University College London

FTSE 100 Price Prediction with a multi-factor model using Genetic Programming

BSc Intelligent Systems Project 2005/2006

By Harjinder Bagria

Supervisor: Chris Clack

This report is submitted as part requirement for the BSc Computer Science degree in the Department of Computer Science at University College London. It is substantially the result of my own work except where explicitly indicated in the text. This report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

This report presents a Genetic Program (GP) based system that searches for the most profitable financial rules for making buy and sell decisions in the FTSE 100 stock index. These rules are generated from combinations of frequently used fundamental and technical indicators. The main purpose of this project is to support investors in making more informed decisions on future investments. The design includes an investment simulator that evaluates every rule (a GP chromosome), returning a Sharpe Ratio, which represents the fitness of a rule. The trading performance of the system was tested in experiments using historical price data from many stocks in the FTSE 100 stock index. The results have shown that the GP-system can generate rules that return significant profit in excess of the FTSE 100 index when applied to the stock market.

1. Project Introduction

This project aims to assist people in making more informed decisions on future investments. I am focusing on the FTSE 100 equities (stocks). Investment Banks and individuals all around the world invest in these equities every day. In order to assist people, I am going to use historical Technical and Fundamental information of the FTSE 100 equities to forecast future price movements. In order to obtain all this information I am using Reuters3000Xtra to obtain data, and importing the data using PowerPlusPro to create spreadsheets for the stocks.

Technical analysis is a method of evaluating equities by analysing various statistics generated by market activity, past prices and volume. Technical analysis involves looking at stock charts for indicators and various patterns that will be used to determine a stock's future price performance. Fundamental Analysis involves evaluating a stock by measuring its intrinsic value. This involves studying everything from the economy to specific industries, financial conditions and the management of stocks.

The main aim of the project is to create a system to find a Non Linear Factor Model (NLFM) with good investment performance. A NLFM is a combination of financial indicators and arithmetic functions. Reading in historical data will allow the system to decide what is the best NLFM to use to obtain the greatest return on investment, in future transactions. Due to the search space of NLFMs being enormous, I believe the most optimal solution would be to use an evolutionary computation technique, to generate NLFMs based on the theory of Darwinism. Producing a genetic program in Java would allow me to do this. There have been previous projects that have used genetic programming in order to manipulate technical analytical rules. However, I have not found a reference to a study that has used a Genetic Programming System in order to search for a NLFM using both technical and fundamental rules. I believe that the more information used for an equity will allow a greater amount of accuracy when assessing its potential for profitability.

1.1 Project Objectives

The objectives of the project were:

1. To program PowerPlusPro spreadsheets in order to capture stock data.
2. To design and implement an investment simulator that uses a non-linear multi-factor equation (chromosome) to determine buy and sell decisions for a long/short equities hedge fund (based on FTSE 100 stocks).
3. To modify and extend an existing Genetic Program System, which calls the investment simulator to determine the fitness of each chromosome.
4. To run experiments to determine the efficacy of the Genetic Program and Investment Simulator system. The experiments should monitor:
   i) The behaviour of the system during "training".
   ii) The behaviour during the out-of-sample test.
5. To undertake a brief critical evaluation of the system and experimental results

1.2 Chapter Summary

In this report the following chapters will cover:

Background: This chapter gives a comprehensive description of the stock analysis techniques used and details what a genetic program is.

Design: This section explains thoroughly the design of the investment simulator and the genetic program. The design of the historical data used in the project is also included.

Implementation: This chapter characterises how the design was implemented. Therefore, the implementation of the data, investment simulator and the GP system is included.

Experiment: This documentation outlines how the two experiments were set-up and the results of these experiments.

Summary & Conclusion: This chapter provides a summary of the objectives achieved in the project. A critical evaluation and conclusion of the project along with further work that could be done to improve it, is outlined.

2. Background

This chapter gives a comprehensive description of the stock analysis techniques used in the project and details what a genetic program is.

2.1 Stock Analysis Techniques

2.1.1 Fundamental Analysis

Fundamental analysis concentrates on the forces of supply and demand, which directly cause stock prices to fluctuate [1]. This approach examines all of the factors that directly affect the price of stocks, and are used to determine the intrinsic value of stocks. The intrinsic value is known as the value of the stock based on the underlying perception of the value. If the intrinsic value is under the current price then the stock should be sold. If the intrinsic value is above the current price then the stock is forecasted for a future price increase and should be bought. This theory of the intrinsic value can also apply to the market as whole. Fundamental analysts have to always know the reason for a stock's price movement. The various fundamentals being studied on this project include Earnings Per Share and the Return on Equity, amongst others.

2.1.2 Technical Analysis

Technical analysis is the study of market action, primarily through the use of charts, for the purpose of forecasting future price trends. The main sources of information used for technical analysis are the stock price and trading volume. Technical analysis is a very important form of stock analysis and many analysts only use this to forecast future stock price movements.

John J. Murphy suggested the three main premises on which the technical approach is based on is [2]:

1. Market action discounts everything.
2. Prices move in trends.
3. History repeats itself.

The first statement that "Market action discounts everything" is known as the most significant rule in technical analysis. The technical analyst believes that anything that can affect the price, such as fundamentals, politics and so on, is reflected in the price of the market. Hence, the technician is claiming that the only attribute beneficial to study is the price action itself. Analysts accept that there are reasons for stock prices going up and down, however, they do not believe this knowledge is necessary in forecasting future prices.

The second statement that "Prices move in trends" is based on the belief that "a trend in motion is more likely to continue than to reverse"[2]. This means that if a stock's price has been steadily increasing then it is more likely to continue in this trend than to begin decreasing. The third statement that "History repeats itself" is suggesting that human psychology does not change. It is loosely stating that the future is simply a repetition of the past.

One very important theory in technical analysis is the idea of "mean reversion". This theory suggests that prices and returns eventually move back towards the mean or average.

One of the technical factors I have assessed for each company is Moving Average Convergence Divergence (MACD). MACD is a trend-following momentum indicator that shows the relationship between two moving averages of prices. The most common form of MACD is calculated by subtracting the 26-day Exponential Moving Average (EMA) from the 12-day Exponential Moving Average. The Exponential Moving Average applies weighting factors for the stock price. The weighting for each day decreases by a factor, or percentage, on the one before it. Therefore a 12-day EMA will take into account the price movement of the previous 12 days, with the previous day having the highest weighting on the average. The weighting then decreasing by a factor on the 2nd previous day and so on.

2.1.3 Trading Strategies

The two main ways of investing in stocks is:

- **Buying Long:** This involves buying a stock at a particular price and making profit if the stock price increases and making a loss if the stock price decreases.
- **Selling Short:** This involves investors selling an equity that they do not own. If the price of the stock decreases then the investors can buy the stock at a lower price than that they sold at. However, if the stock value increases, the investors are liable to incur a loss, because they will have to buy the stock at a higher price than the price they sold at. In essence, it is the opposite of buying long.

2.2 Genetic Programming

2.2.1 Genetic Algorithms

Genetic Algorithms are heuristic search algorithms that are based on the idea of natural selection [3]. Genetic Algorithms are designed to carry out processes in the form of Charles Darwin's theory of survival of the fittest. The following describes the algorithms carried out:

1. A random population is generated P(0). Each member of the population is a string representing a possible solution and is a point in the search space.

Begin Loop:

2. Generate and save