo Mission Homepage.* http://www.jpl.nasa.gov/galileo, 1995.

[Roy 88] A.E. Roy. *Orbital Motion.* Adam Hilger, Bristol and Philadelphia, 1988.

[Syswerda 89] G. Syswerda. A study of reproduction in generational and steady-state genetic algorithms. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 116–121. Morgan-Kaufmann, 1989.

[Welkowitz *et al.* 82] J. Welkowitz, R.B. Ewen, and J. Cohen. *Introductory Statistics for the Behavioural Sciences.* Harcourt Brace Jovanovich, Inc, 1982.

[Whitley & Kauth 89] D. Whitley and J. Kauth. Genitor: a different genetic algorithm. In *Proc. 3rd International Conf. on Genetic Algorithms*, pages 116–121. Morgan-Kaufmann, 1989.

[William 92] H. William. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1992.

[Wilson 93] S. Wilson. How to grow a starship pilot. *AI Expert*, December 1993.

Appendix A

Runge-Kutta Function Validation

The second order Runge-Kutta algorithm was tested upon using a couple of second order ordinary equations for which exact solutions had been found. The most complicated example is reproduced in this appendix as verification that the algorithm works as planned.

$$\frac{d^2x}{dt^2} - 5\frac{dx}{dt} - 6y = 14e^{-t} \tag{A.1}$$

with the initial boundary conditions $y = \frac{dx}{dt} = 9$ at $t = 0$. Has an exact solution given by,

$$x = 2e^{6t} + e^{-t} - 2xe^{-t} \tag{A.2}$$

Figure A–1 shows the exact solution plotted with numerical solutions of various step sizes. It can been seen that at step sizes of 0.1 or less the numerical and exact solutions lie on top of each other thus giving a good validation that the code is working correctly.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double function(double, double, double);
void runge(double*, double*, double*, double);

void main(void)
```

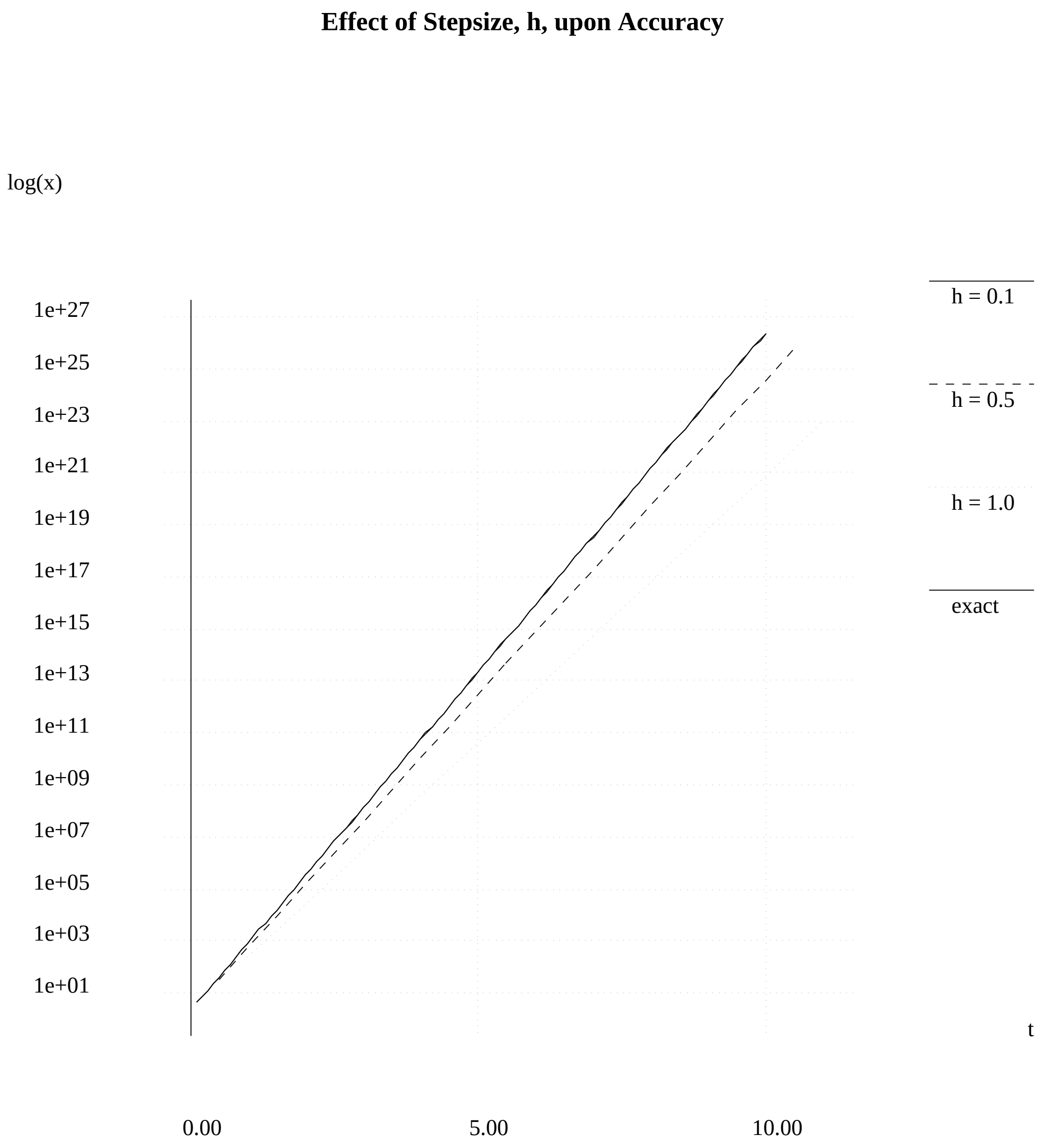

Figure A–1: Numerical v's Exact Solution

```
{
  int i;
  double x,t,p,h,tf,e,err;

  tf = 10;       /* Final Time */
  h = 0.01;      /* Stepsize */
  t = 0.0;       /* Initial Time */
  x = 3.0;       /* Boundary Condition */
  p = 9.0;       /* Boundary Condition */

  while (t <= tf) {

    runge(&t, &x, &p, h);

    e=(2.0*exp(6.0*t))+exp(-t)-(2*t*exp(-t));

    err = 100-(x/e)*100.0;

    printf("t= %e, x= %e, p= %e e= %e, error= %f\n", t, x, p, e, err);
  }
}

void runge(double *temp_t, double *temp_x, double *temp_p, double h)
{
  double x, t, p;
  double xk, pk, xl, pl, xm, pm, xn, pn;

  t = *temp_t;
  x = *temp_x;
  p = *temp_p;

  xk = p;
  pk = function(t,x,p);
  t = t+0.5*h;
  xl = p+(0.5*h*pk);
  pl = function(t,x+(0.5*h*xk),xl);
  xm = p+(0.5*h*pl);
  pm = function(t,x+(0.5*h*xl),xm);
  t = t+0.5*h;
  xn = p+(h*pm);
  pn = function(t,x+(h*xm),xn);
  x = x+((h/6)*(xk+(2.0*xl)+(2.0*xm)+xn));
  p = p+((h/6)*(pk+(2.0*pl)+(2.0*pm)+pn));

  *temp_t = t;
  *temp_x = x;
  *temp_p = p;
}

double function(double t, double x, double p)
{
  double f;
```

```
  f = (5.0*p)+(6.0*x)+14*exp(-t);
  return f;
}
```

Appendix B

Selected Thrust Unit Calibration Test Results

KEY

0: The Sun

1: The Earth

2: Venus

3: Mars

4: Jupiter

5: Saturn

6: The Probe

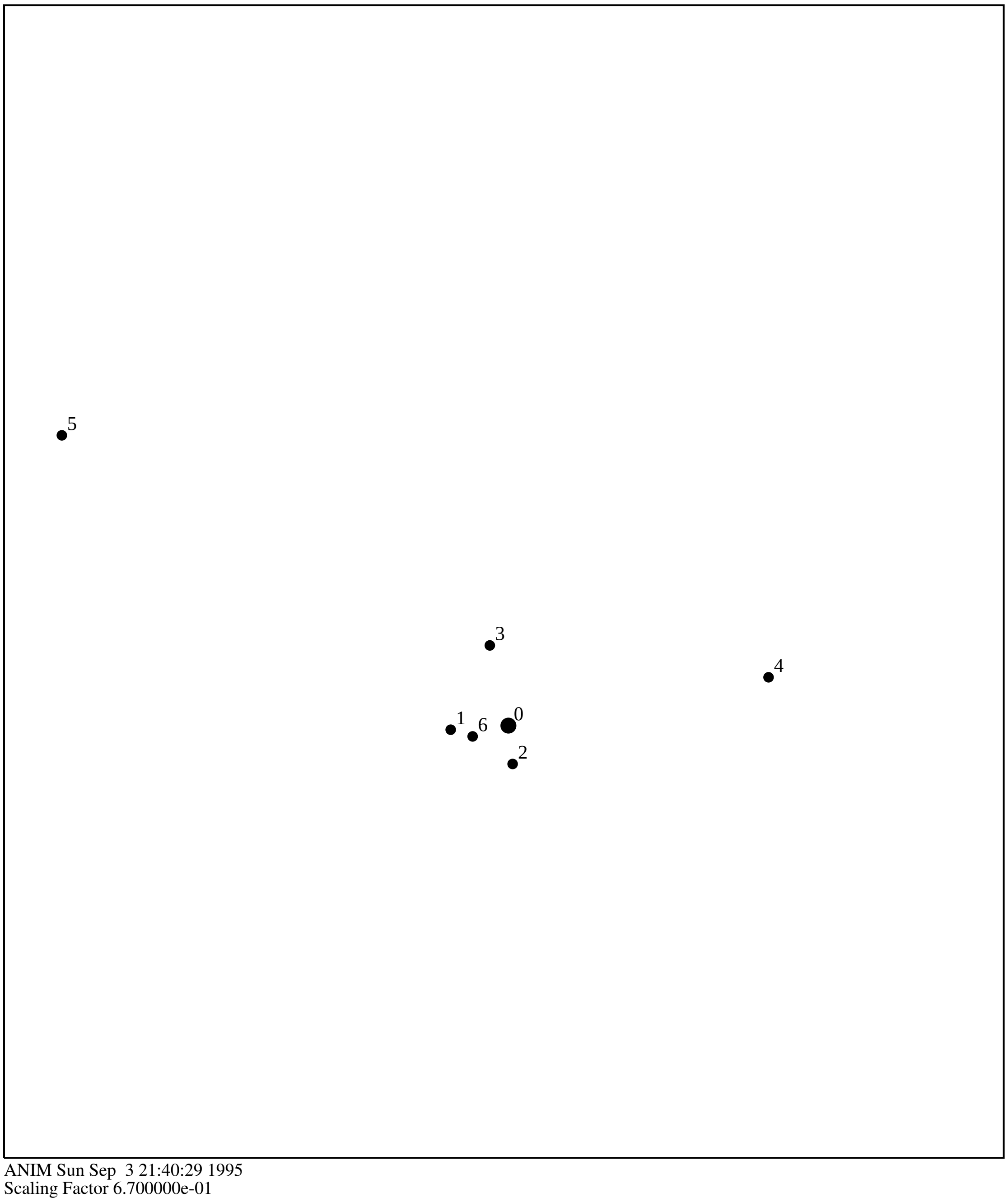

Figure B–1: Starting Positions of Planets and Probe

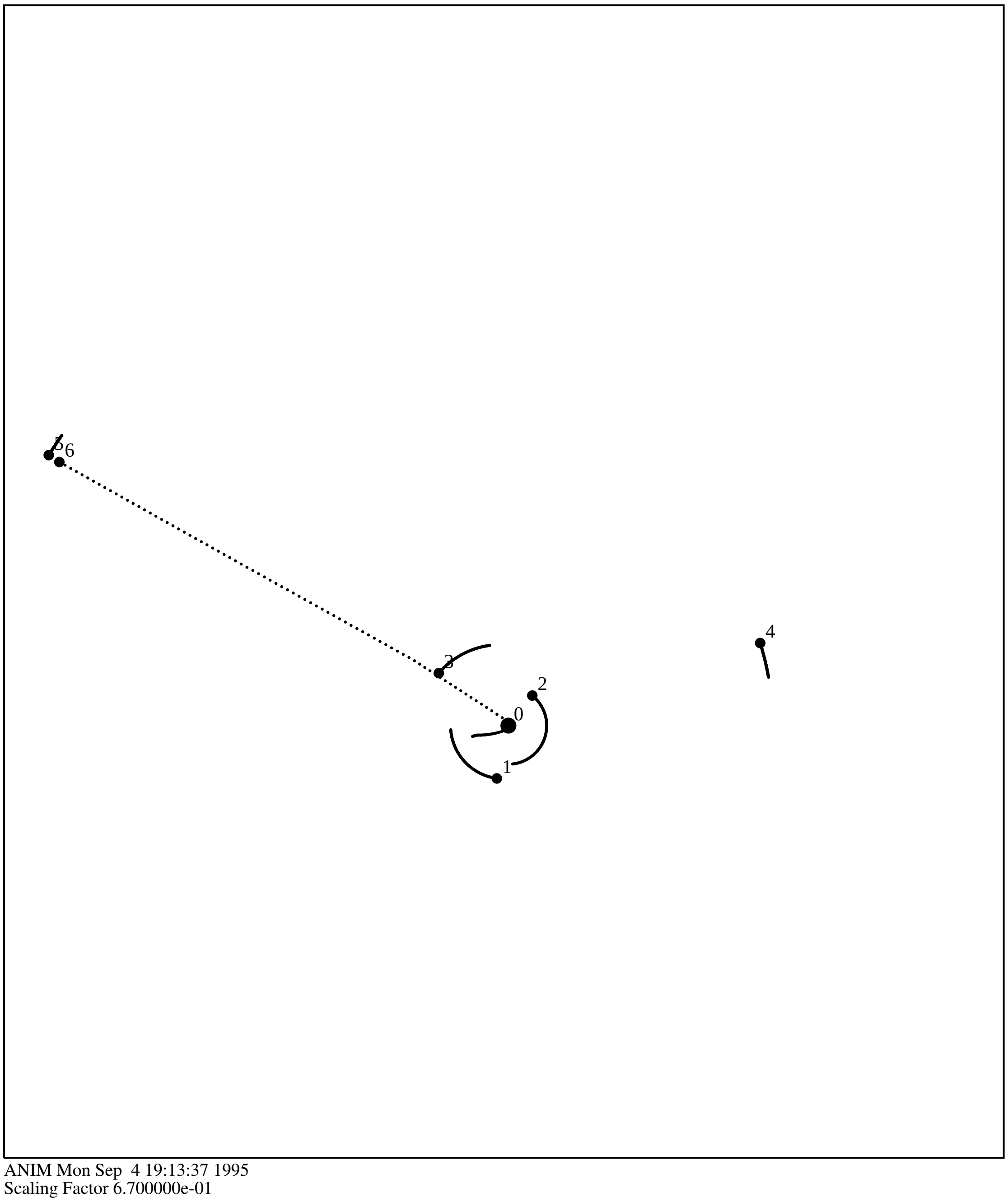

Figure B–2: Thrust Scaling Factor = 0.03

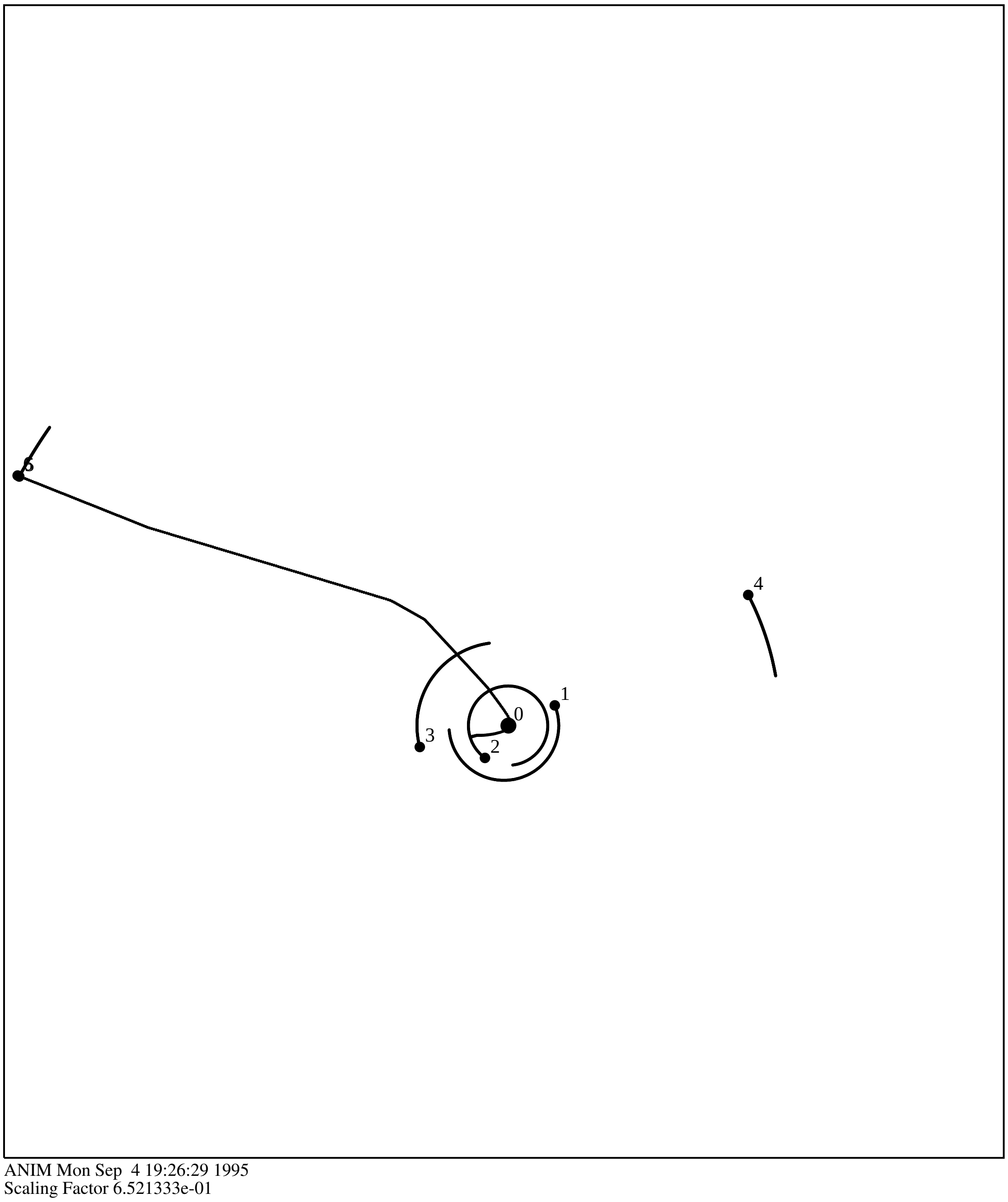

Figure B–3: Thrust Scaling Factor = 0.01

Appendix C

Realistic Problem Results

KEY

0: The Sun

1: The Earth

2: Venus

3: Mars

4: Jupiter

5: Saturn

6: The Probe

Thrust Scaling Factor = 0.005

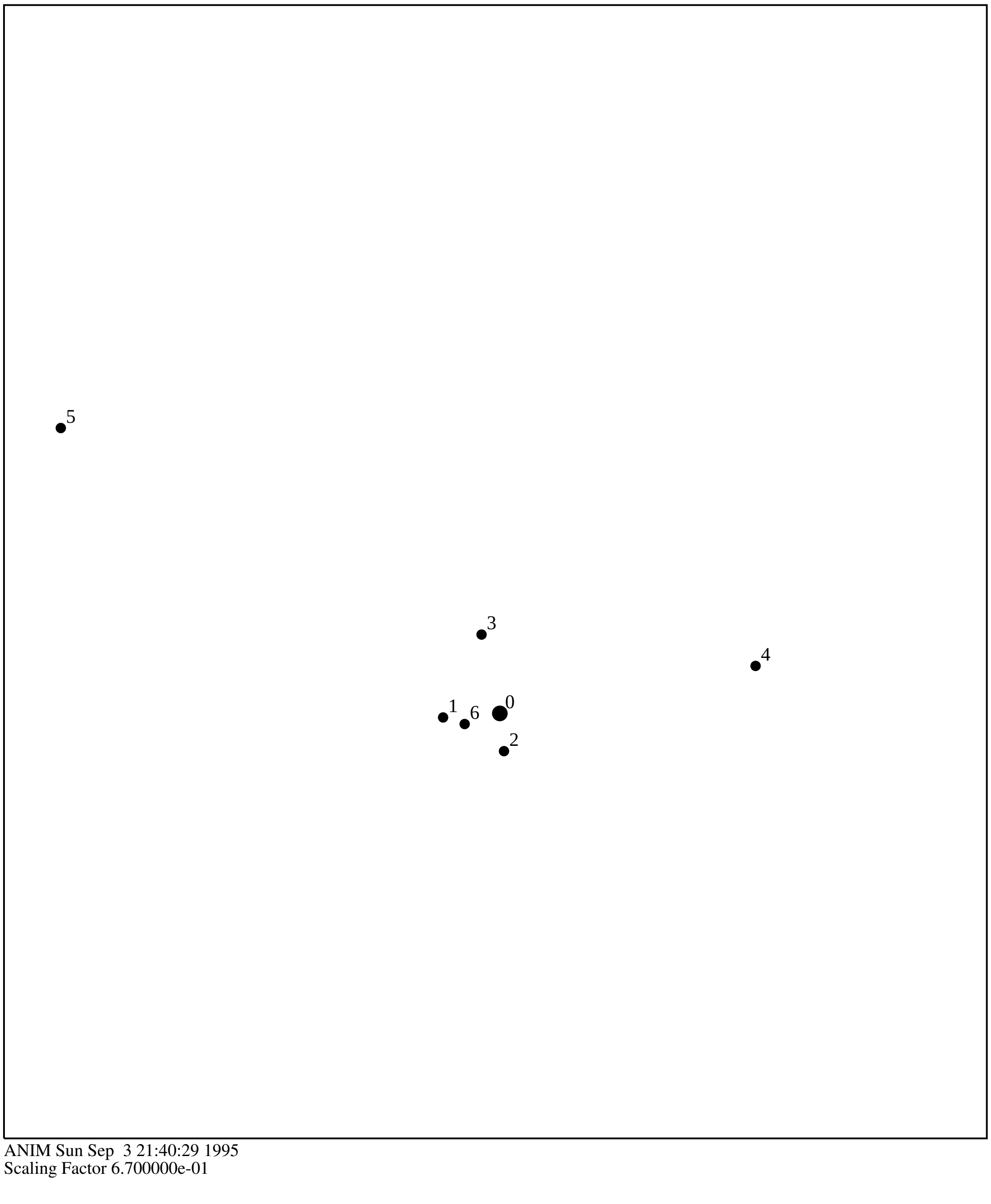

Figure C–1: Starting Positions of Planets and Probe

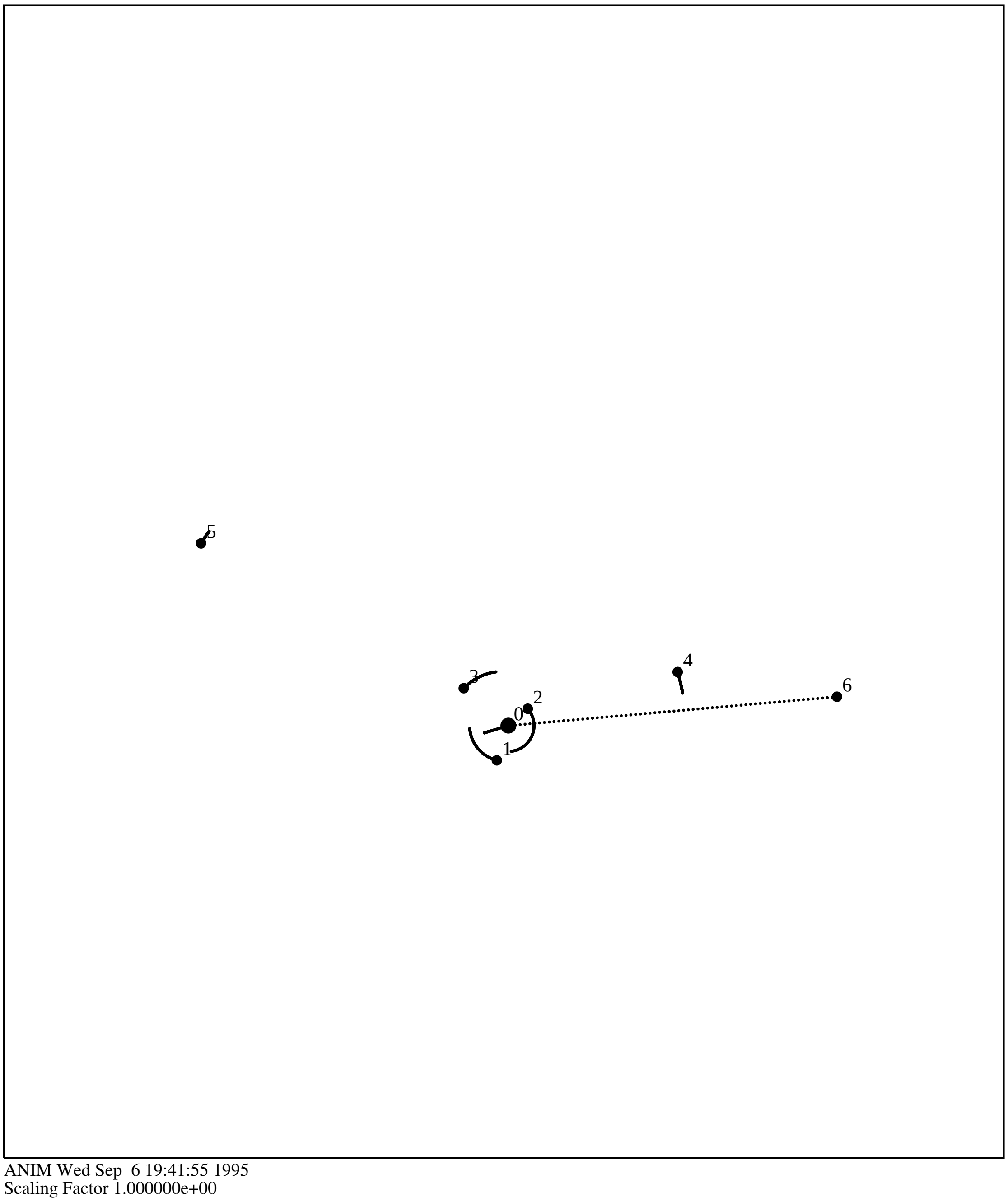

Figure C–2: Initial Population - Chromosome 3

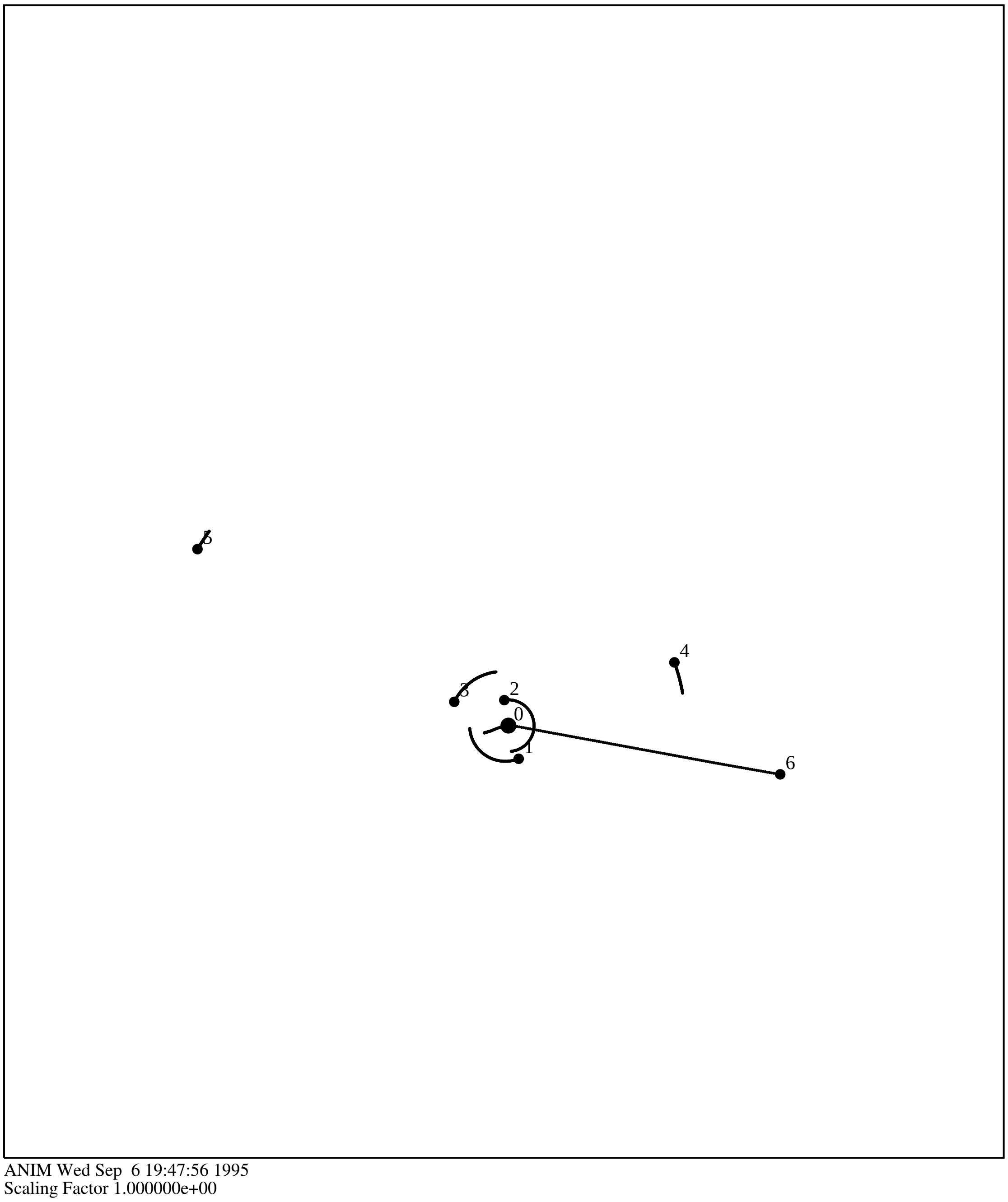

Figure C–3: Initial Population - Chromosome 5

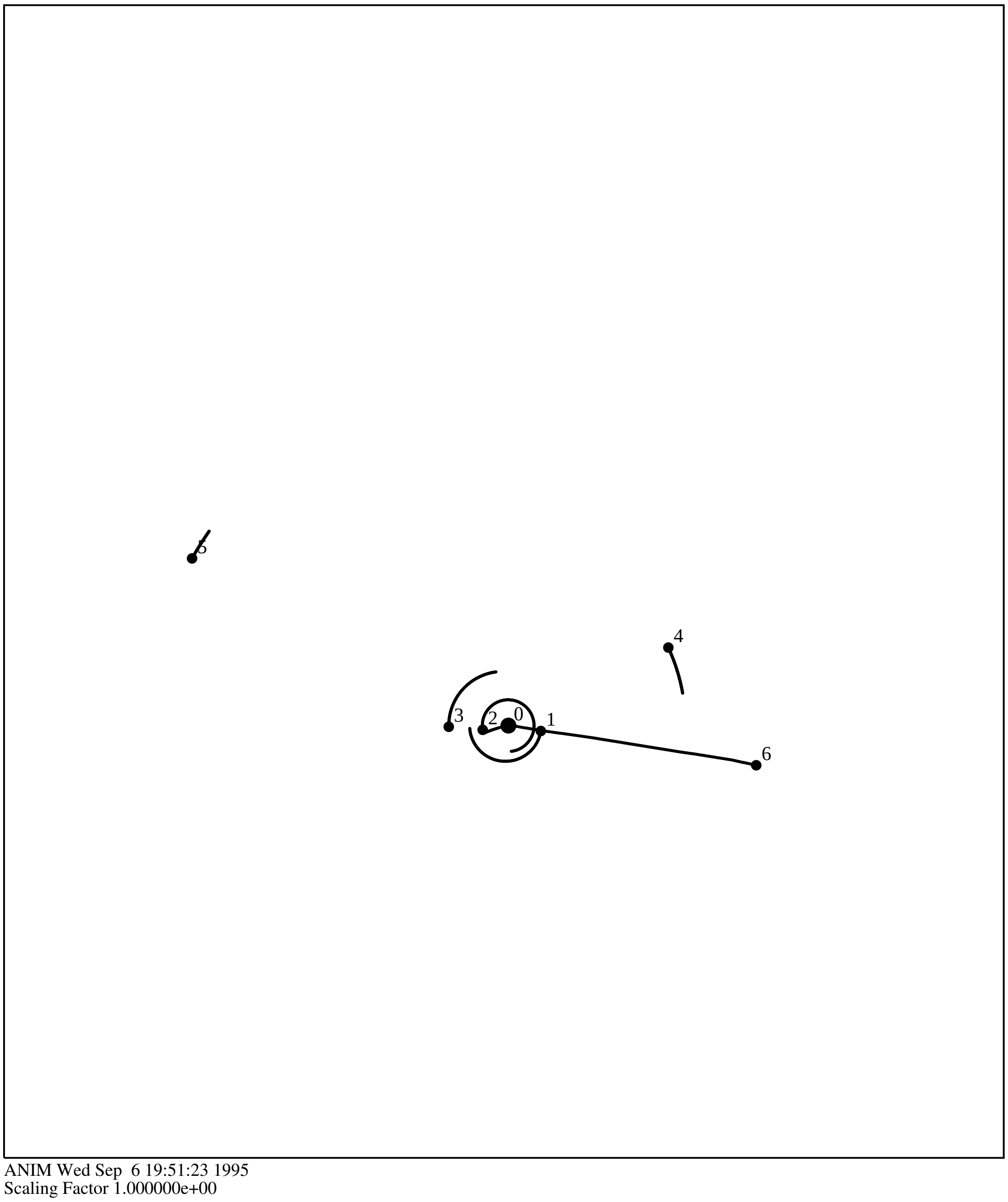

Figure C–4: Initial Population - Chromosome 7

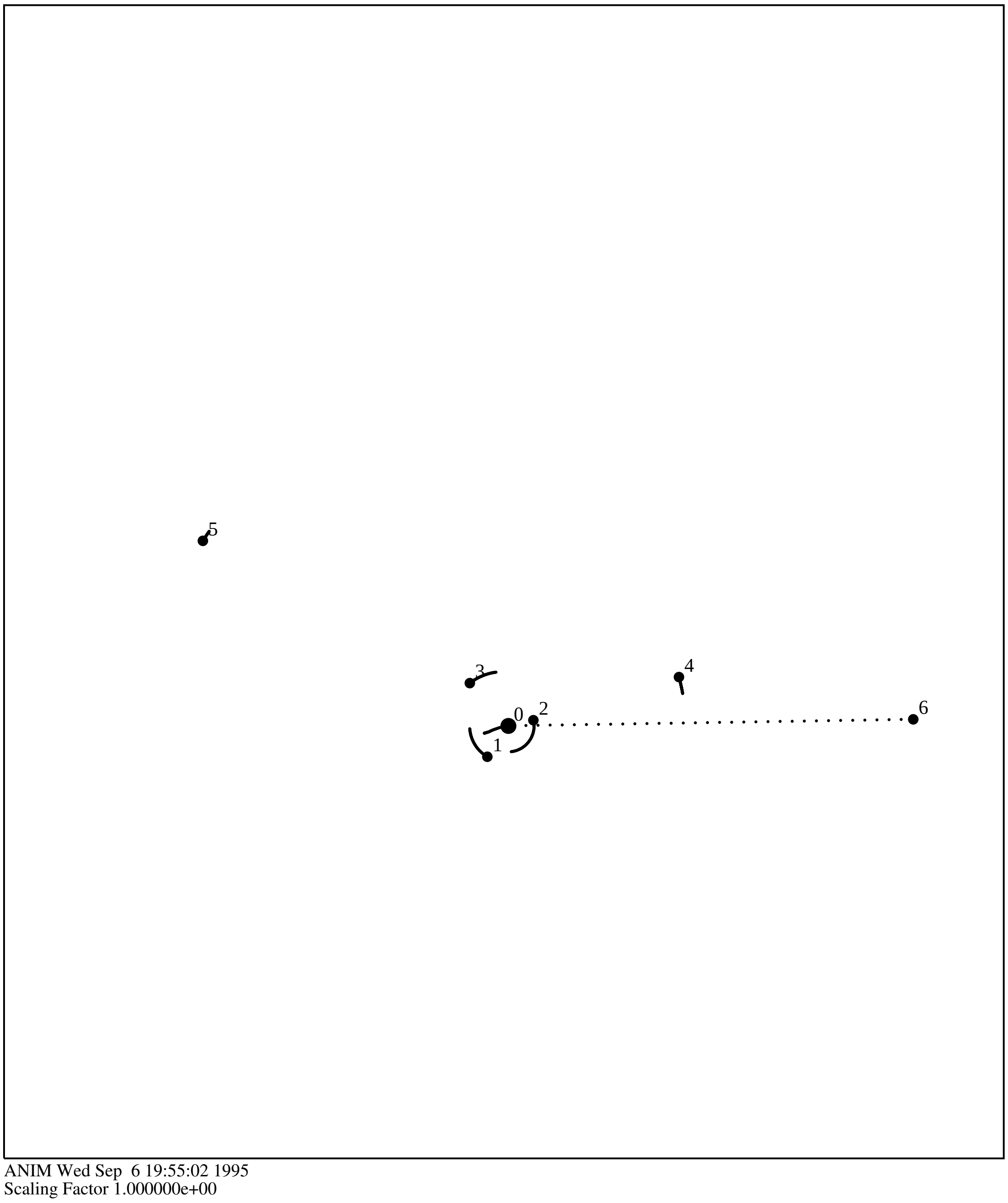

Figure C–5: Initial Population - Chromosome 9

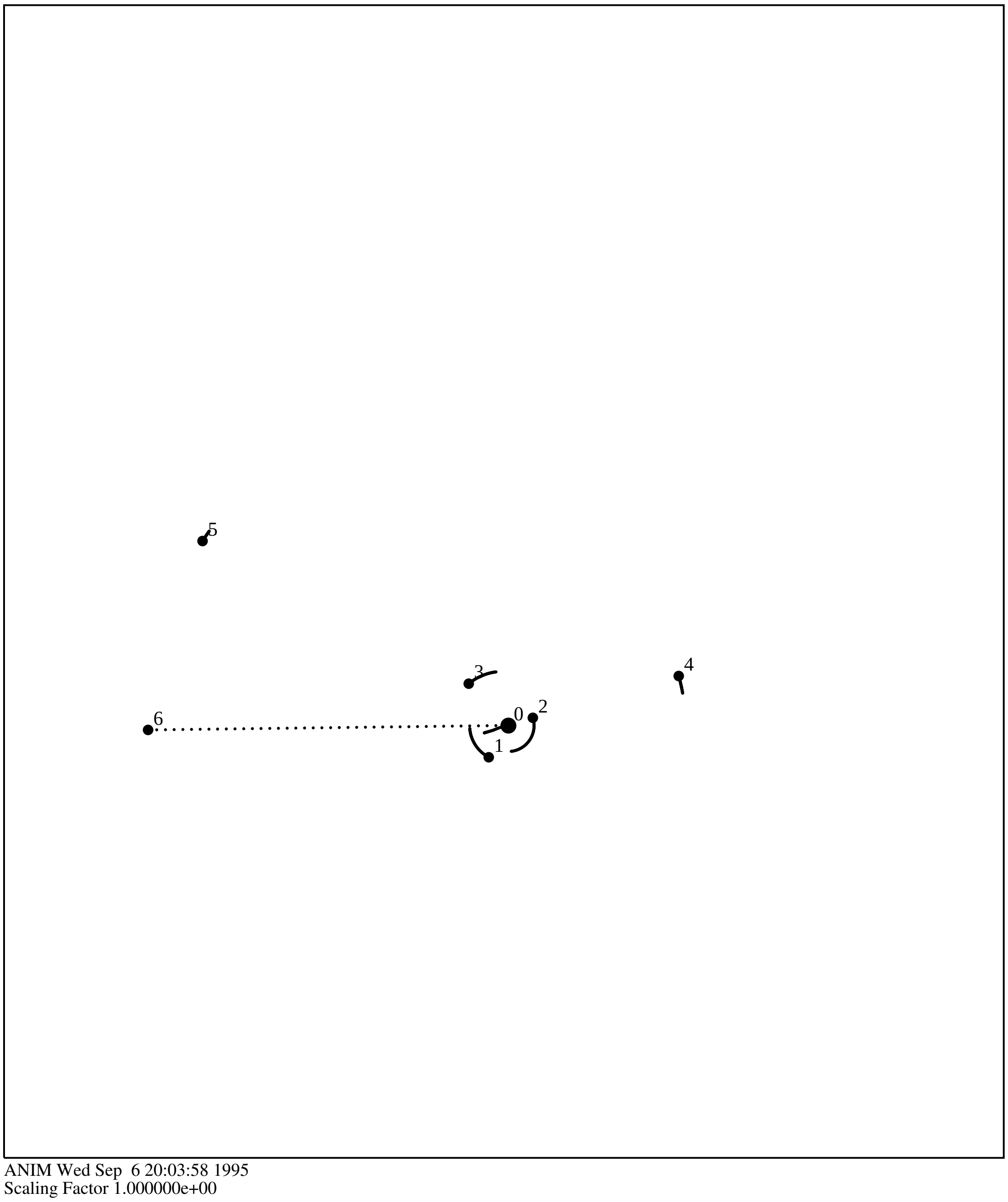

Figure C–6: Initial Population - Chromosome 15

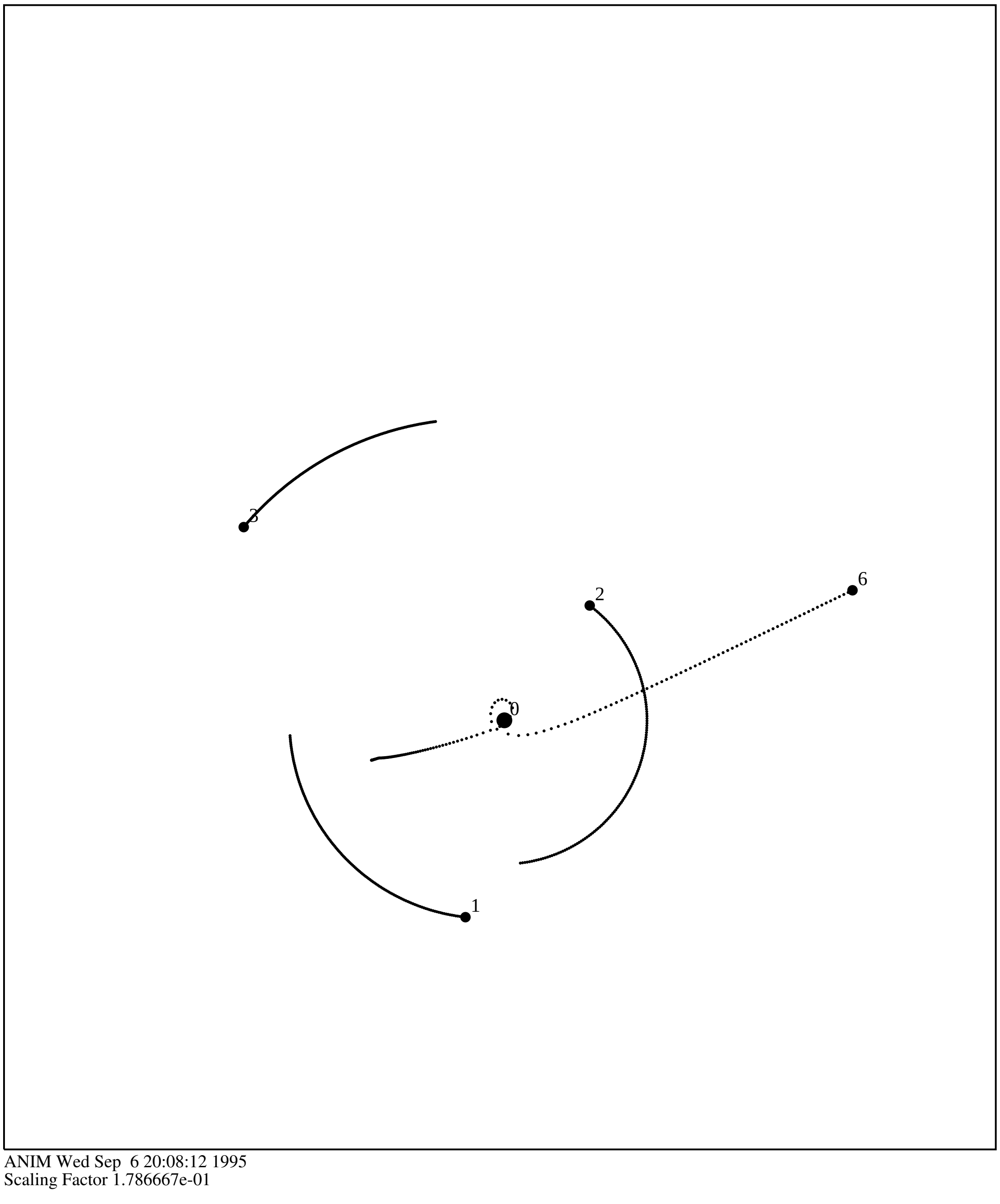

Figure C–7: Initial Population - Chromosome 16

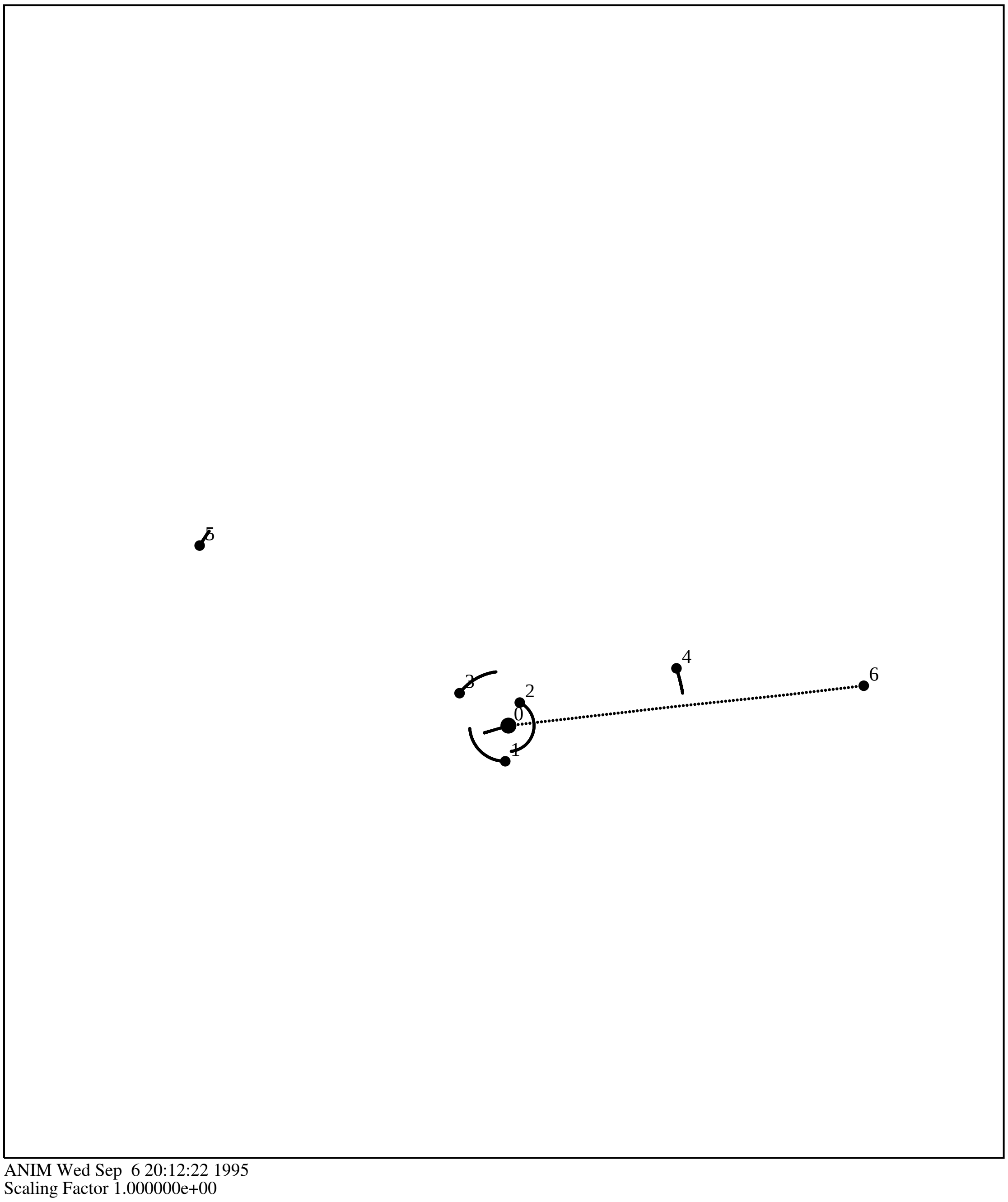

Figure C–8: Initial Population - Chromosome 19

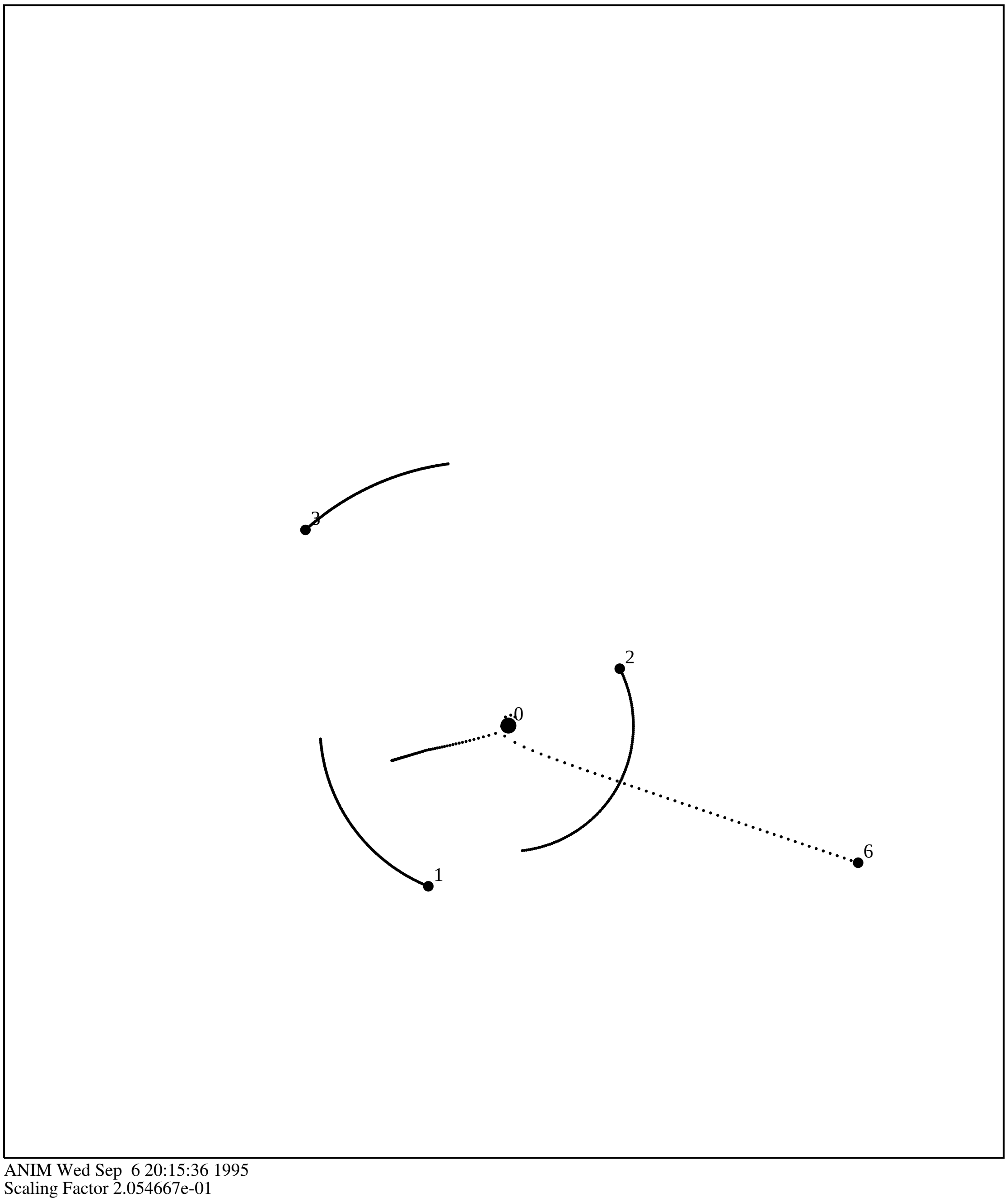

Figure C–9: Initial Population - Chromosome 21

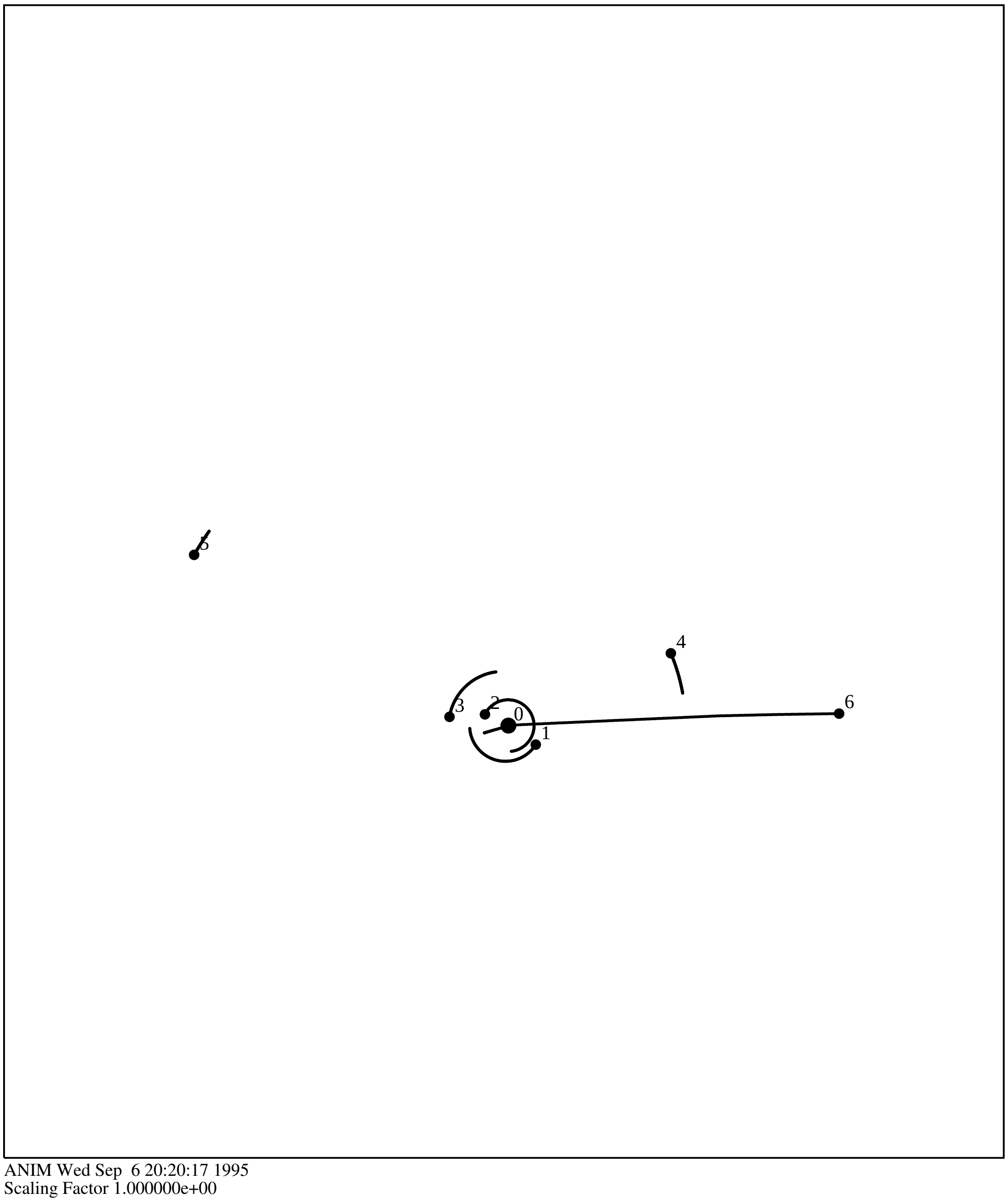

Figure C–10: Initial Population - Chromosome 23

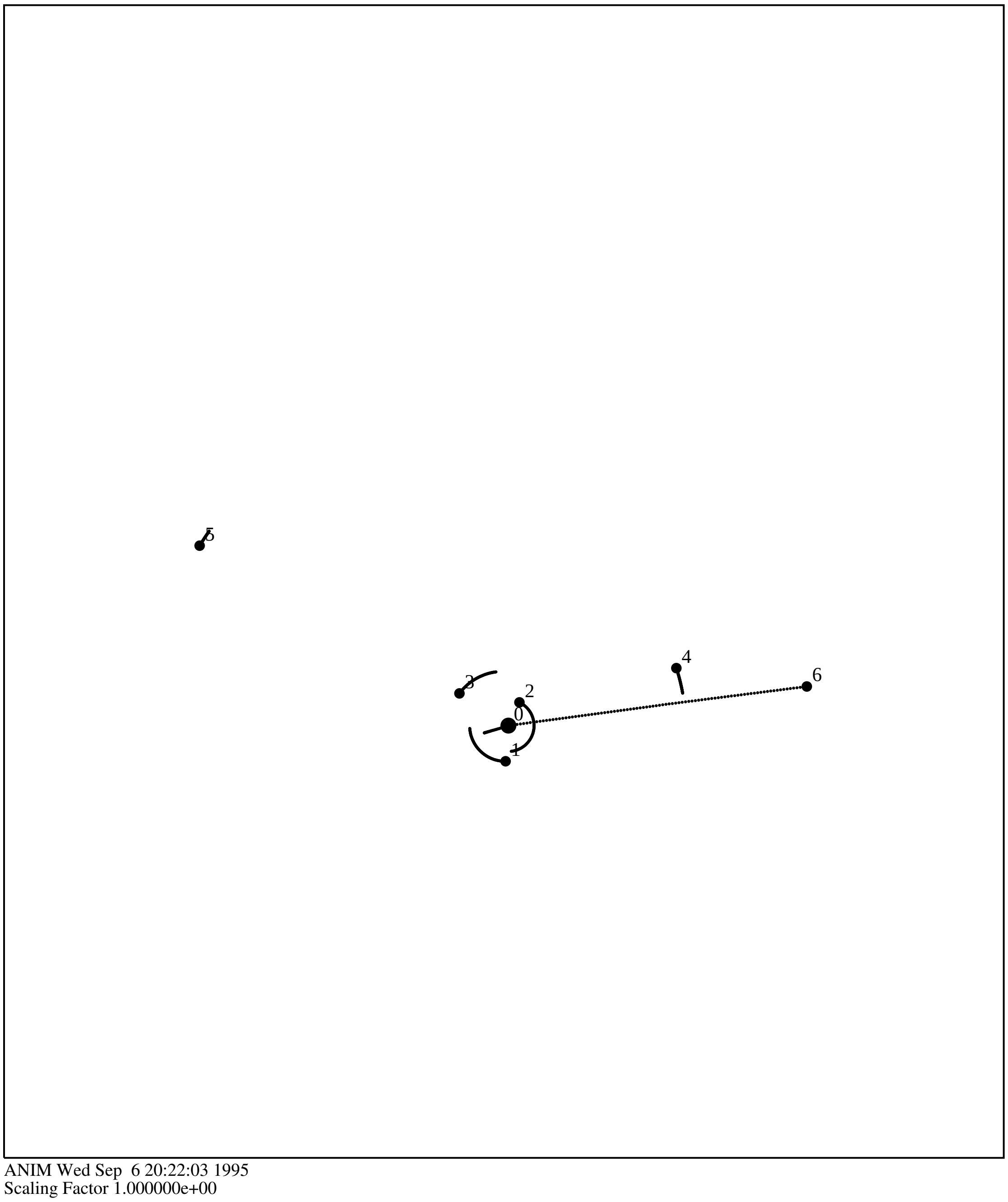

Figure C–11: Initial Population - Chromosome 24

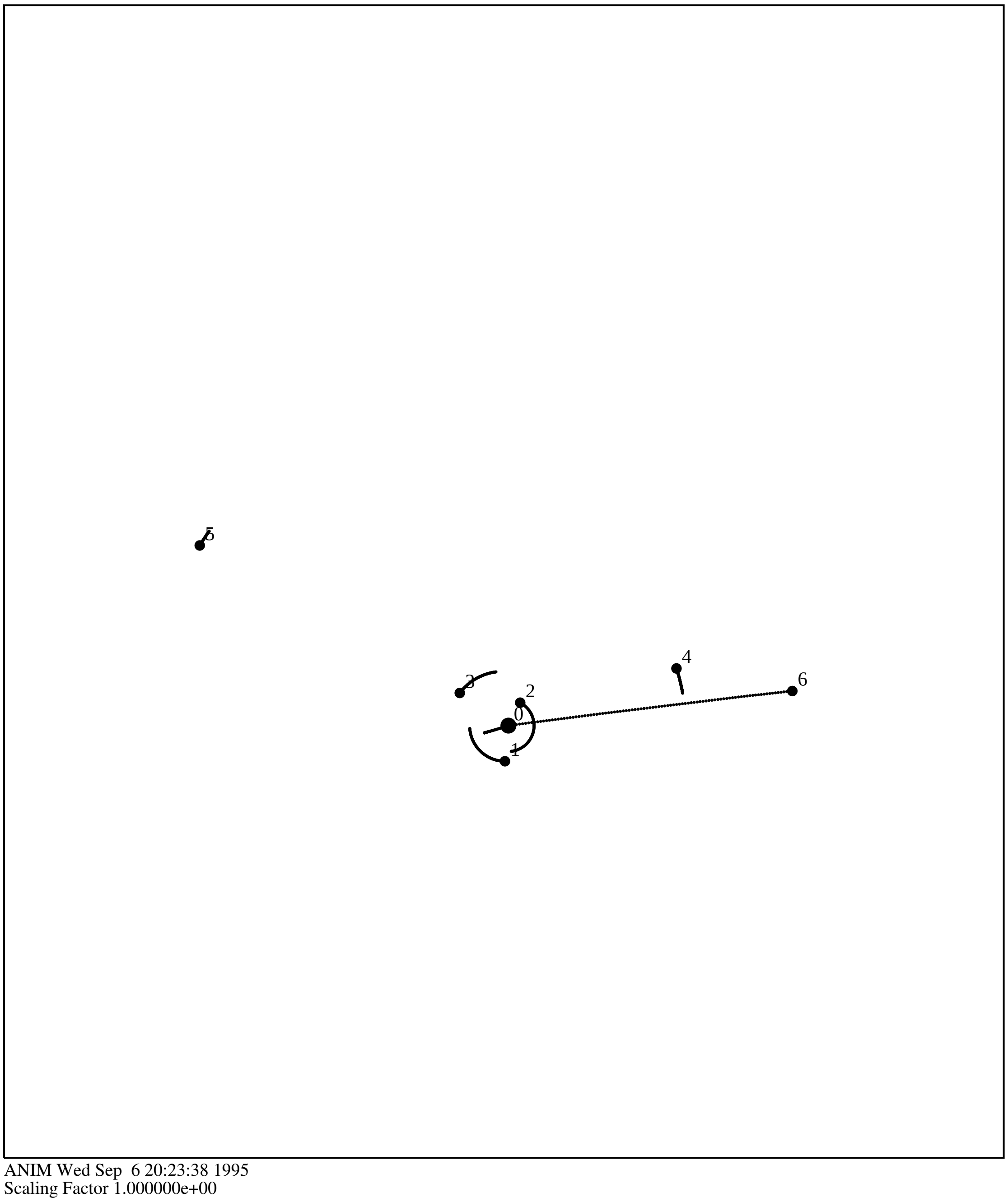

Figure C–12: Initial Population - Chromosome 25

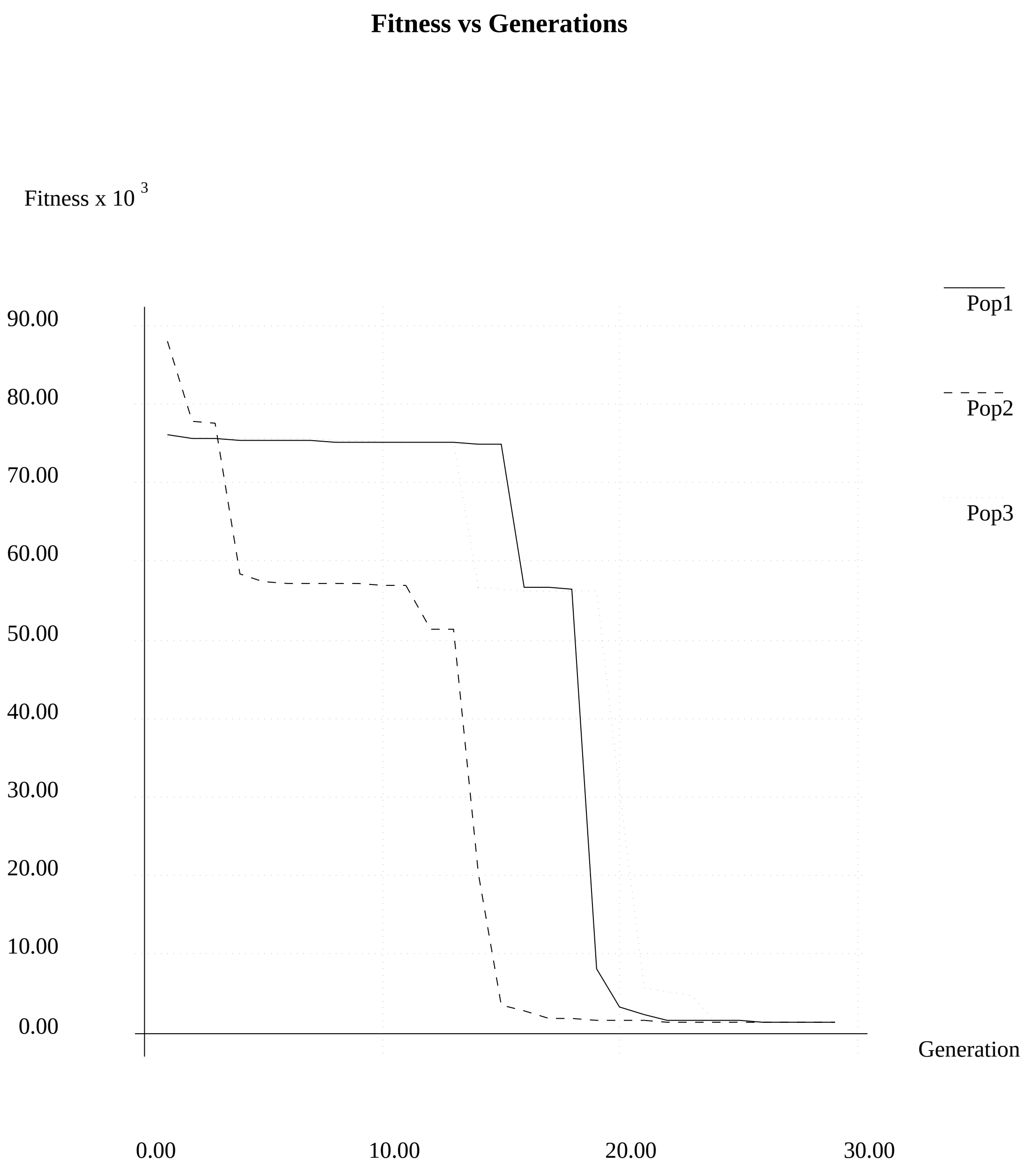

Figure C–13: Convergence of Best Solutions

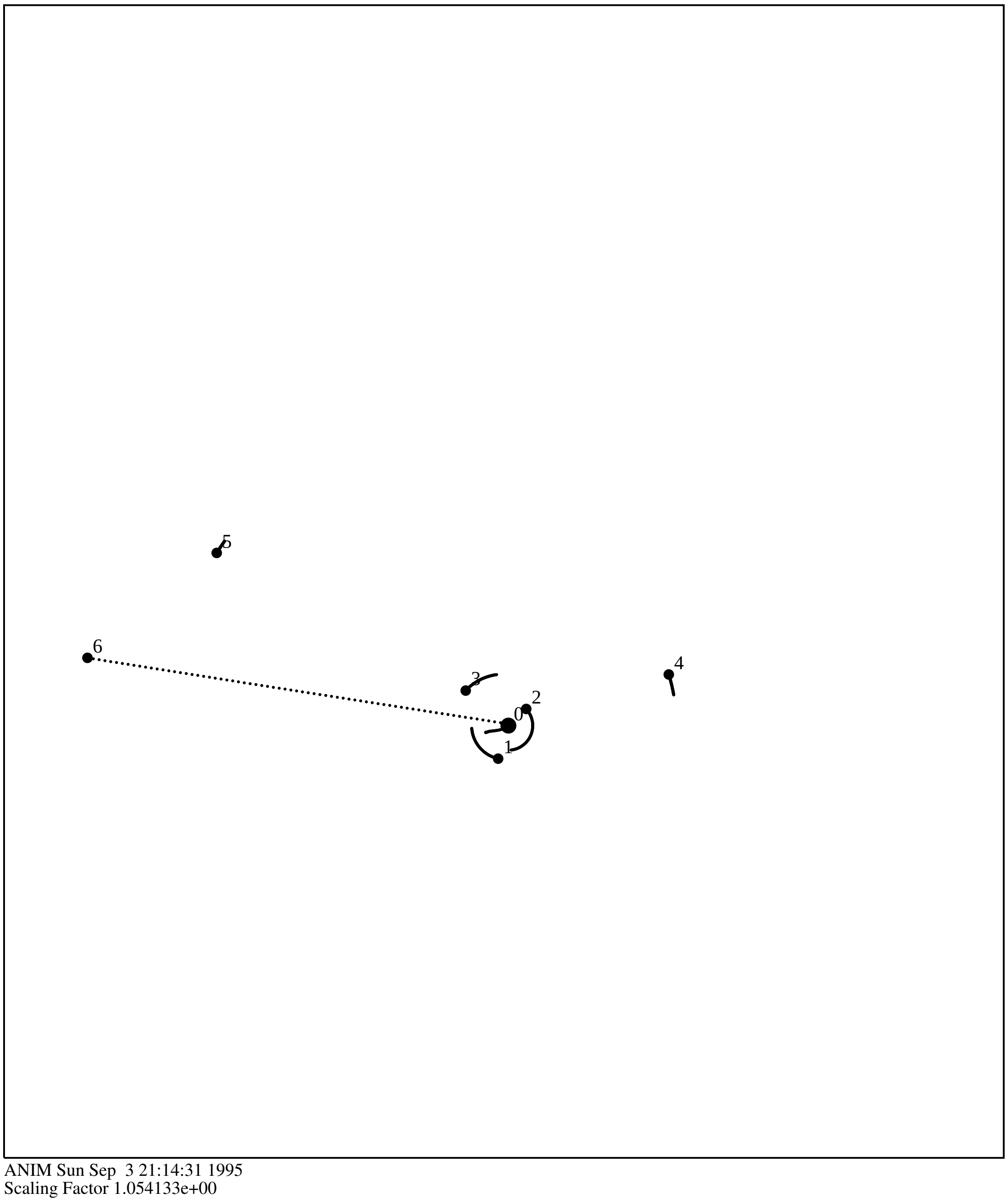

Figure C–14: Generation 11 - Best Solution

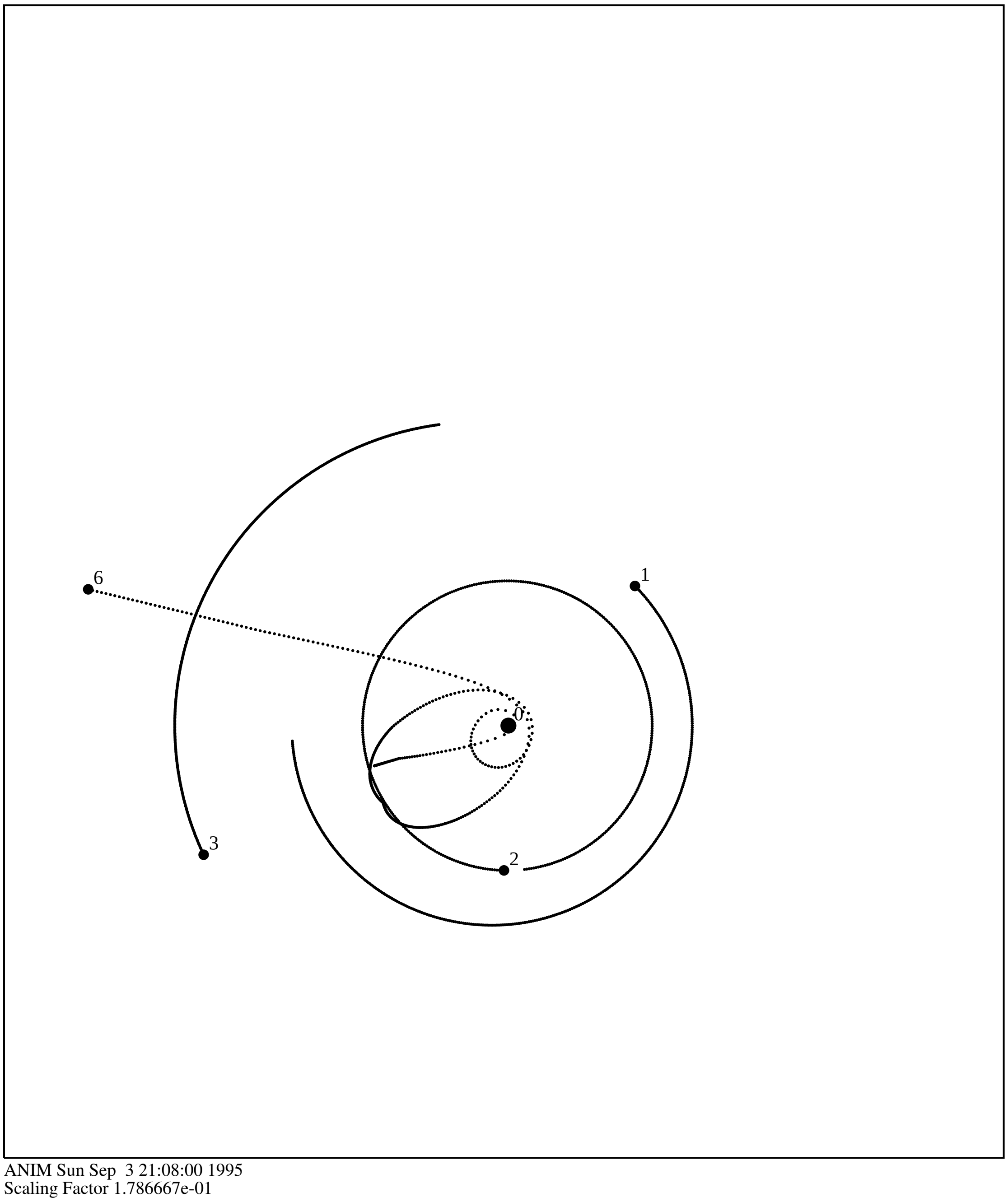

Figure C–15: Generation 19 - Best Solution

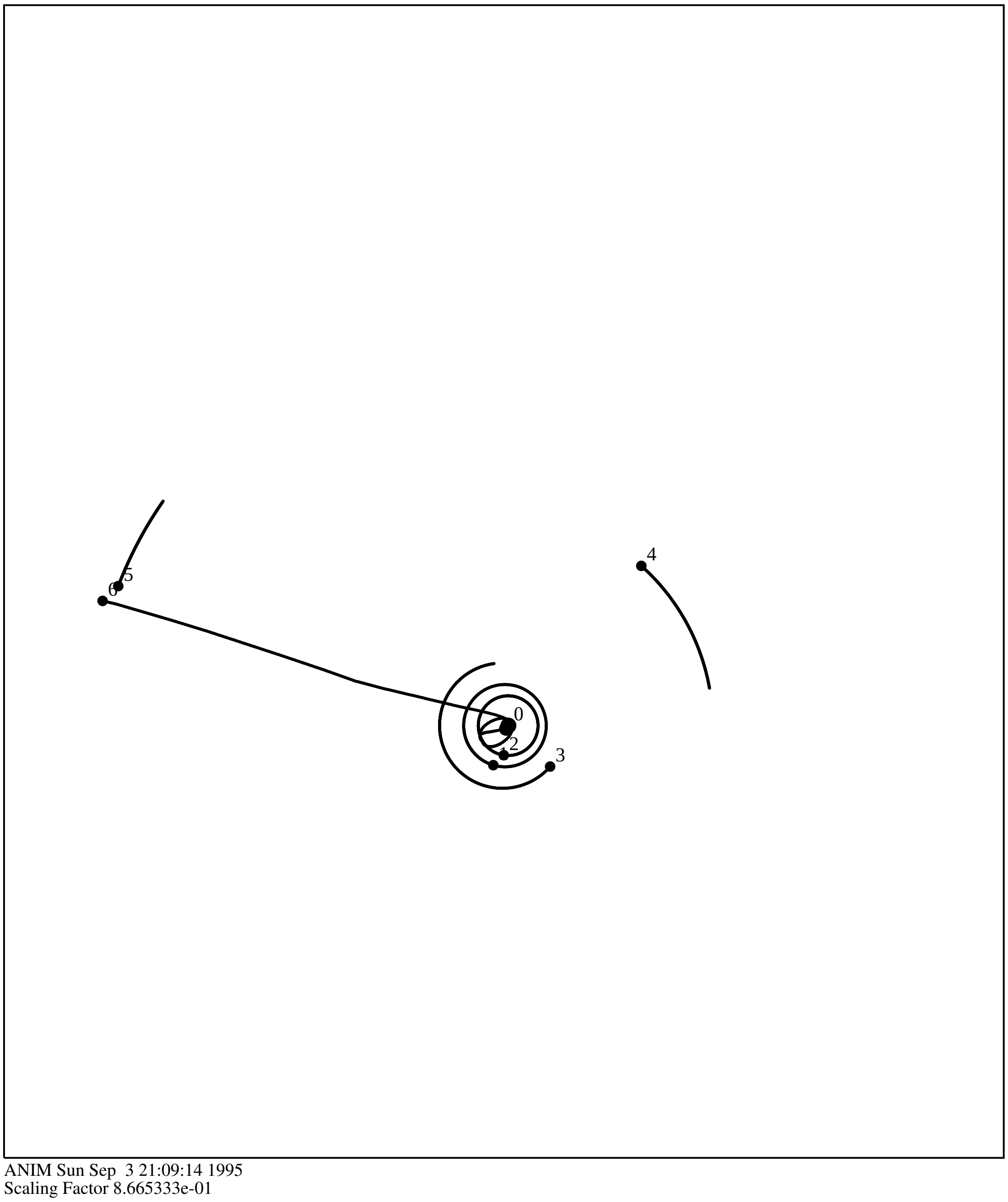

Figure C–16: Generation 19 - Best Solution

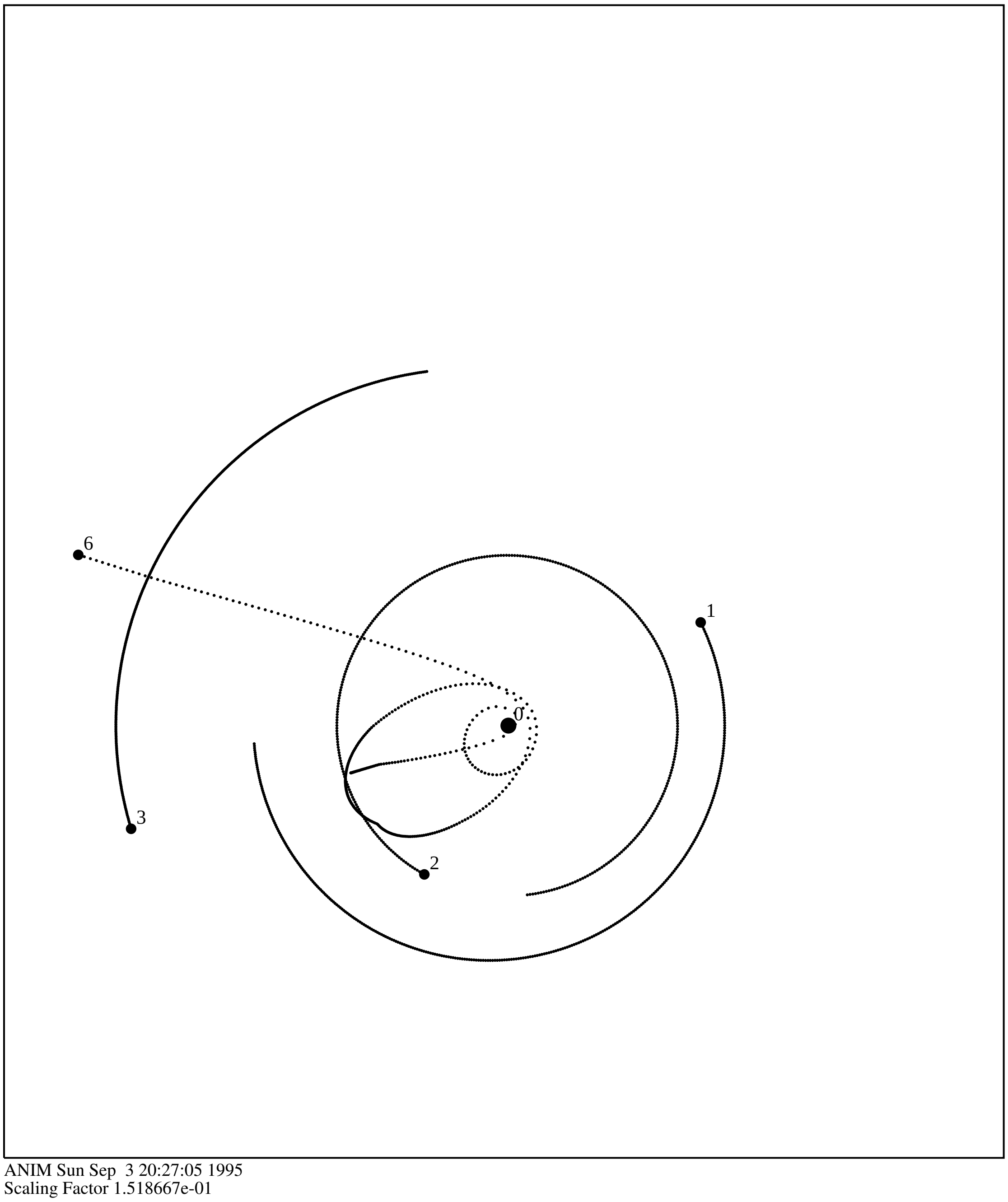

Figure C–17: Generation 30 - Best Solution

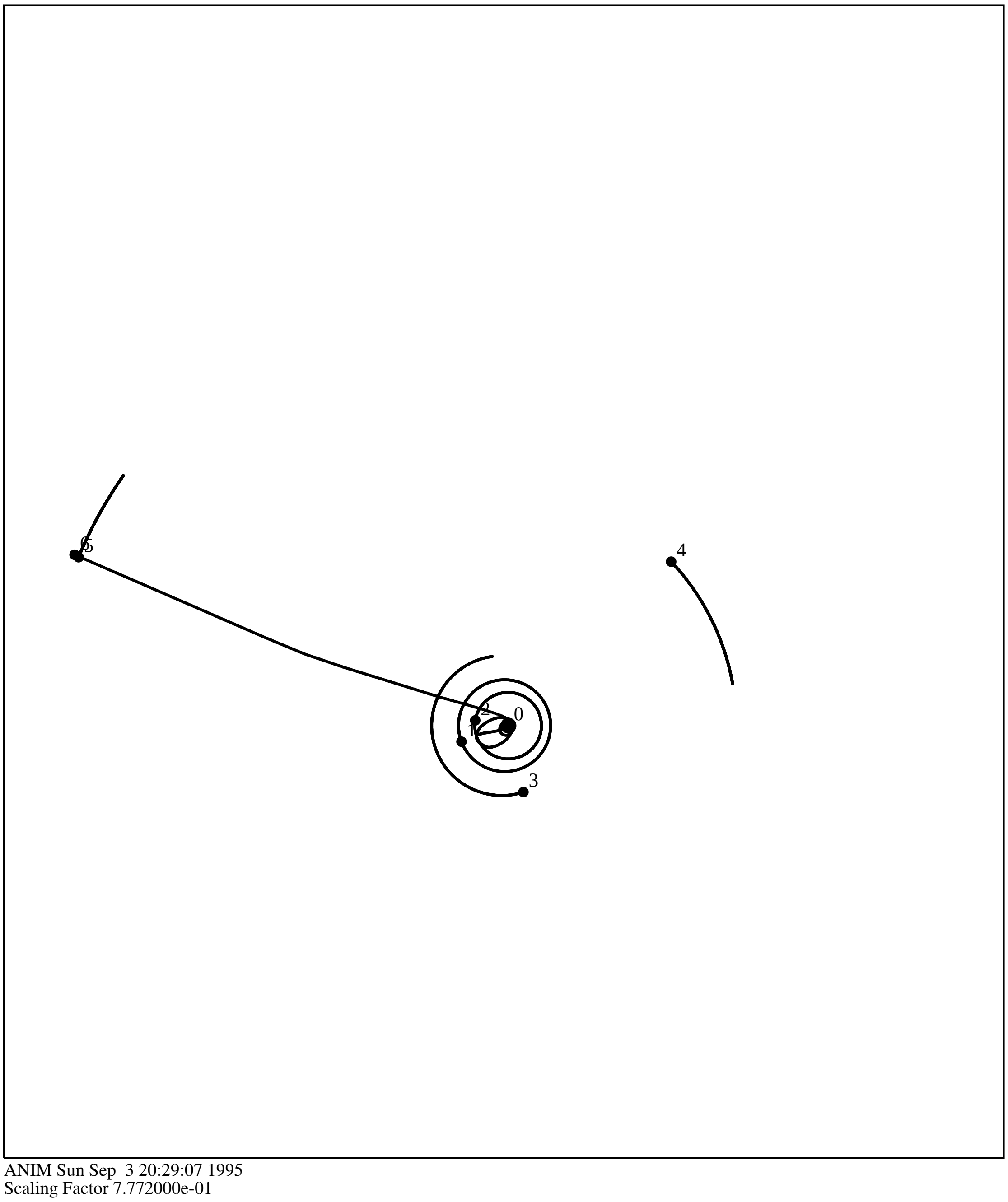

Figure C–18: Generation 30 - Best Solution

Appendix D

Convergence Curves

Steady State

One Point Crossover

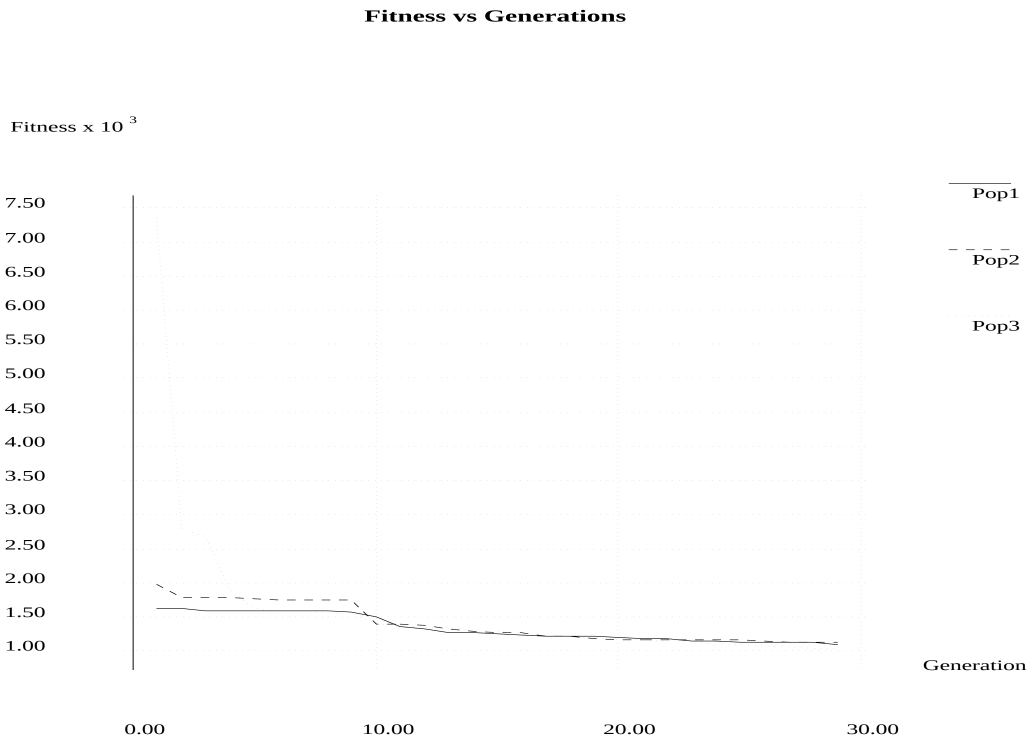

Figure D–1: Run 1 - Steady State, 1pt Cross.

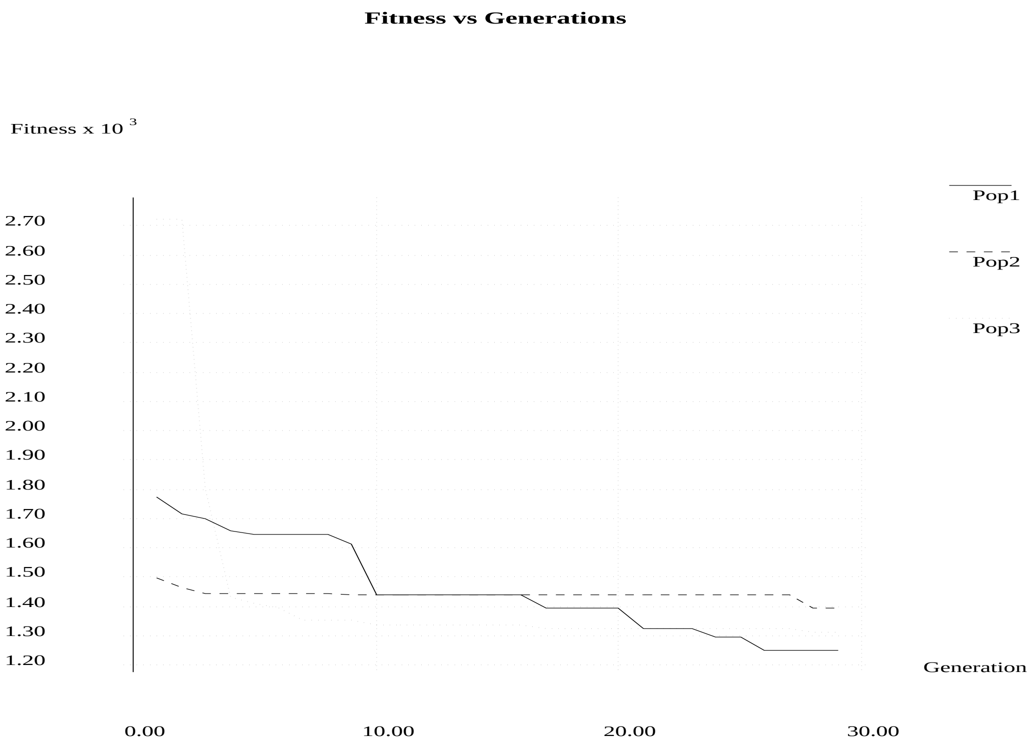

Figure D–2: Run2 - Steady State, 1pt Cross.

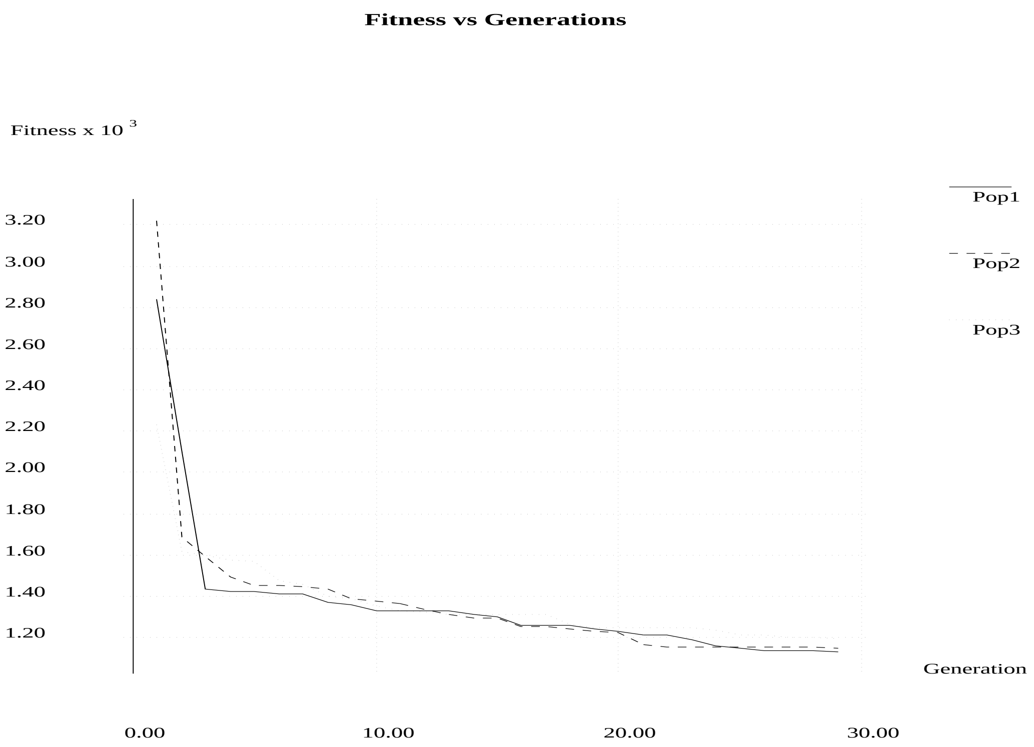

Figure D–3: Run 3 - Steady State, 1pt Cross.

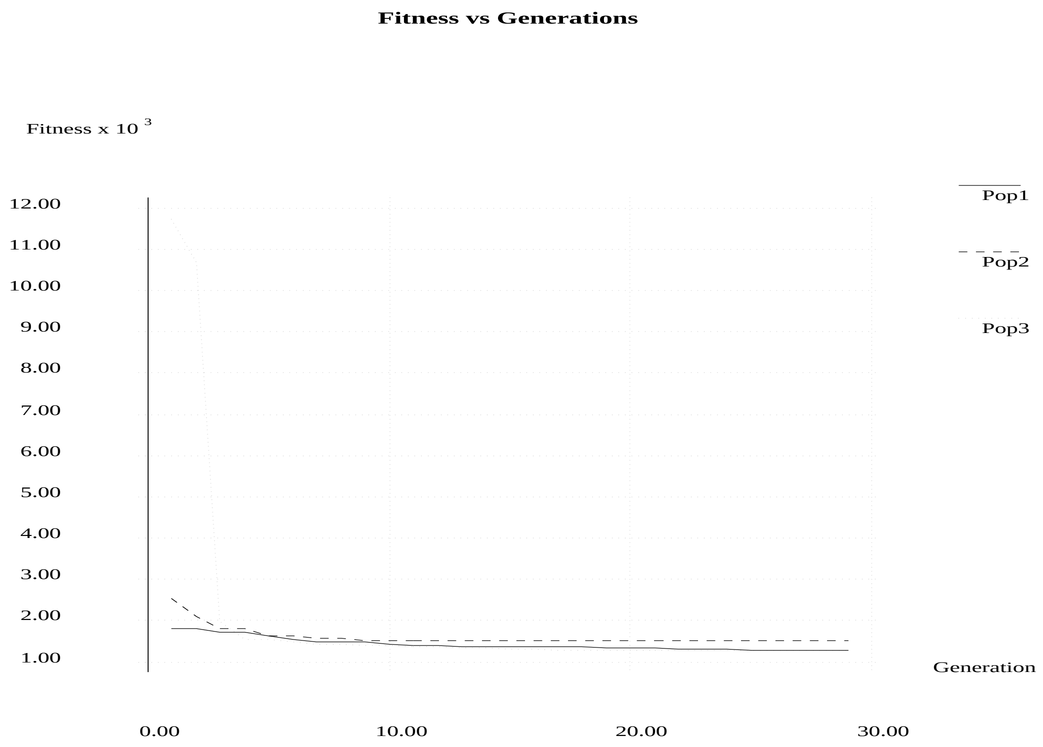

Figure D–4: Run 4 - Steady State, 1pt Cross.

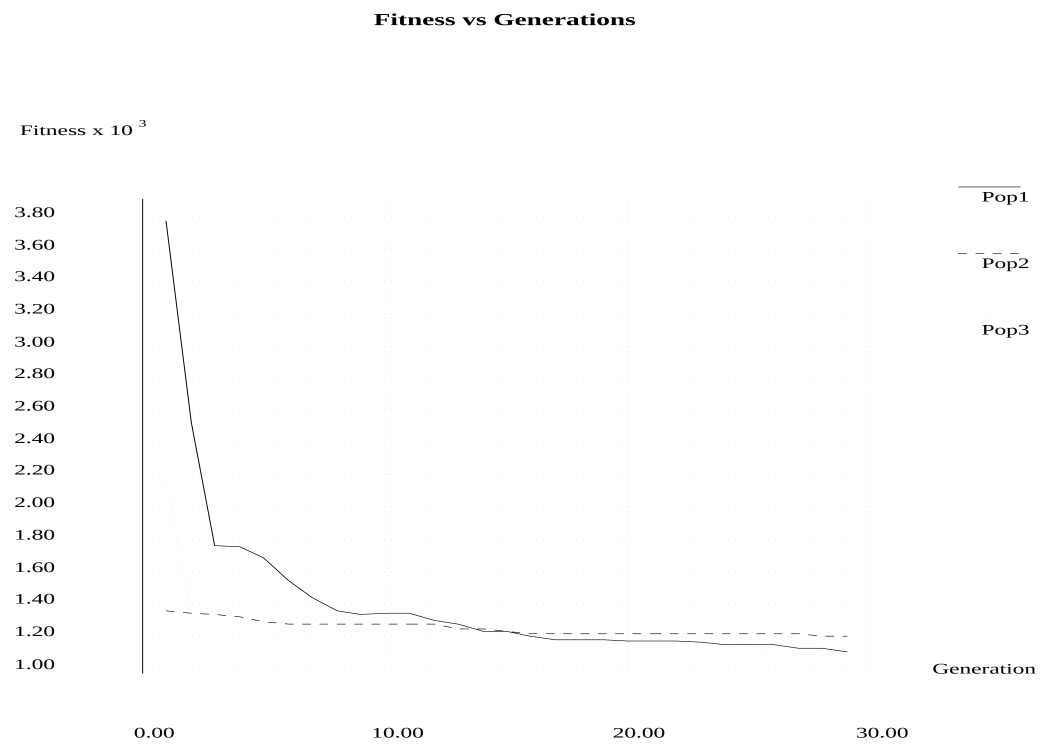

Figure D–5: Run 5 - Steady State, 1pt Cross.

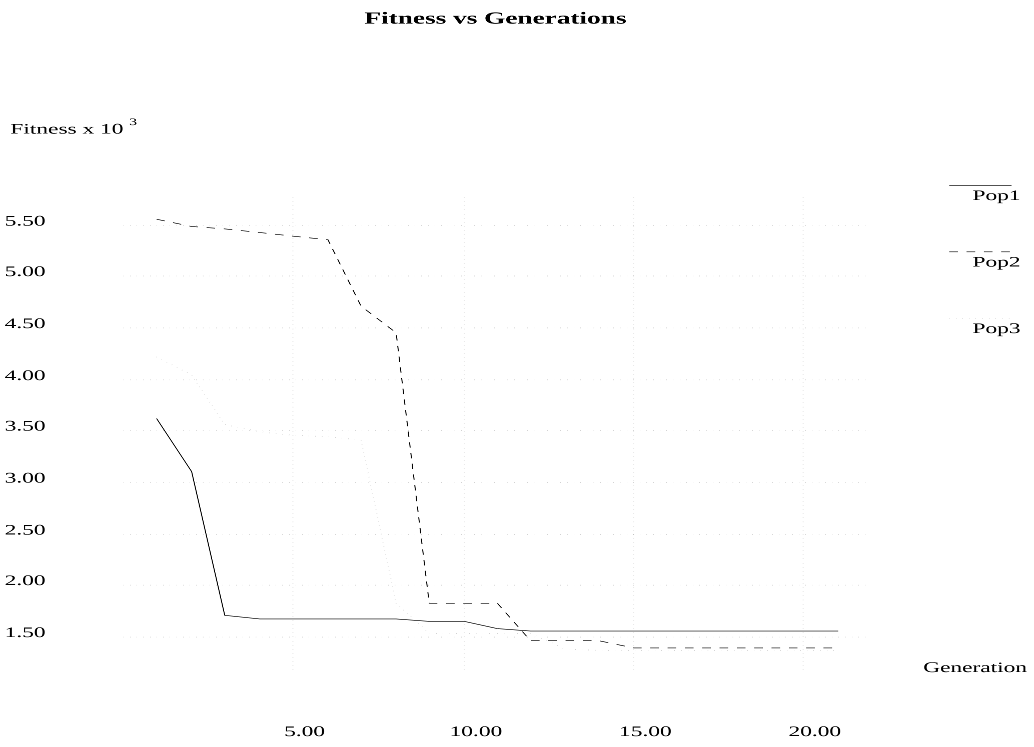

Figure D–6: Run 6 - Steady State, 1pt Cross.

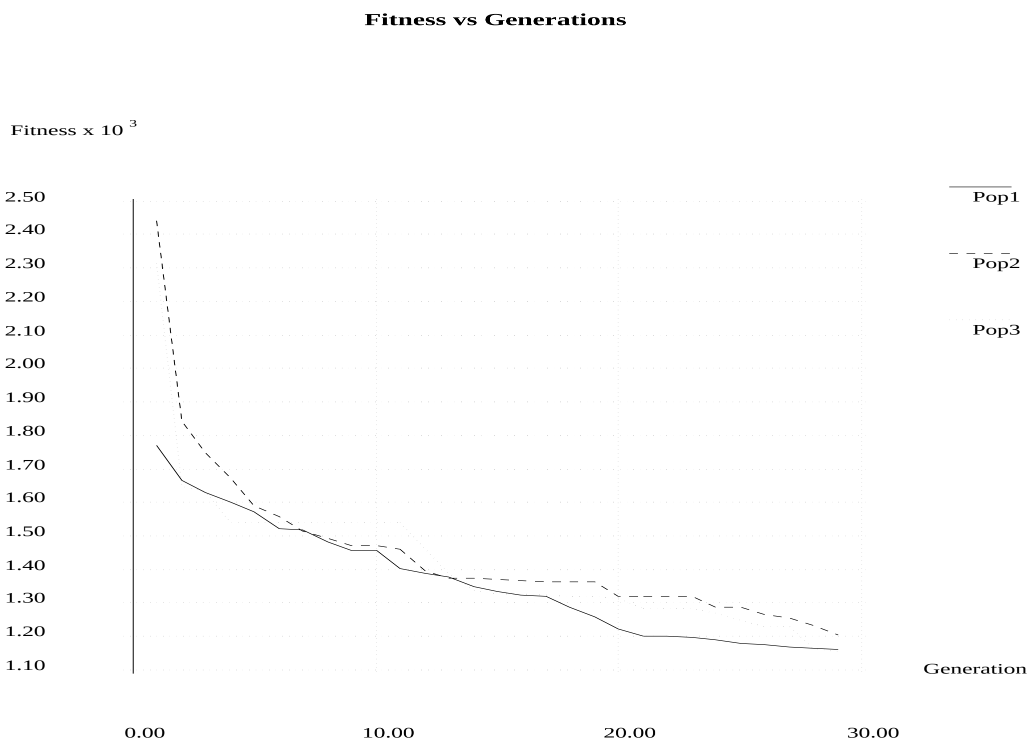

Figure D–7: Run 7 - Steady State, 1pt Cross.

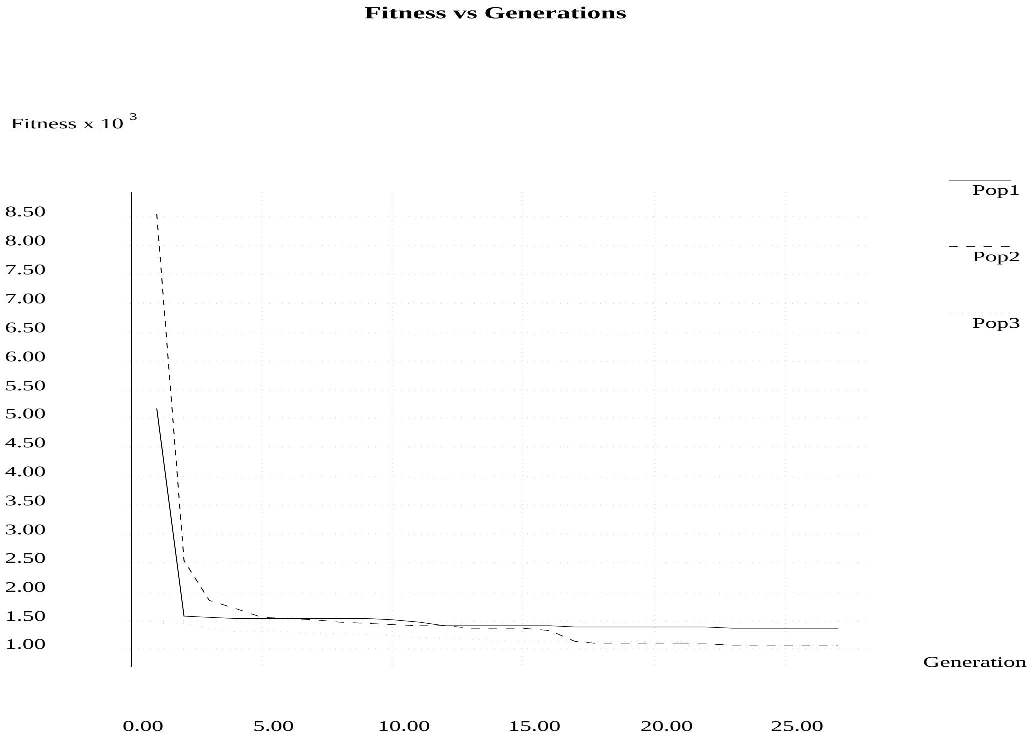

Figure D–8: Run 8 - Steady State, 1pt Cross.

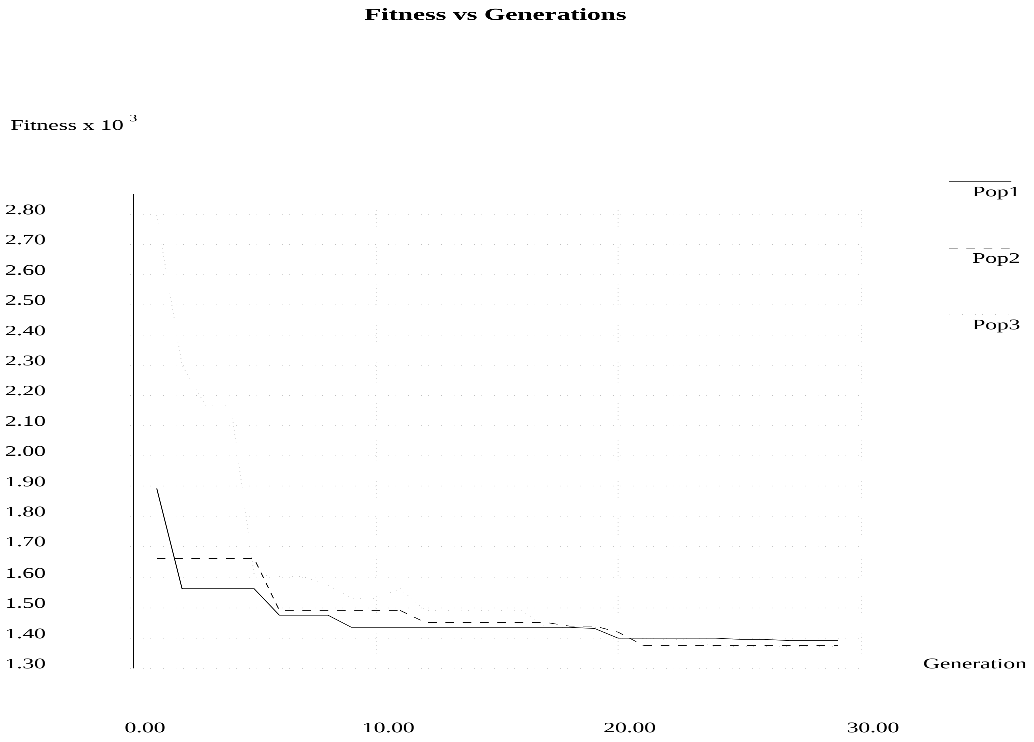

Figure D–9: Run 9 - Steady State, 1pt Cross.

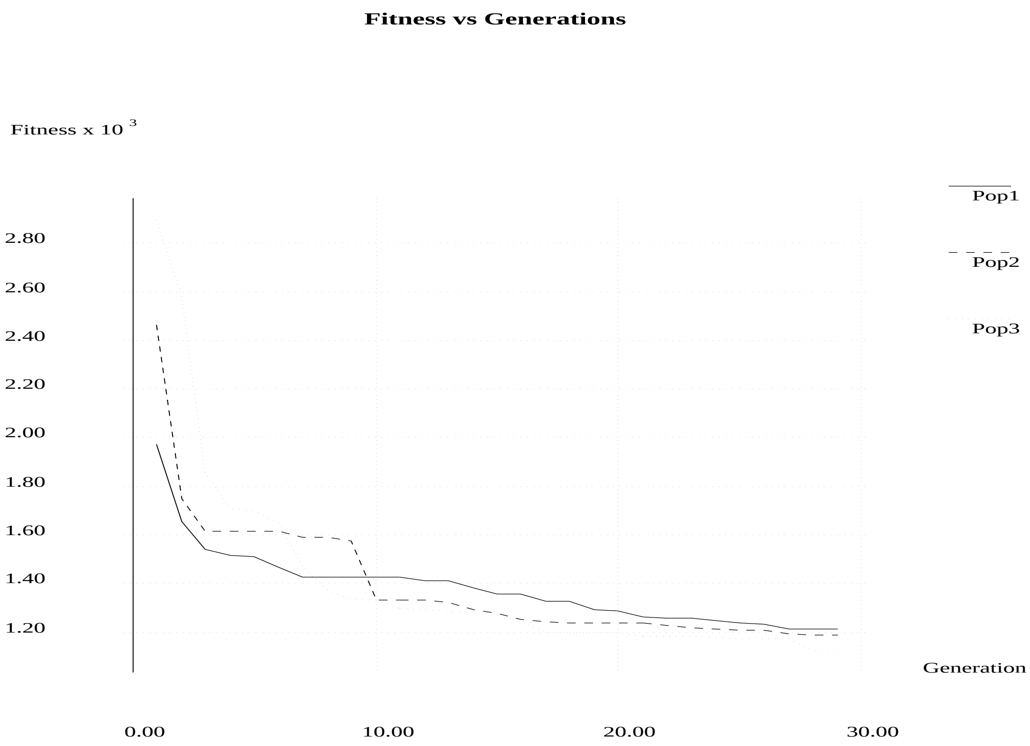

Figure D–10: Run 10 - Steady State, 1pt Cross.

Appendix E

Convergence Curves

Generation-Based

One Point Crossover

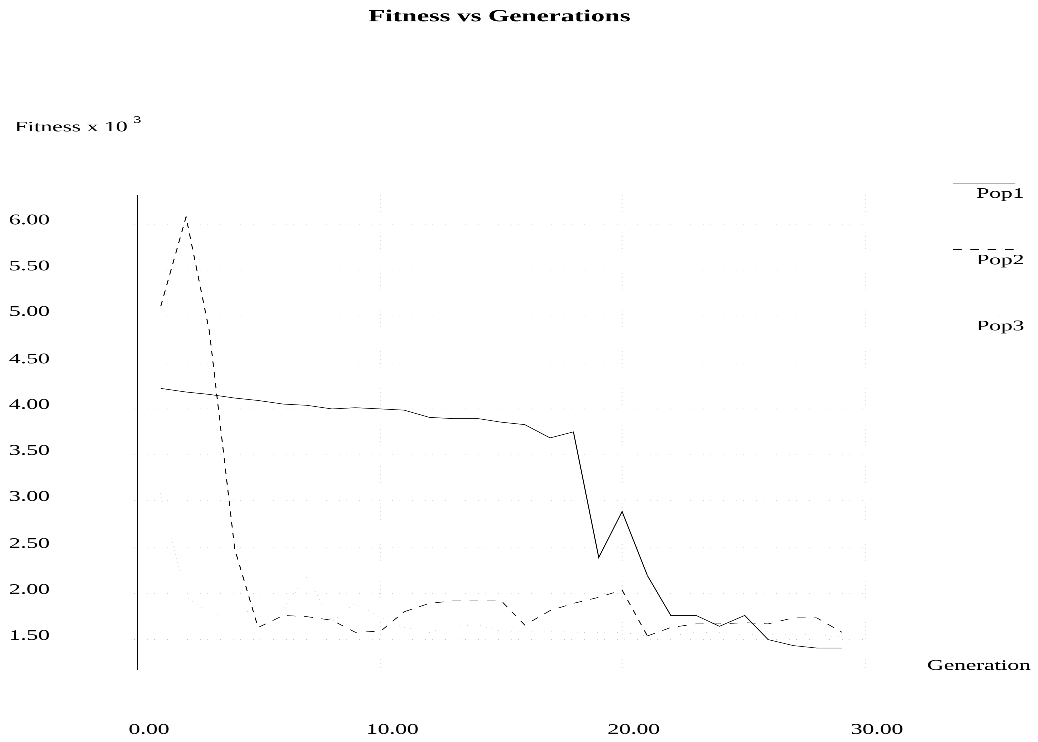

Figure E–1: Run 1 - Generation-Based, 1pt Cross.

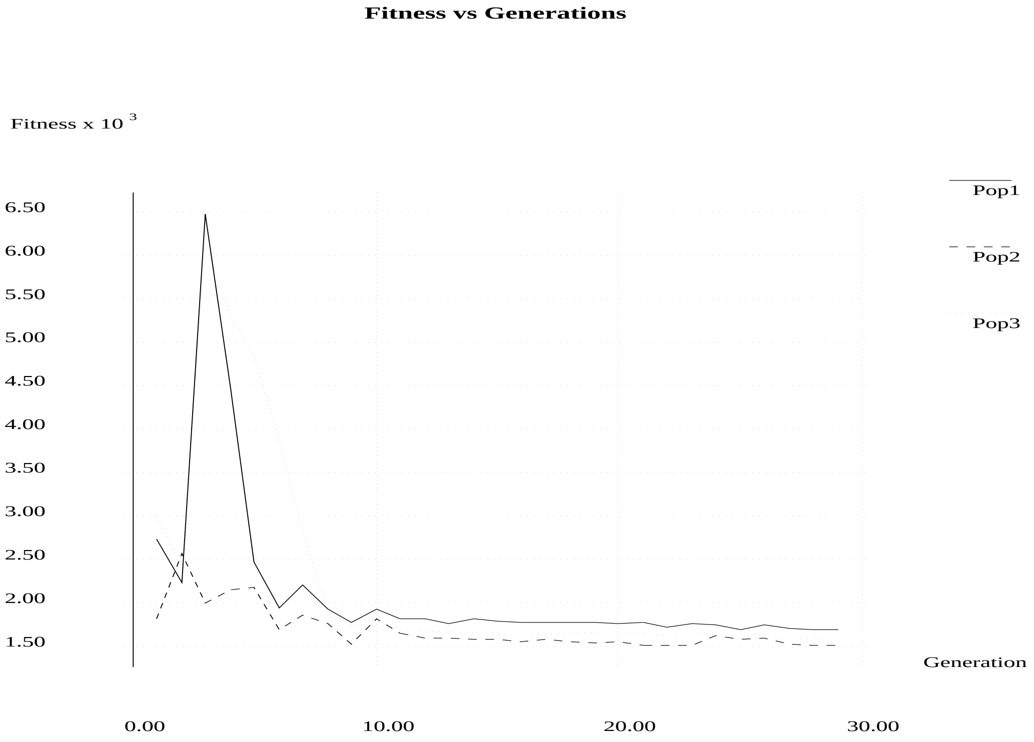

Figure E–2: Run2 - Generation-Based, 1pt Cross.

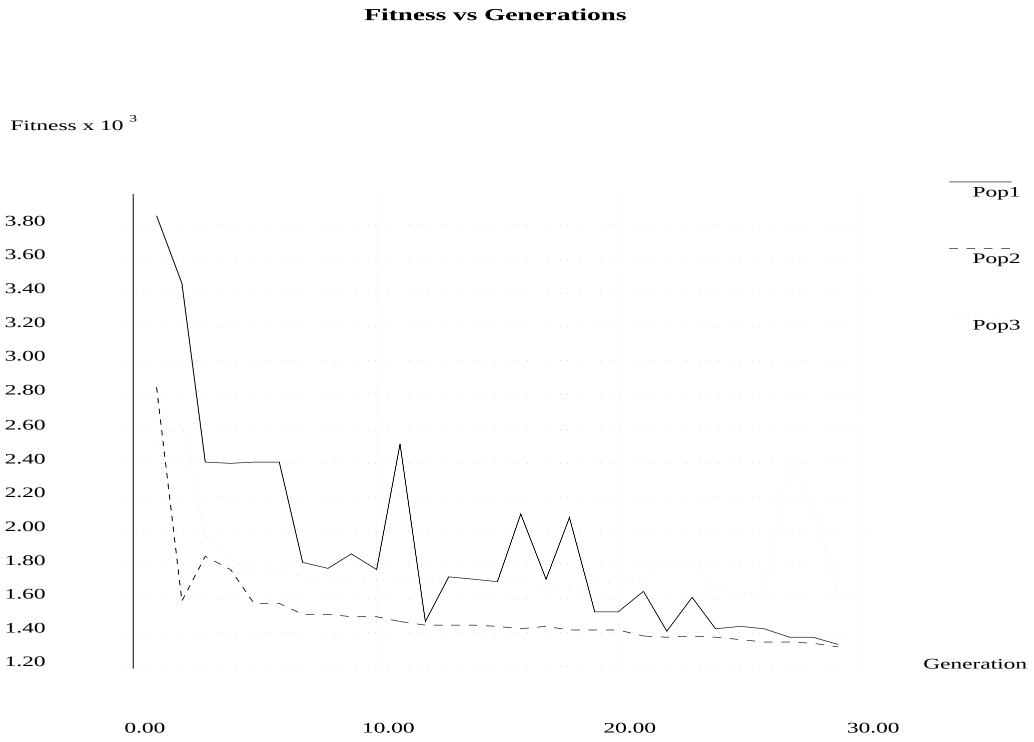

Figure E–3: Run 3 - Generation-Based, 1pt Cross.

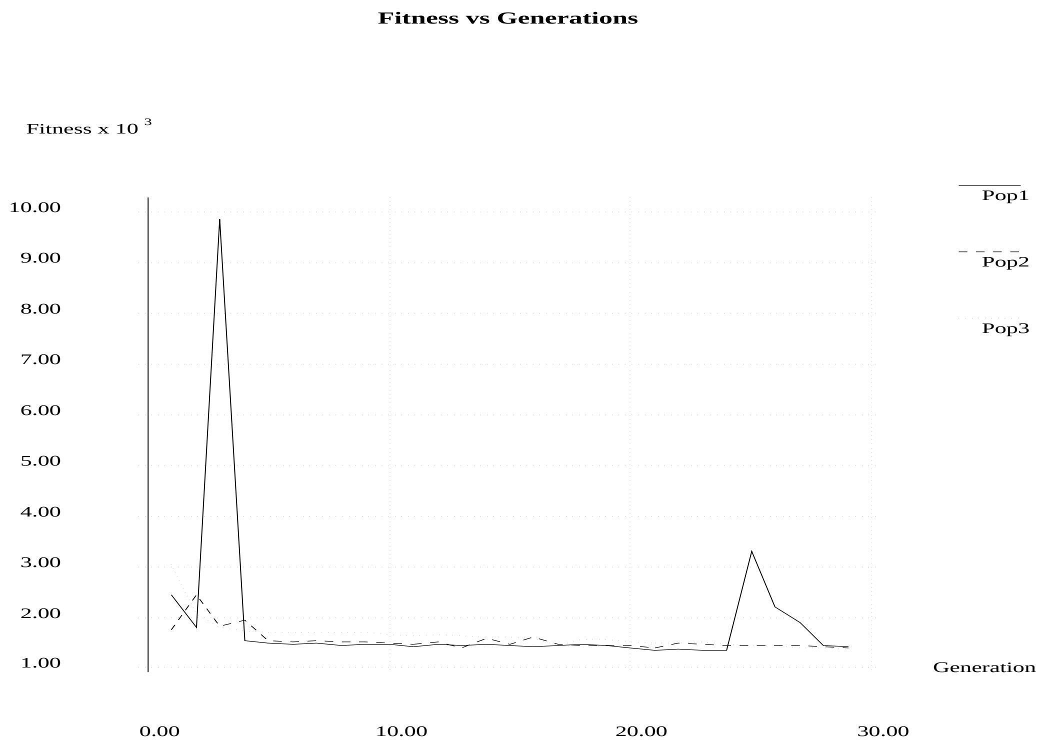

Figure E–4: Run 4 - Generation-Based, 1pt Cross.

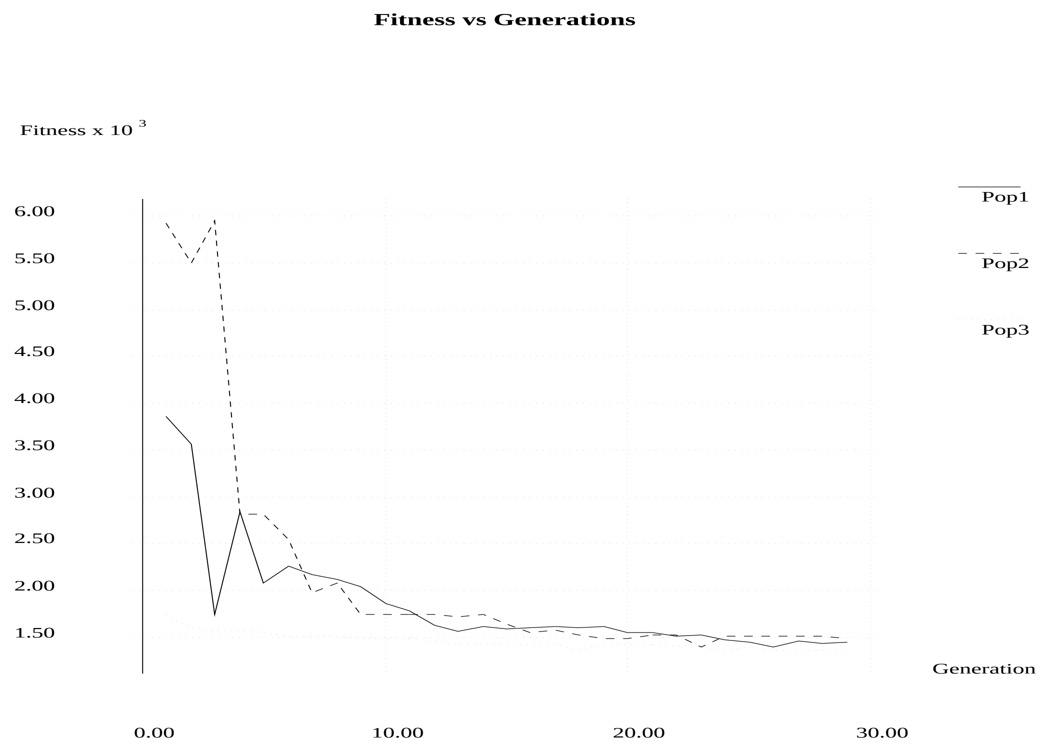

Figure E–5: Run 5 - Generation-Based, 1pt Cross.

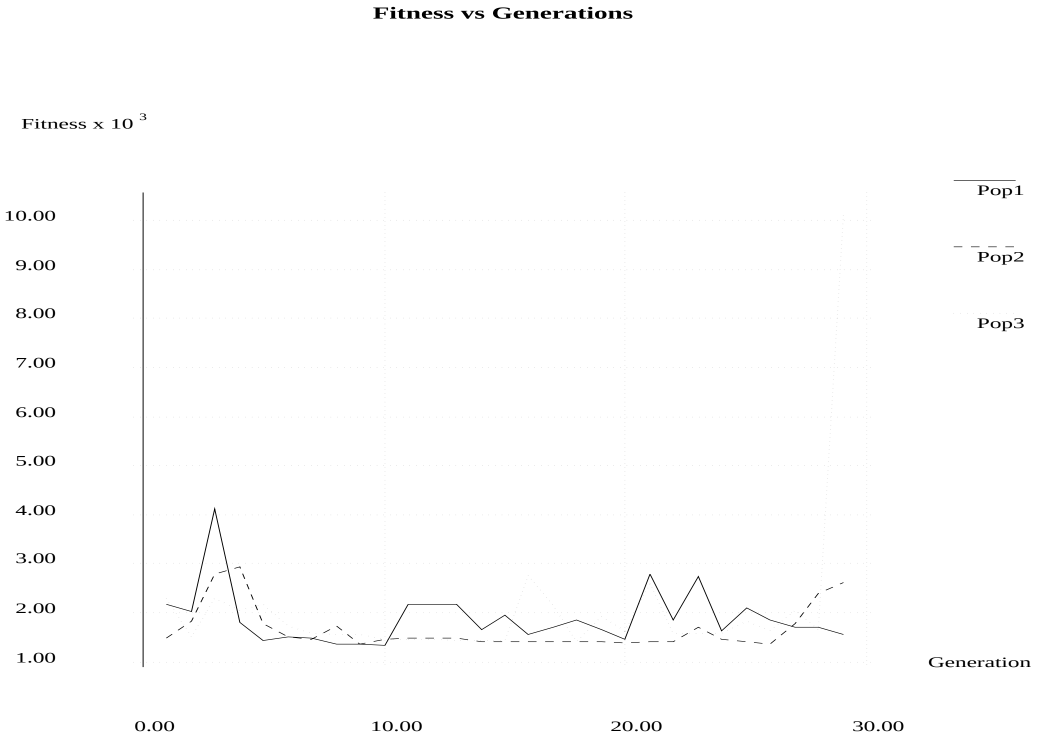

Figure E–6: Run 6 - Generation-Based, 1pt Cross.

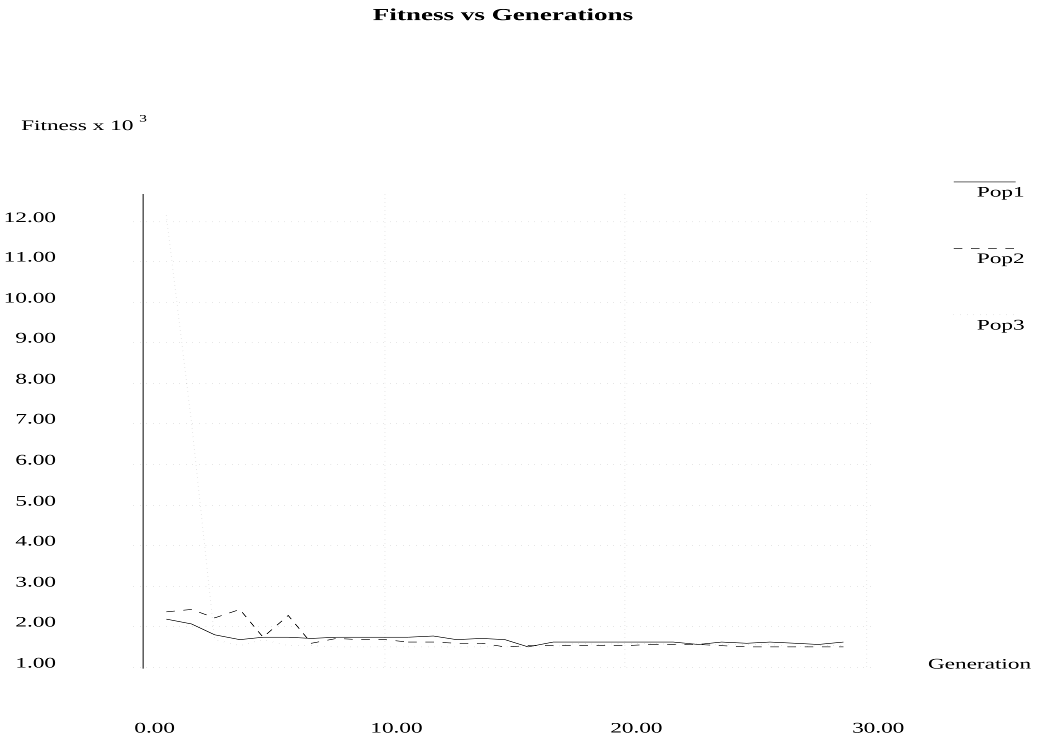

Figure E–7: Run 7 - Generation-Based, 1pt Cross.

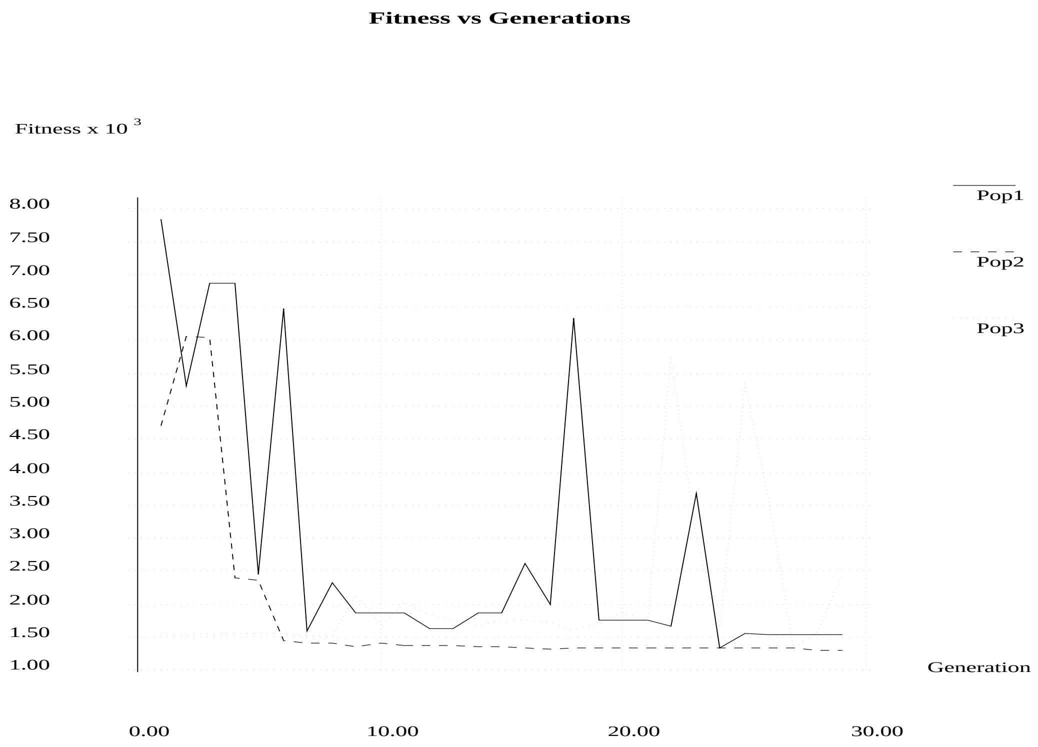

Figure E–8: Run 8 - Generation-Based, 1pt Cross.

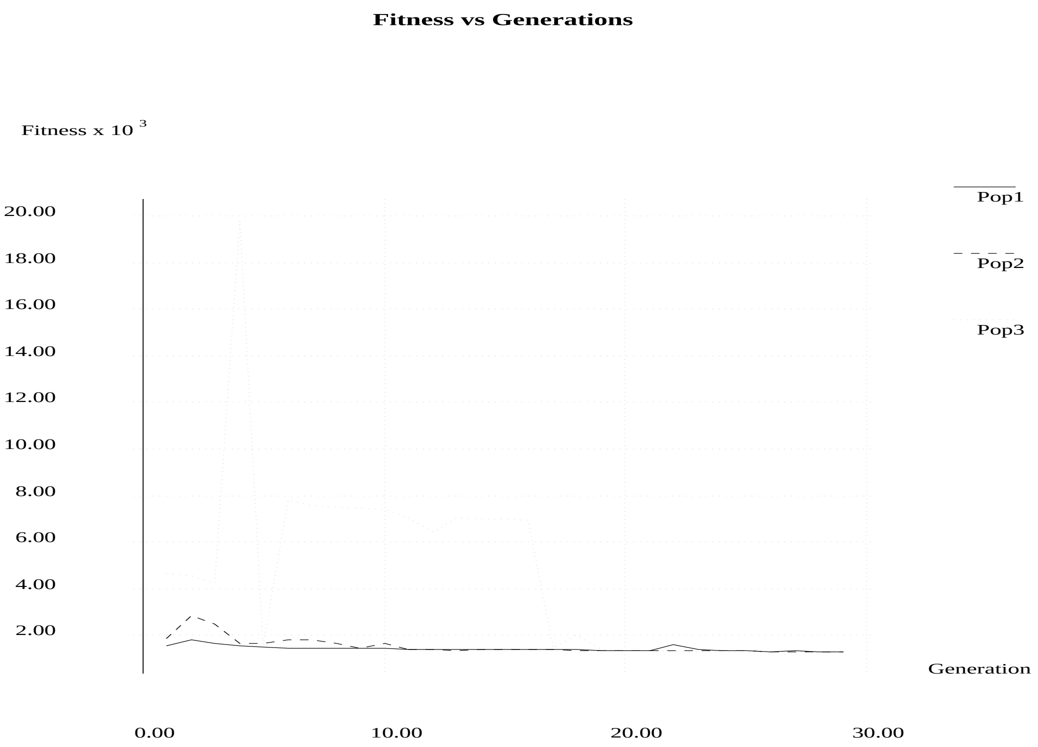

Figure E–9: Run 9 - Generation-Based, 1pt Cross.

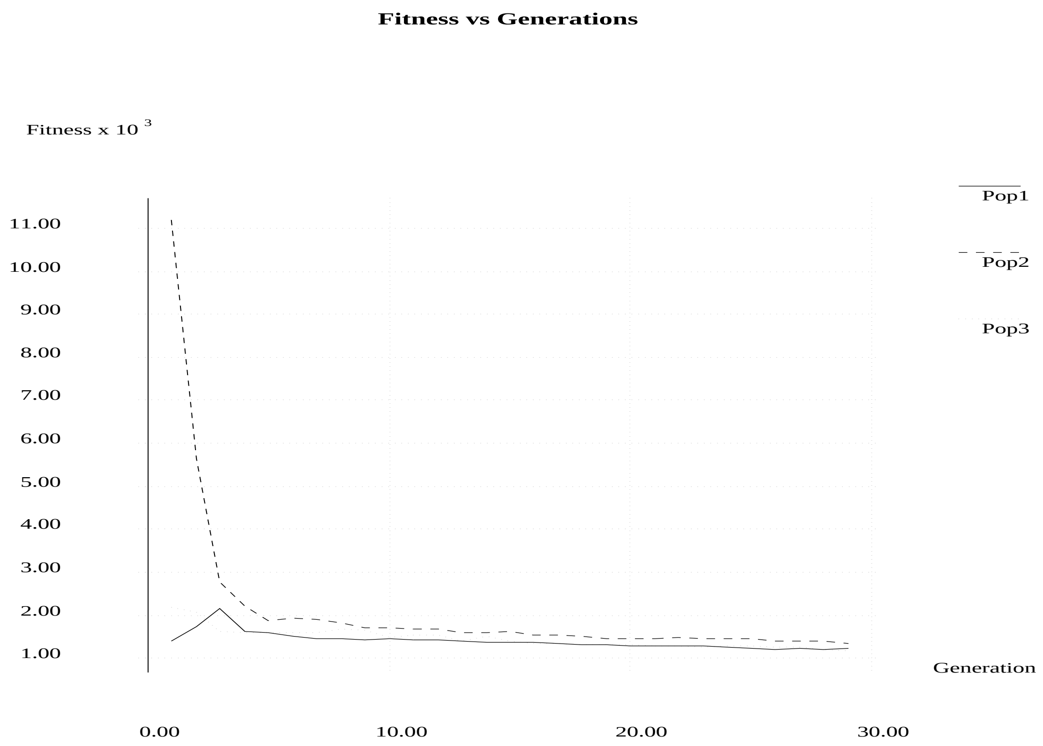

Figure E–10: Run 10 - Generation-Based, 1pt Cross.

Appendix F

Convergence Curves

Steady State

Uniform Crossover

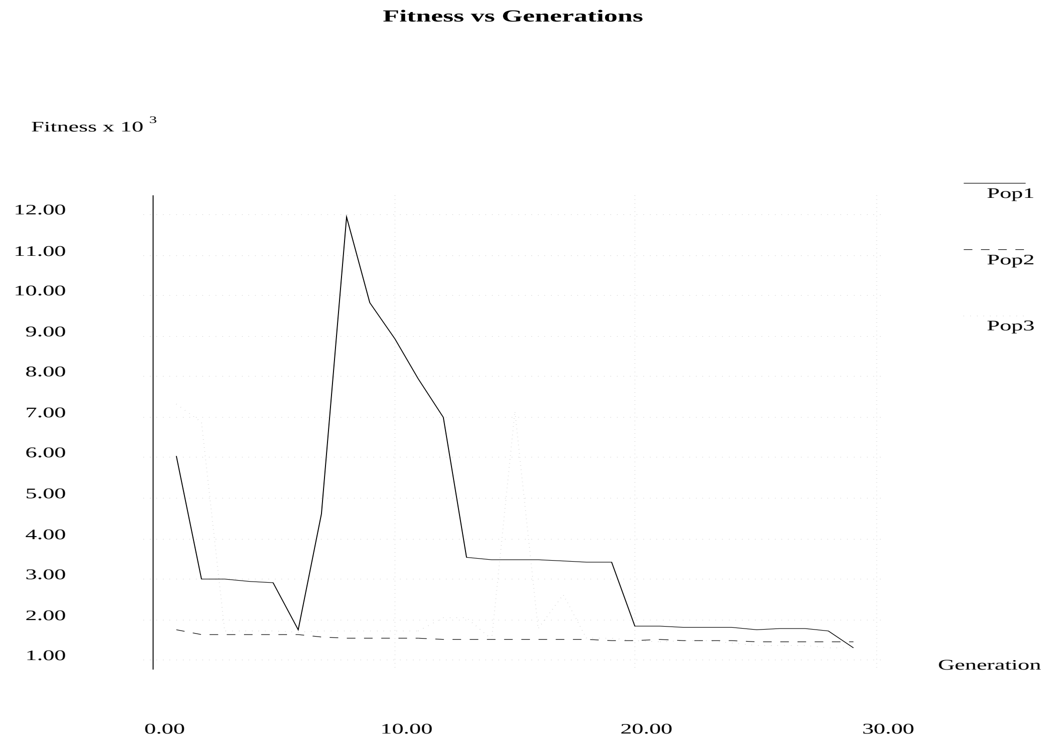

Figure F–1: Run 1 - Steady State, Uniform Cross.

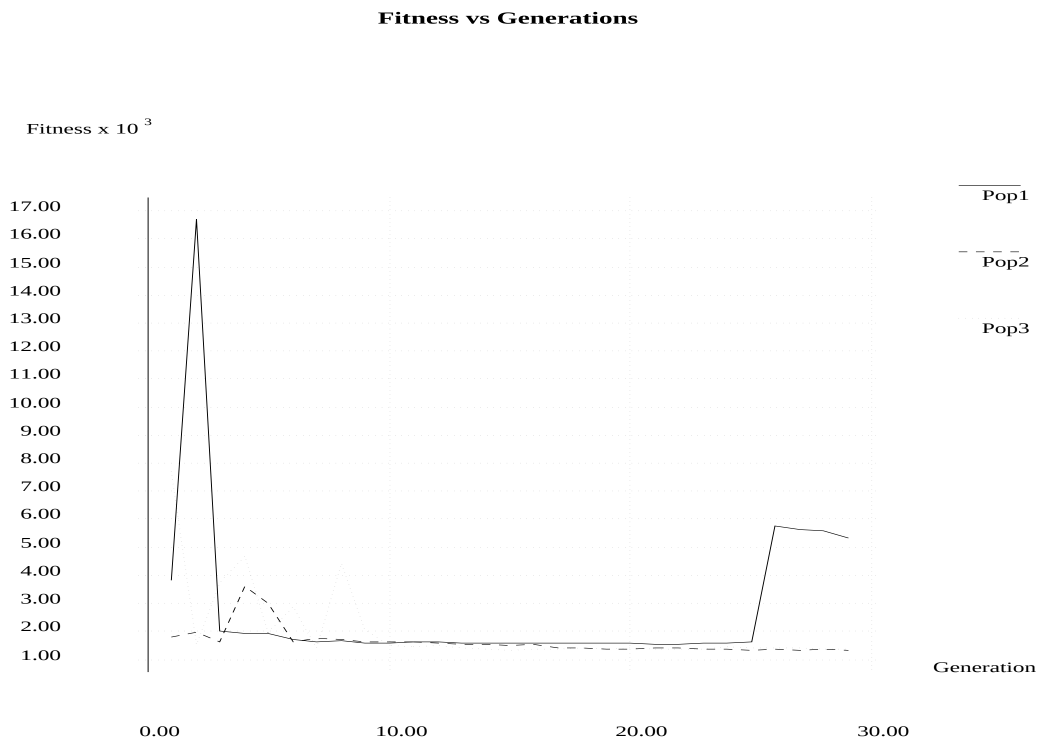

Figure F–2: Run2 - Steady State, Uniform Cross.

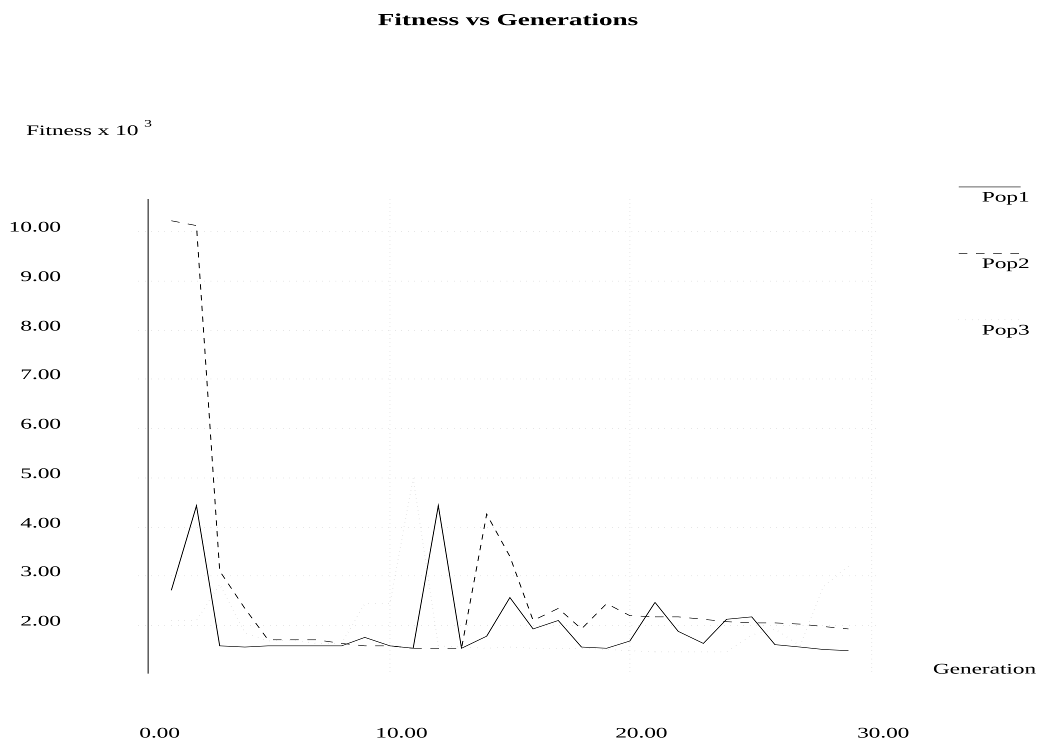

Figure F–3: Run 3 - Steady State, Uniform Cross.

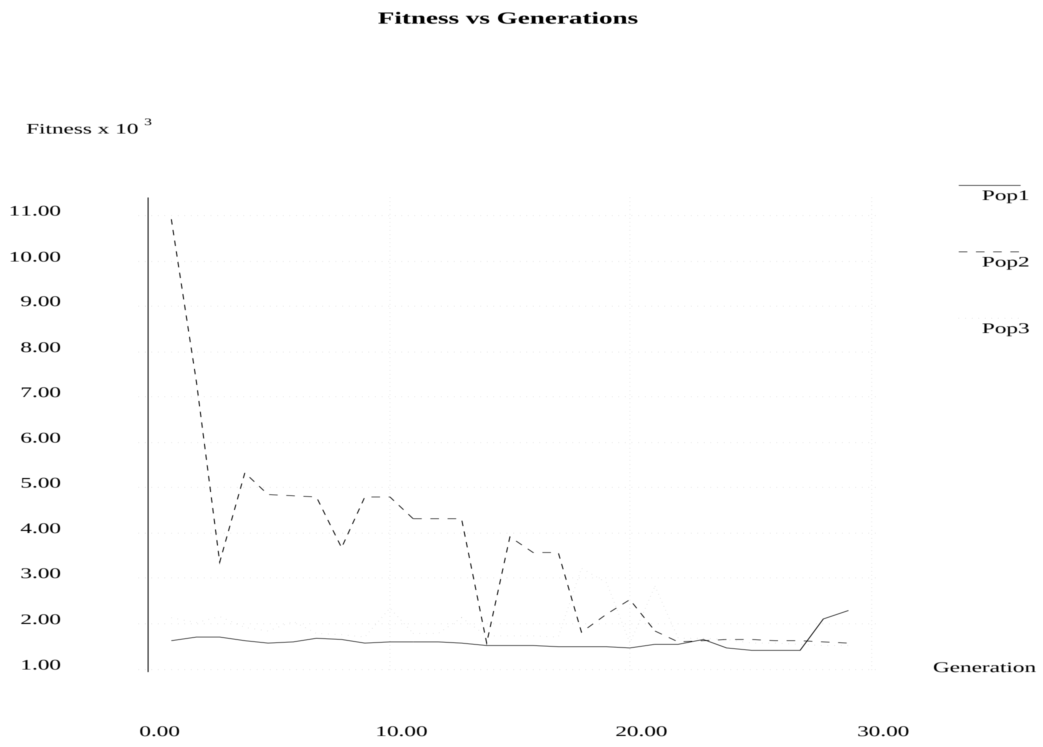

Figure F–4: Run 4 - Steady State, Uniform Cross.

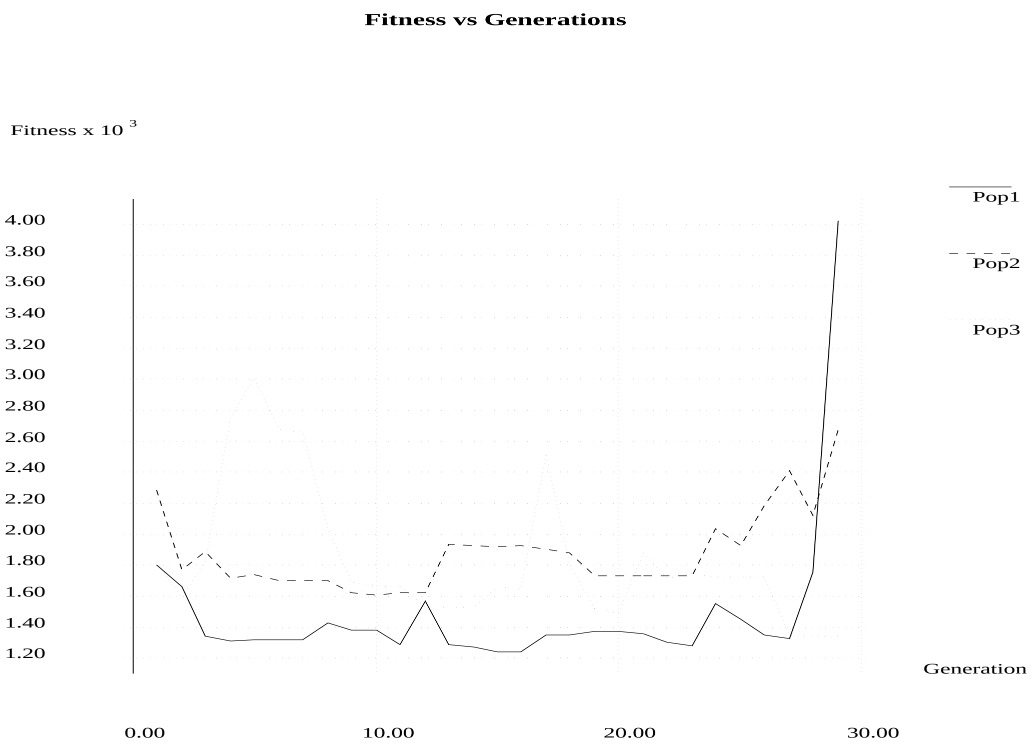

Figure F–5: Run 5 - Steady State, Uniform Cross.

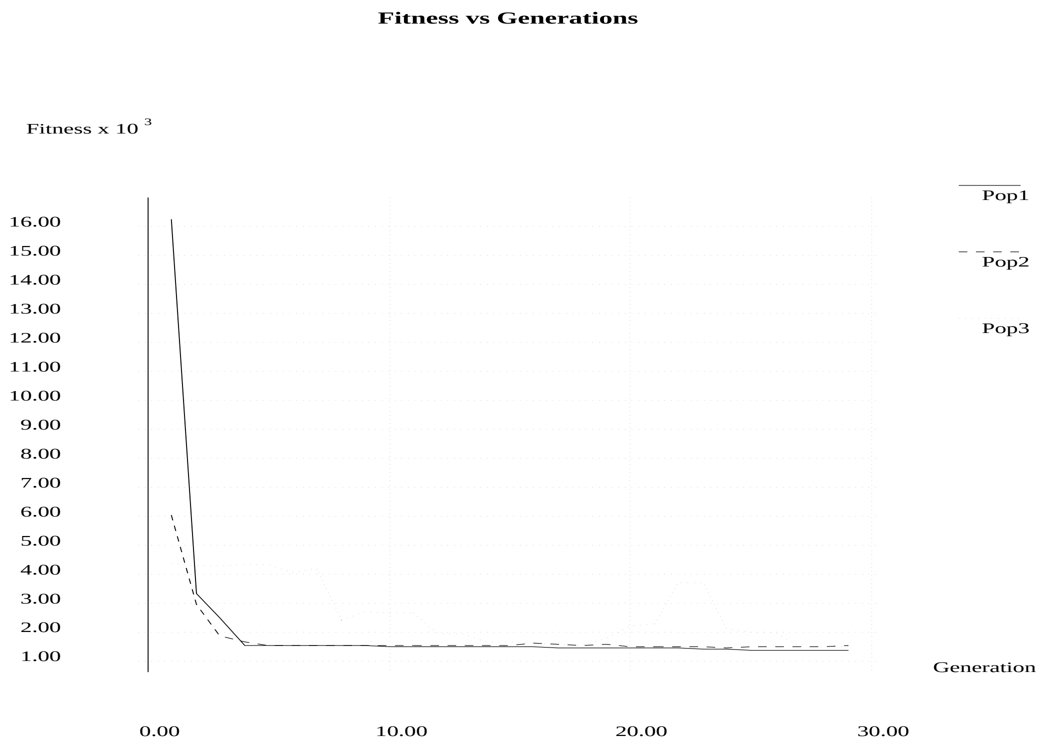

Figure F–6: Run 6 - Steady State, Uniform Cross.

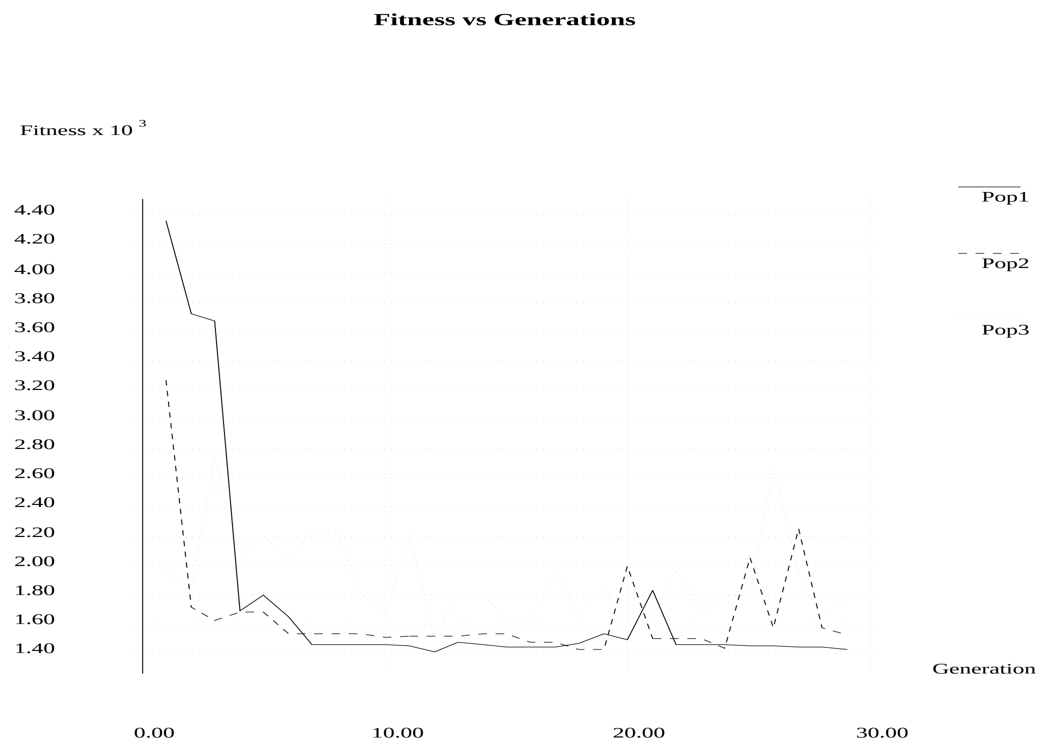

Figure F–7: Run 7 - Steady State, Uniform Cross.

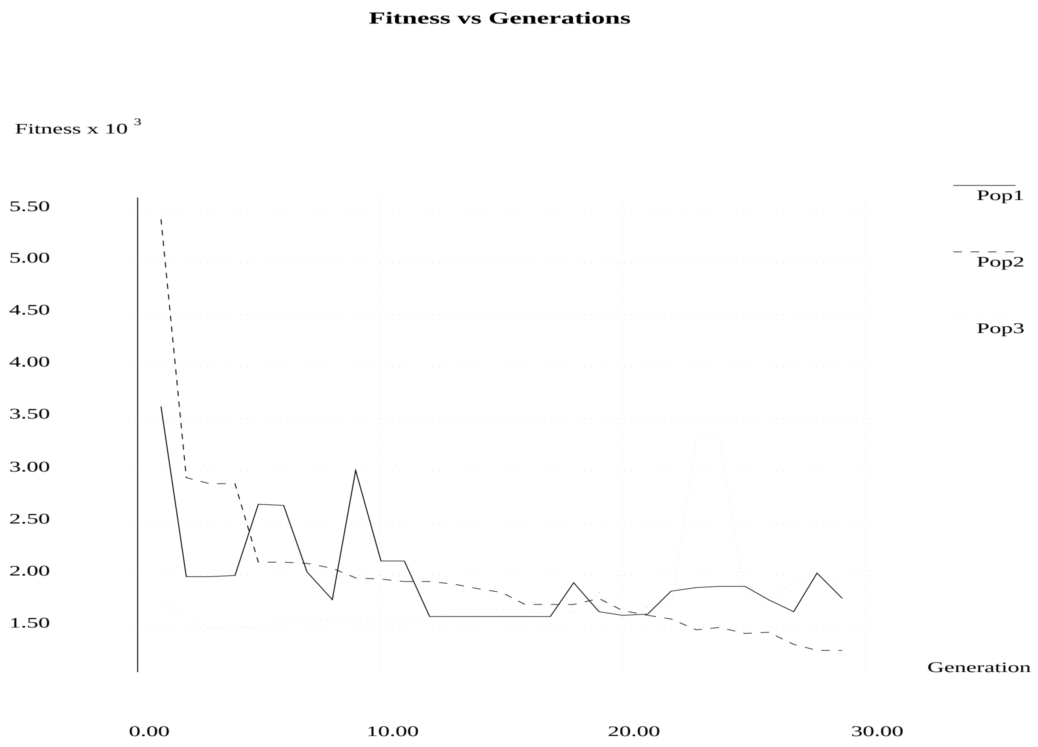

Figure F–8: Run 8 - Steady State, Uniform Cross.

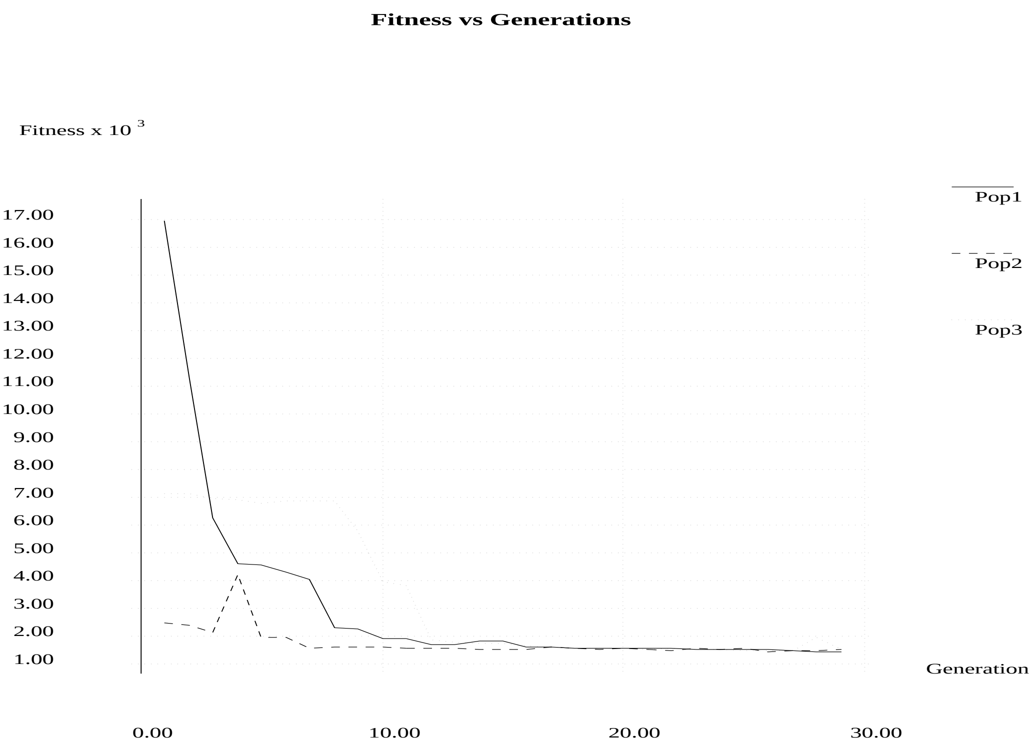

Figure F–9: Run 9 - Steady State, Uniform Cross.

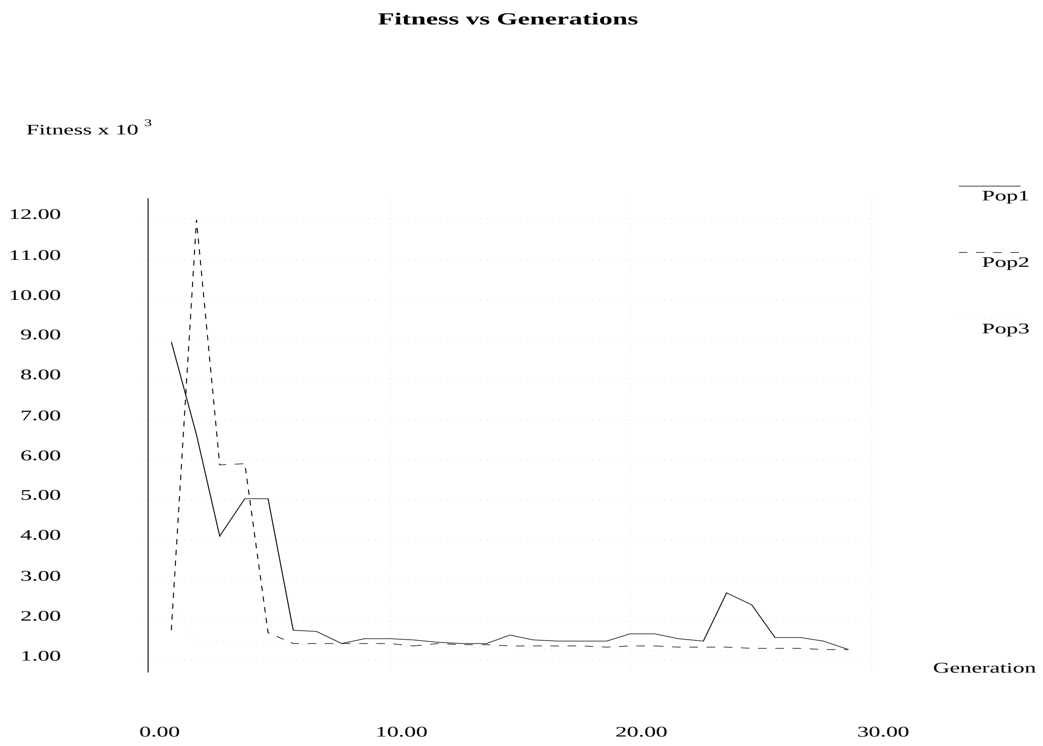

Figure F–10: Run 10 - Steady State, Uniform Cross.

Appendix G

Variance of Best Five Chromosomes

Steady State

One Point Crossover

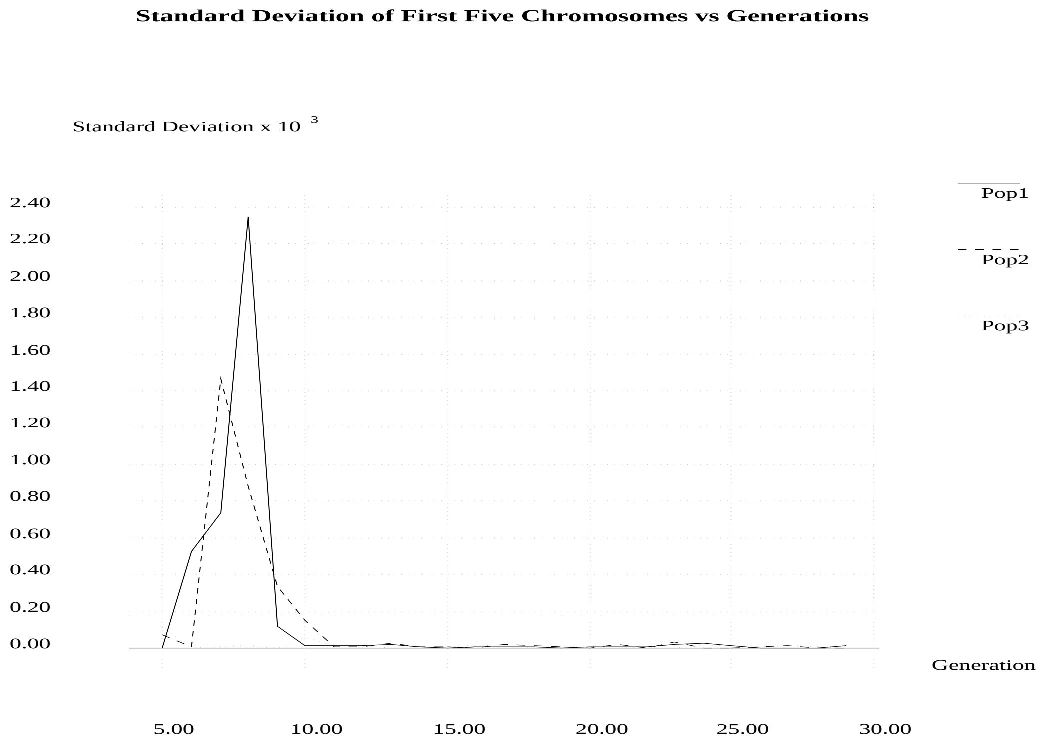

Figure G–1: Run 1 - Variance Test, Steady State, 1pt Cross.

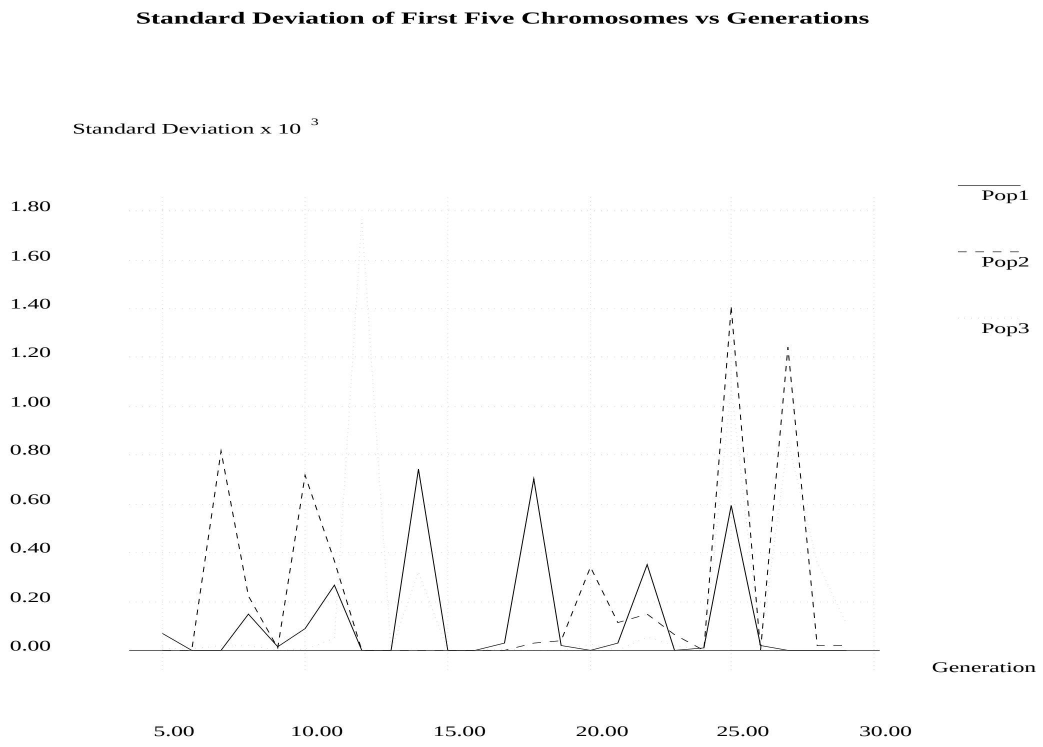

Figure G–2: Run2 - Variance Test, Steady State, 1pt Cross.

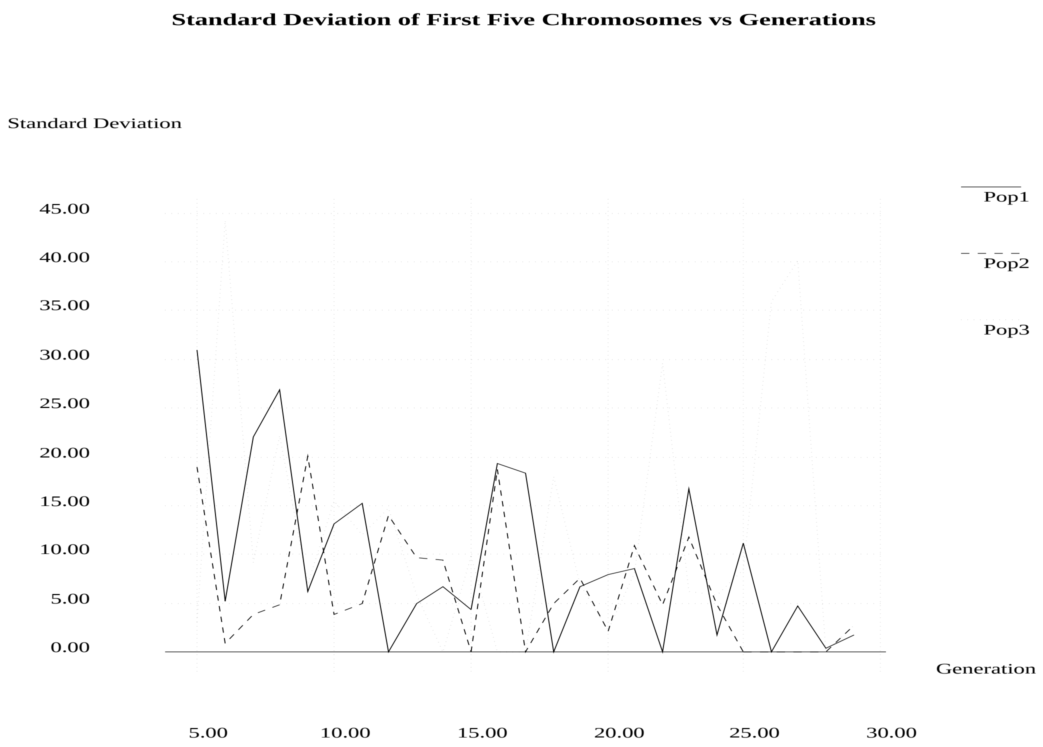

Figure G–3: Run 3 - Variance Test, Steady State, 1pt Cross.

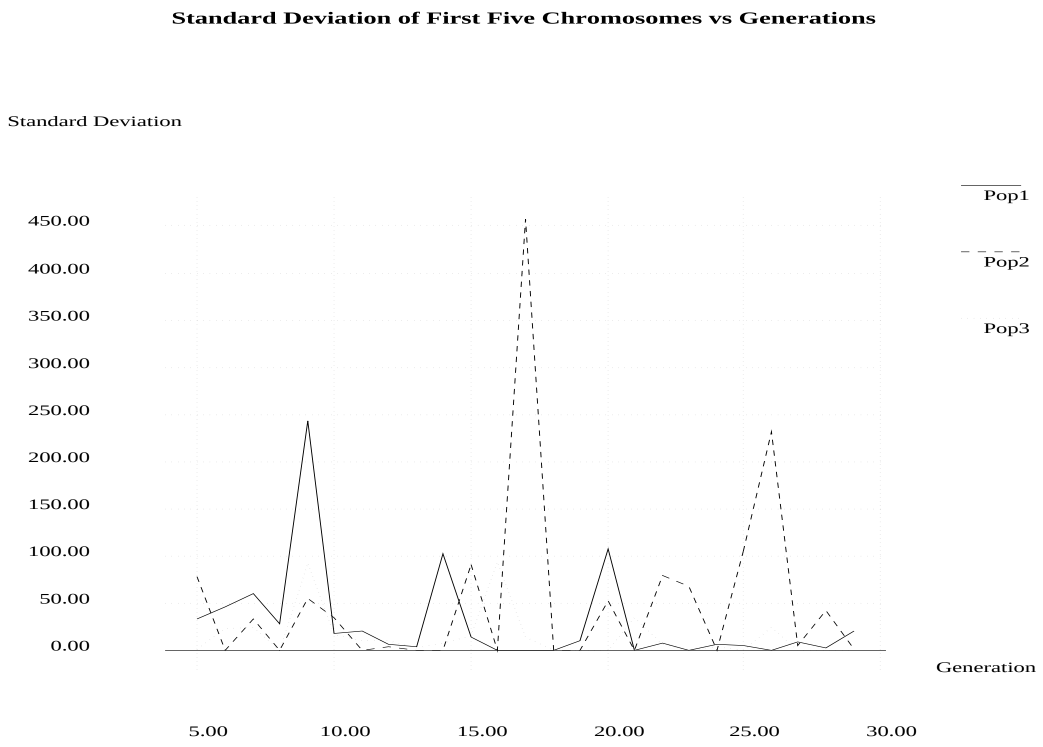

Figure G–4: Run 4 - Variance Test, Steady State, 1pt Cross.

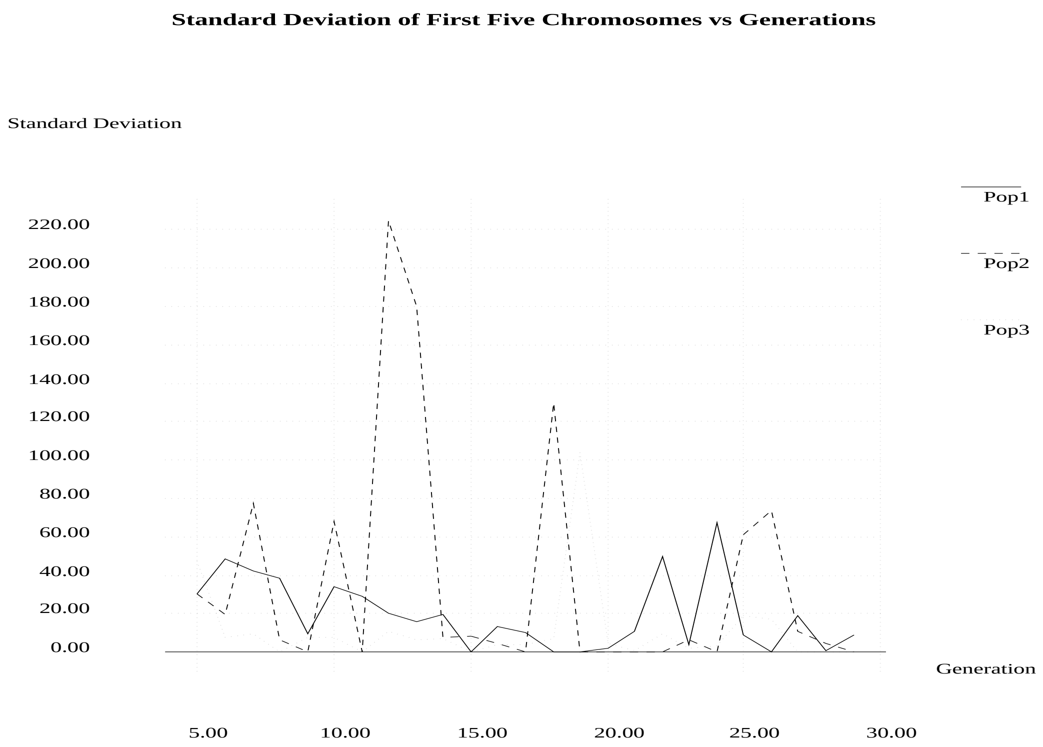

Figure G–5: Run 5 - Variance Test, Steady State, 1pt Cross.

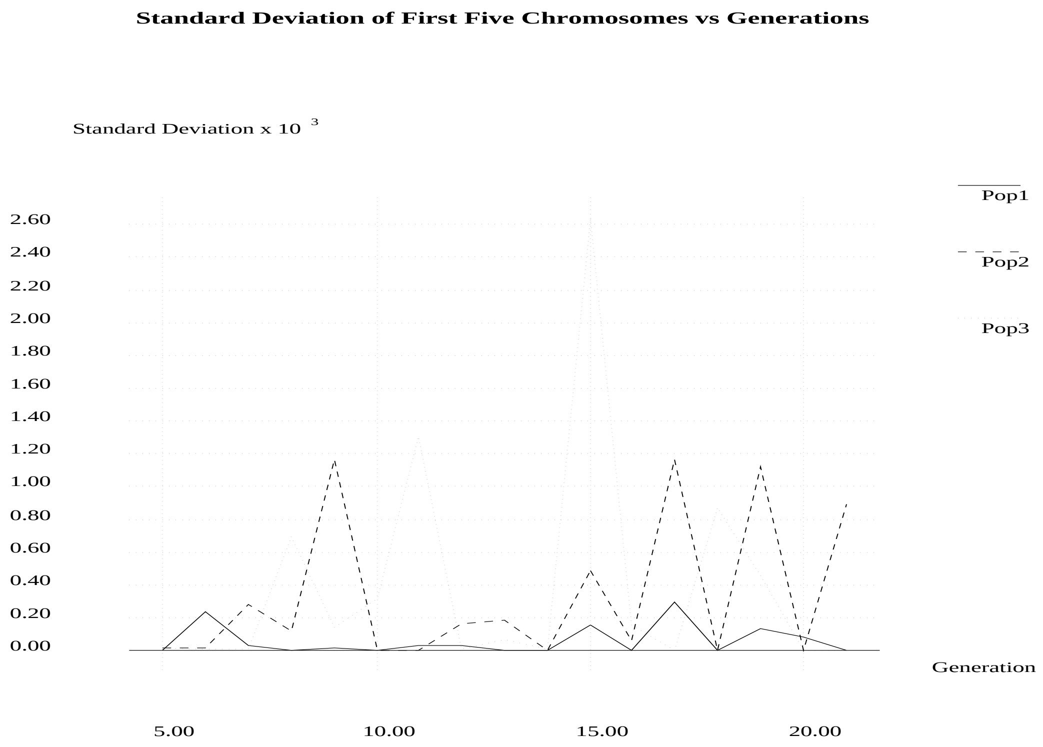

Figure G–6: Run 6 - Variance Test, Steady State, 1pt Cross.

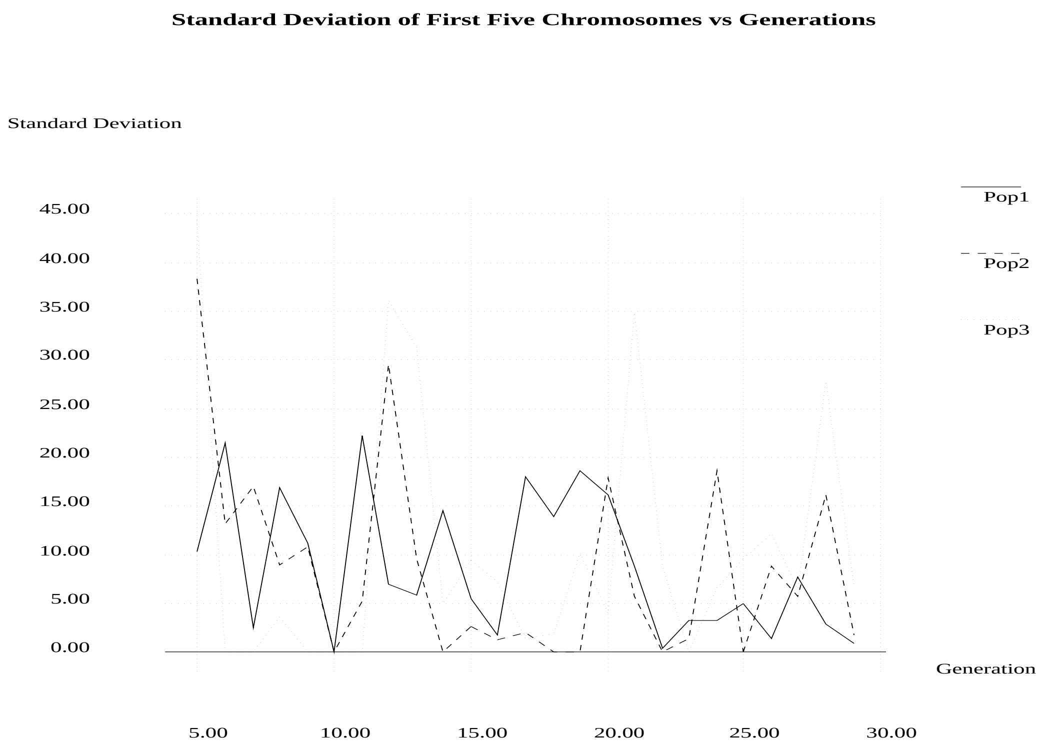

Figure G–7: Run 7 - Variance Test, Steady State, 1pt Cross.

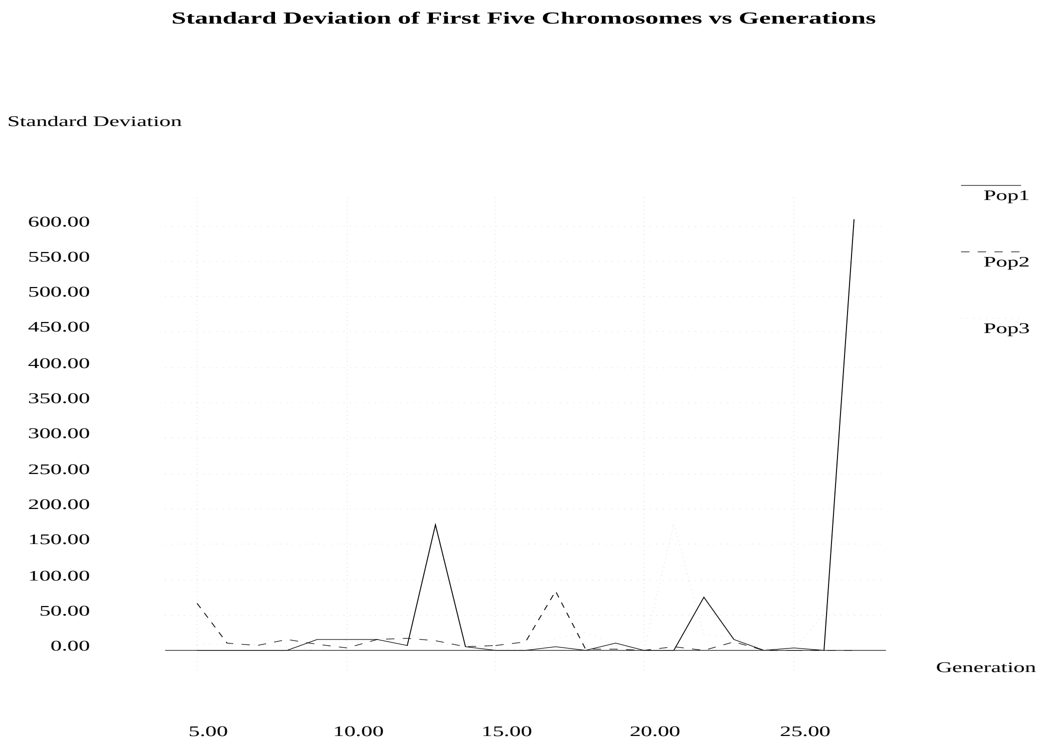

Figure G–8: Run 8 - Variance Test, Steady State, 1pt Cross.

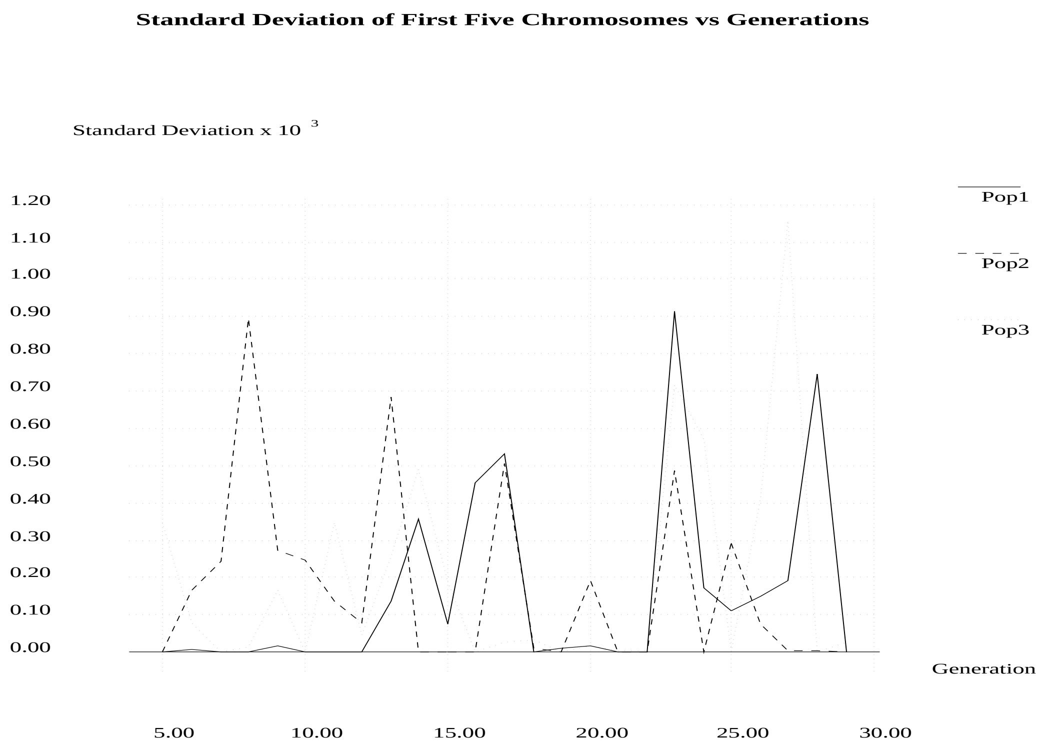

Figure G–9: Run 9 - Variance Test, Steady State, 1pt Cross.

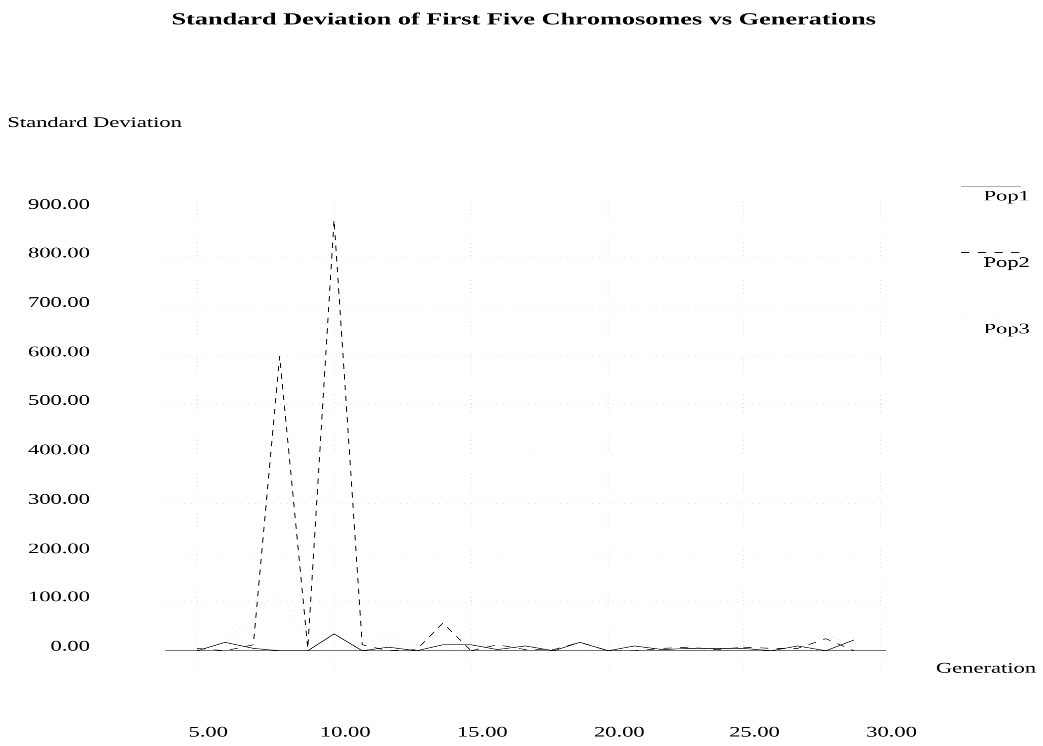

Figure G–10: Run 10 - Variance Test, Steady State, 1pt Cross.

Appendix H

Variance of Best Five Chromosomes

Steady State

Uniform Crossover

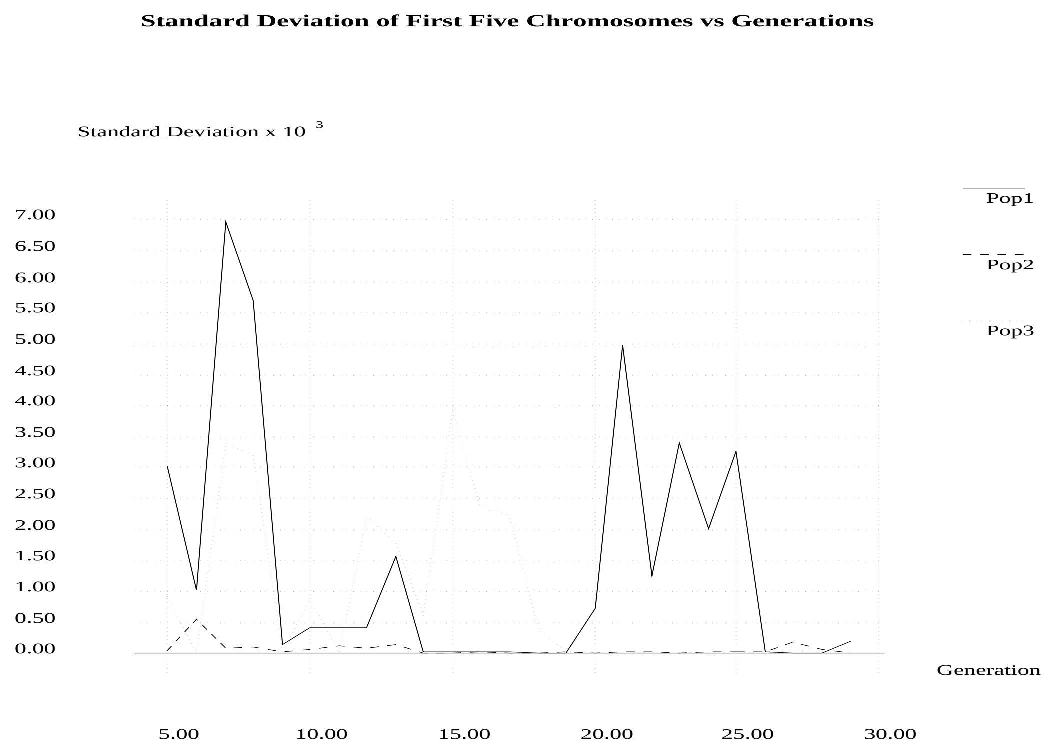

Figure H–1: Run 1 - Variance Test, Steady State, Uniform Cross.

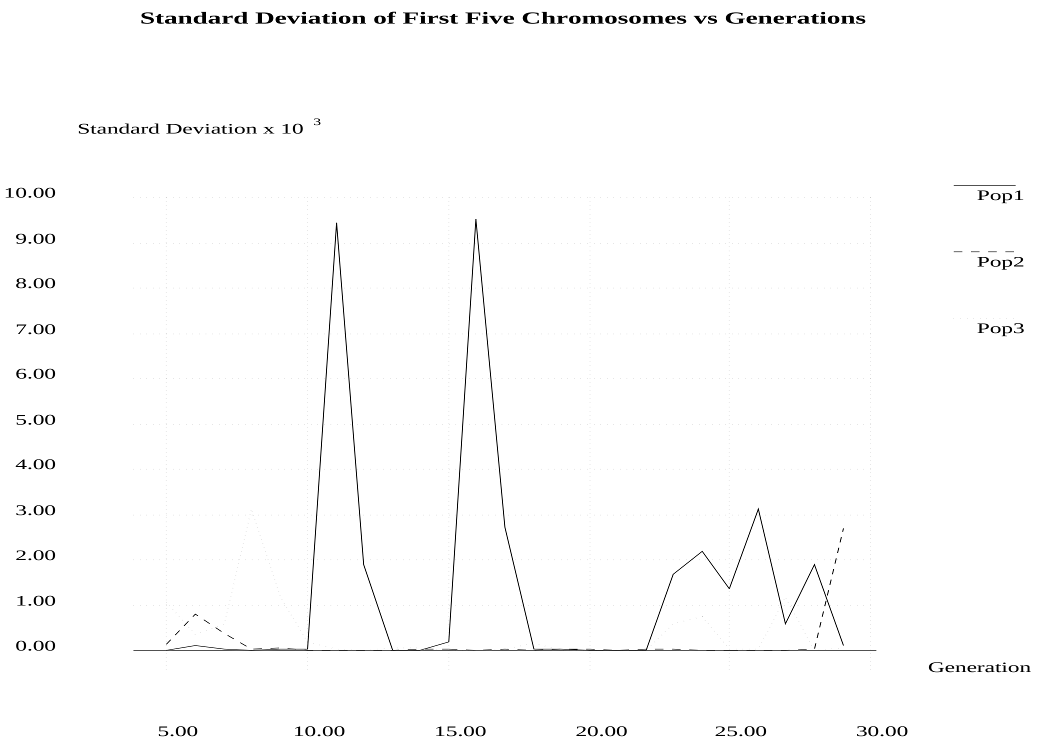

Figure H–2: Run2 - Variance Test, Steady State, Uniform Cross.

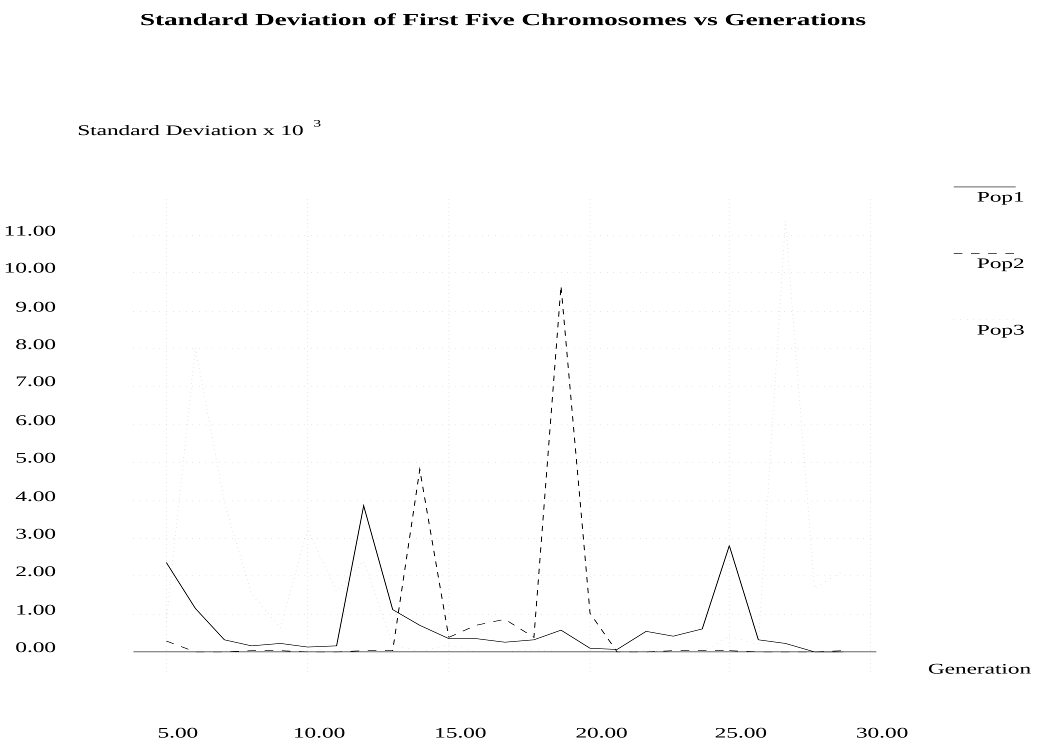

Figure H–3: Run 3 - Variance Test, Steady State, Uniform Cross.

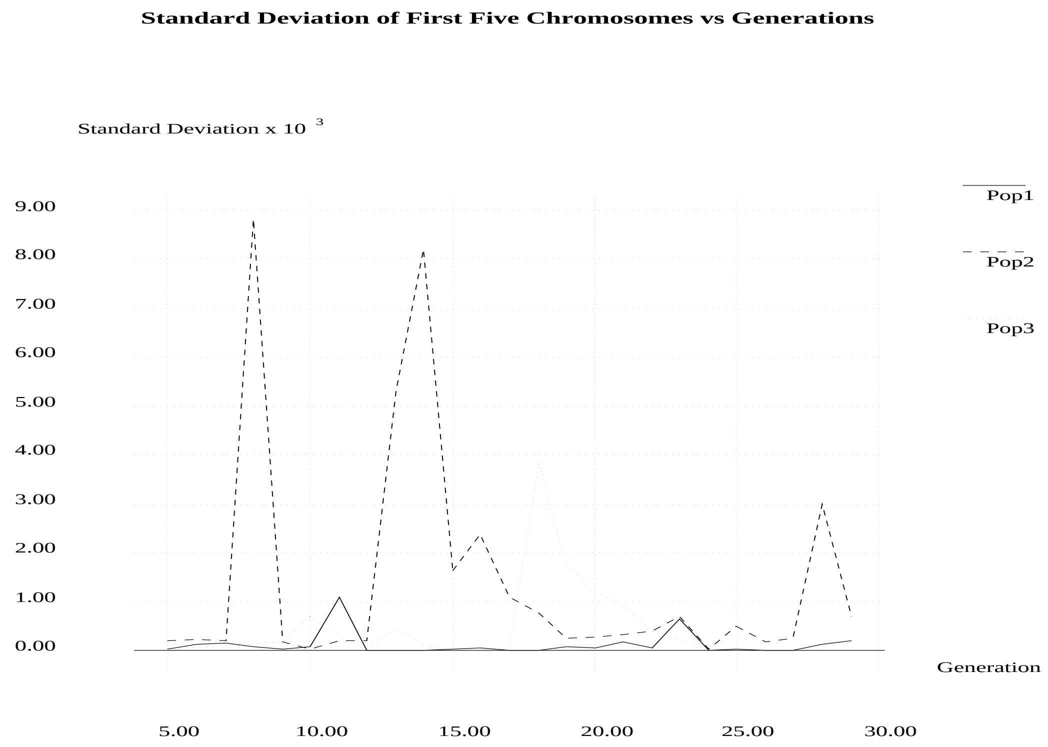

Figure H–4: Run 4 - Variance Test, Steady State, Uniform Cross.

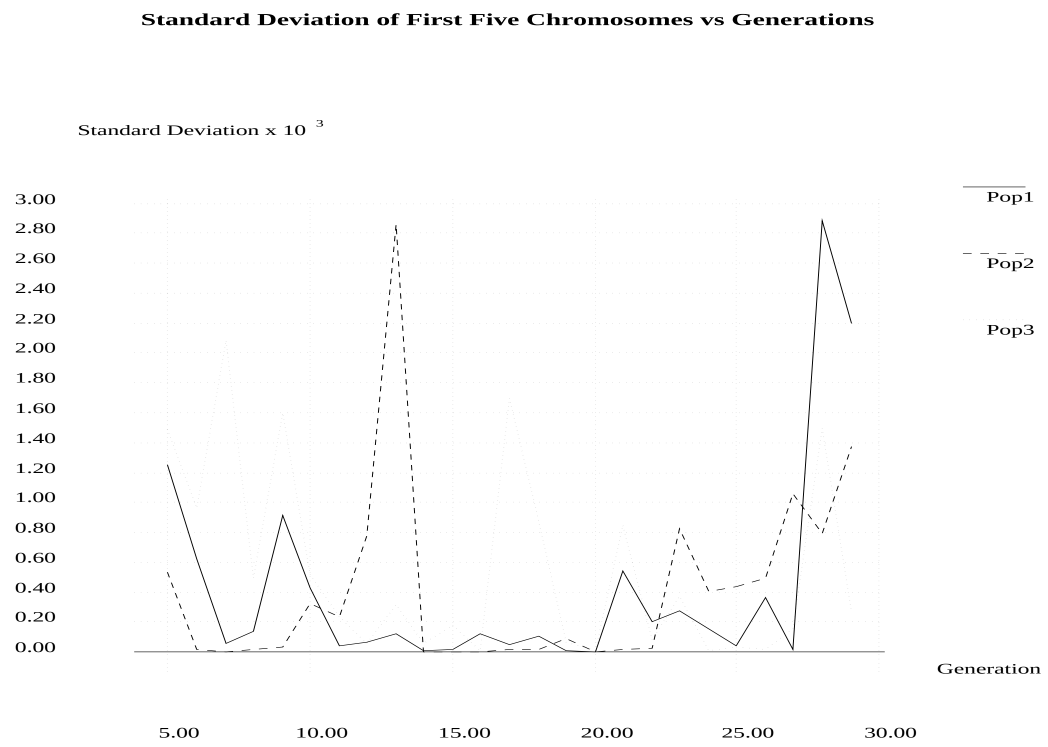

Figure H–5: Run 5 - Variance Test, Steady State, Uniform Cross.

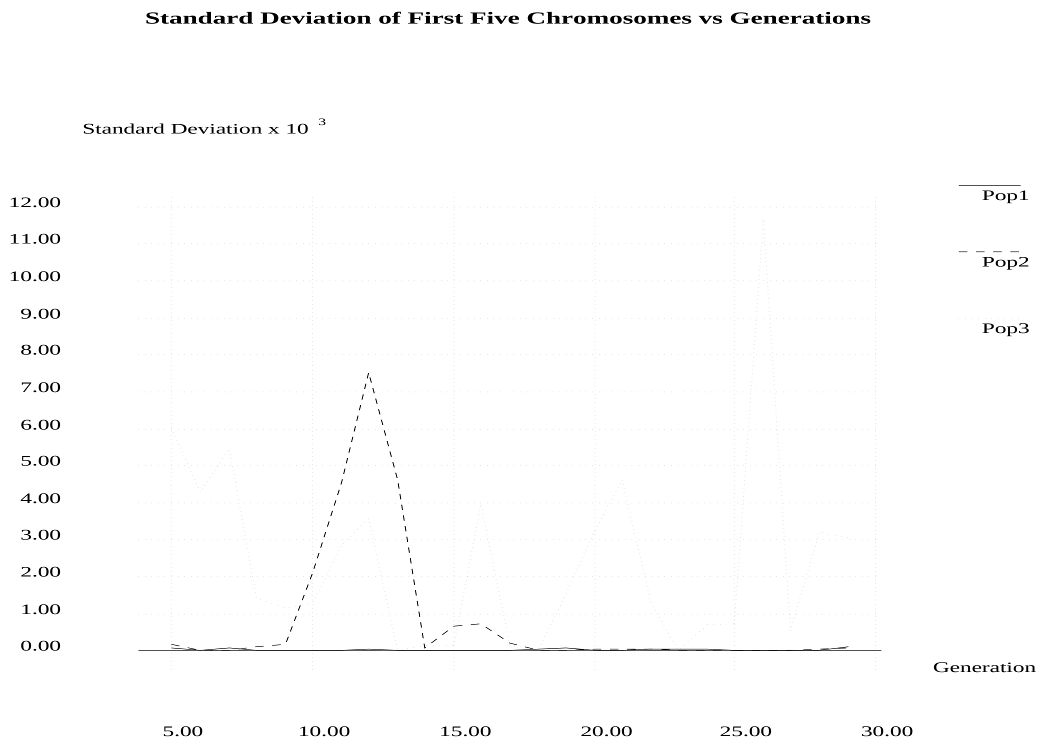

Figure H–6: Run 6 - Variance Test, Steady State, Uniform Cross.

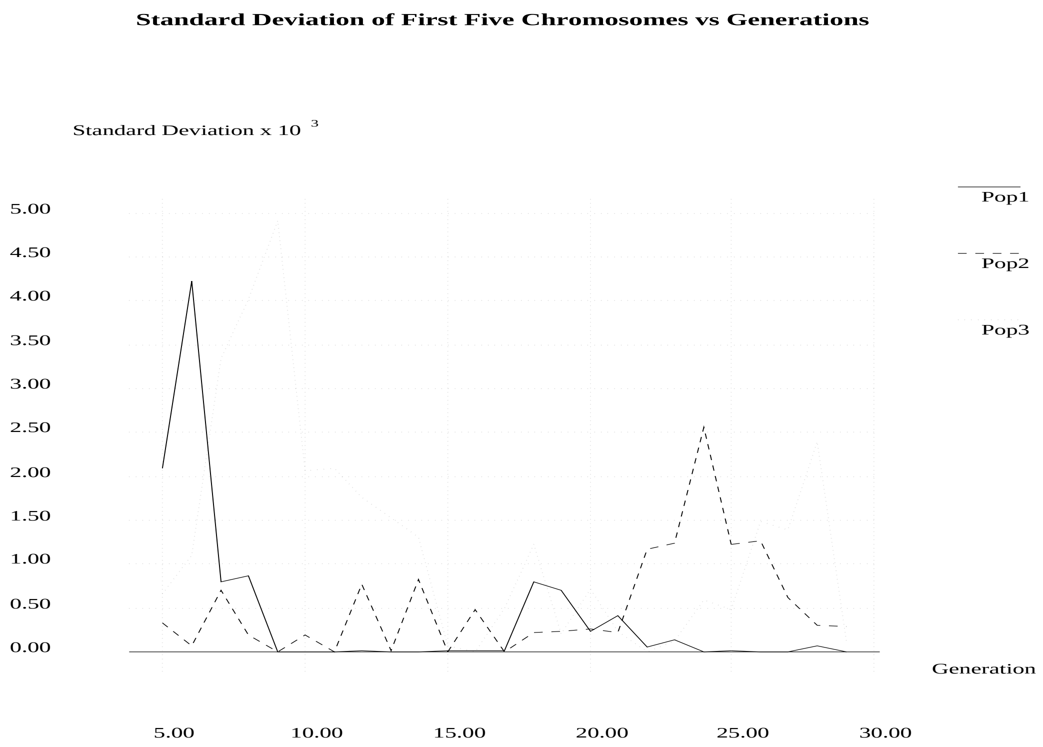

Figure H–7: Run 7 - Variance Test, Steady State, Uniform Cross.

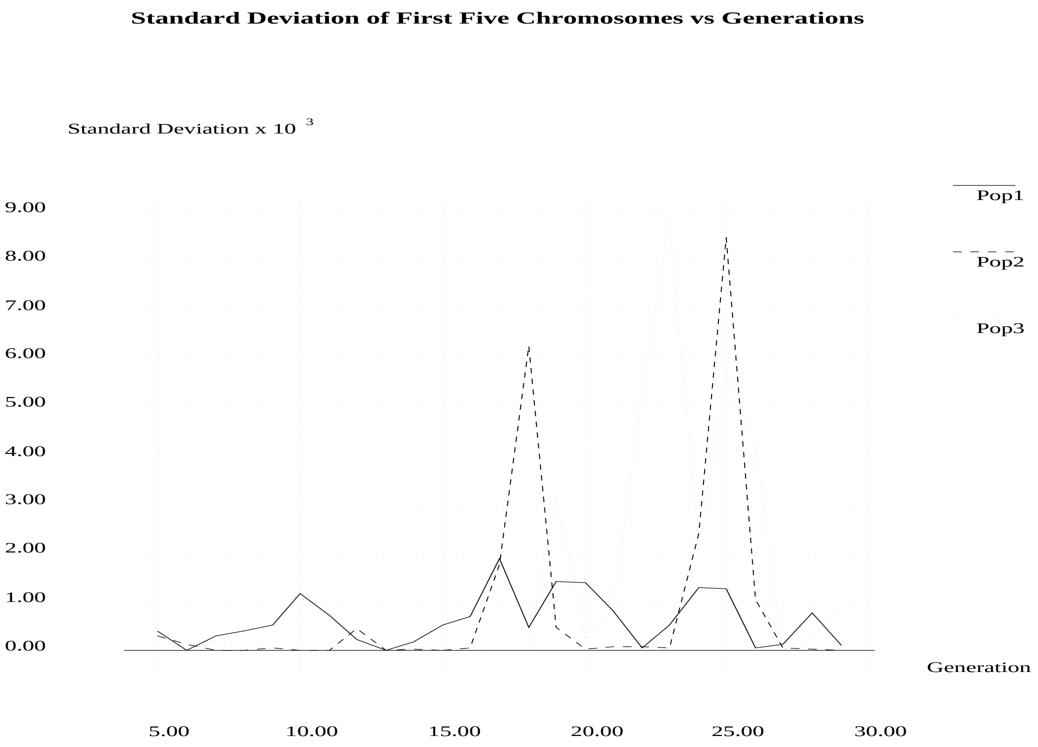

Figure H–8: Run 8 - Variance Test, Steady State, Uniform Cross.

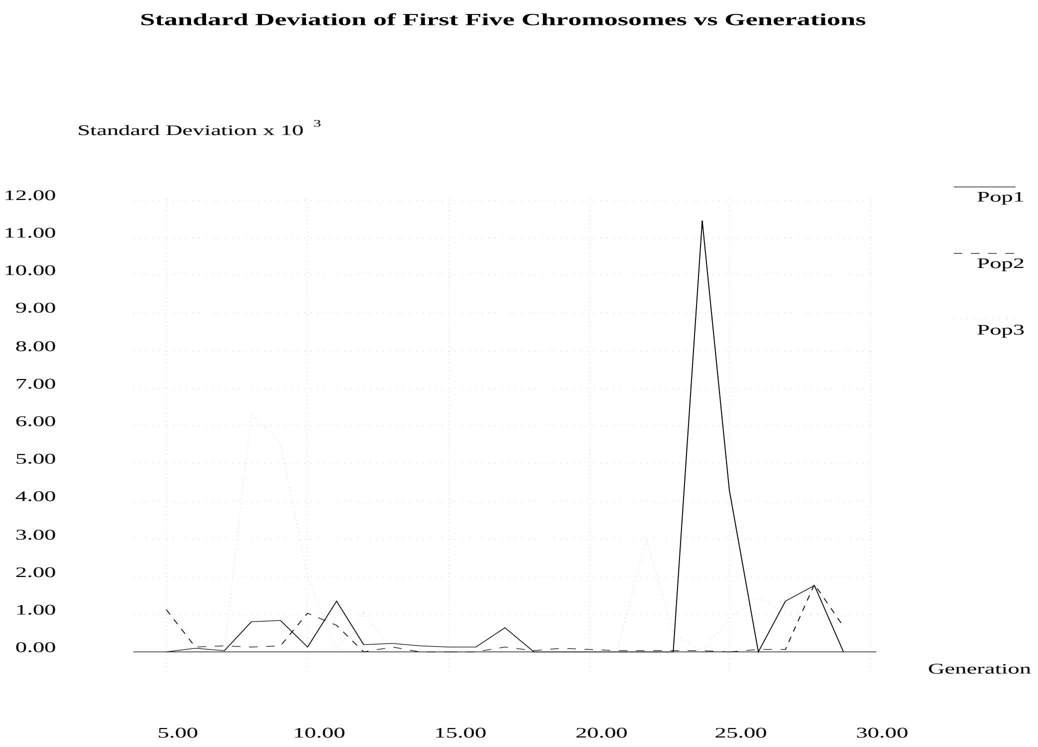

Figure H–9: Run 9 - Variance Test, Steady State, Uniform Cross.

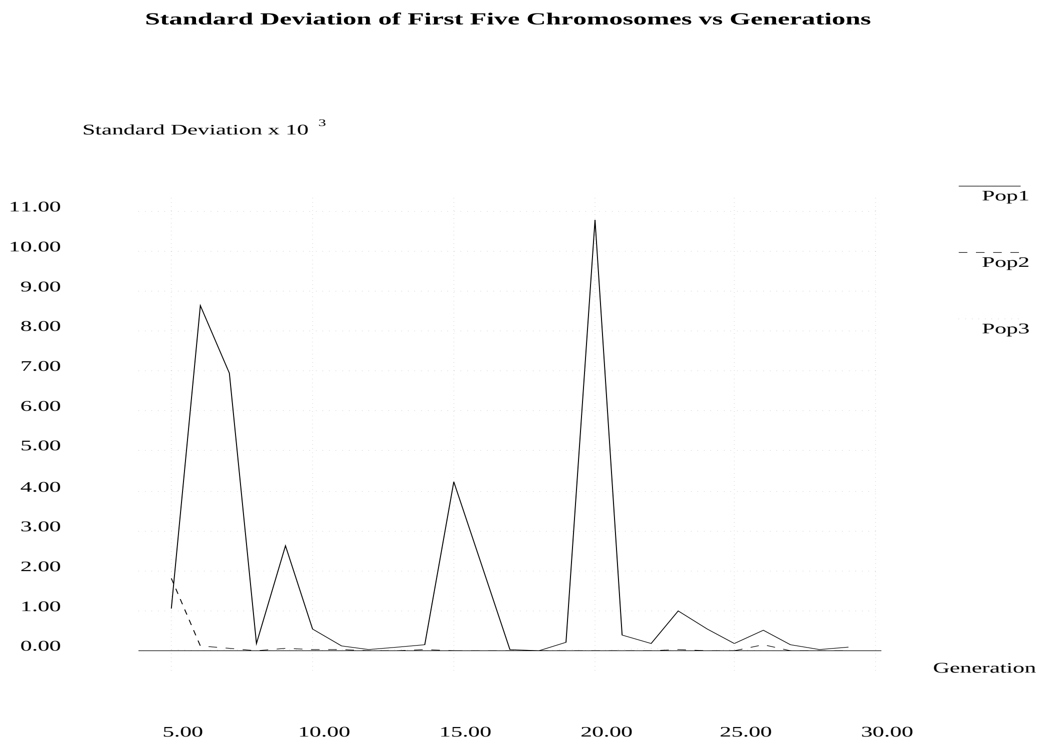

Figure H–10: Run 10 - Variance Test, Steady State, Uniform Cross.

Appendix I

Astronomical and Related Constants

I.1 The Sun

Radius = 696000 km

Mass = 1.99×10^{30} kg

I.2 The Earth

Mass = 5.98×10^{24} kg

Mean distance from Sun = 149.5×10^{6} km

Orbit eccentricity = 0.08183

Equatorial Radius = 6378 km

I.3 Venus

Mass $= 4.89 \times 10^{24}$ kg

Mean distance from Sun $= 108.1 \times 10^{6}$ km

Orbit eccentricity $= 0.006793$

Equatorial Radius $= 6200$ km

I.4 Mars

Mass $= 6.46 \times 10^{23}$ kg

Mean distance from Sun $= 227.8 \times 10^{6}$ km

Orbit eccentricity $= 0.093368$

Equatorial Radius $= 3400$ km

I.5 Jupiter

Mass $= 1.90 \times 10^{27}$ kg

Mean distance from Sun $= 778.0 \times 10^{6}$ km

Orbit eccentricity $= 0.048435$

Equatorial Radius $= 71370$ km

I.6 Saturn

Mass = 5.69×10^{26} kg

Mean distance from Sun = 1426.0×10^{6} km

Orbit eccentricity = 0.055682

Equatorial Radius = 60400 km

I.7 Physical Constants

Universal constant of gravitation = $6.668 \times 10^{-11} \;\; N\, m^2\, kg^{-2}$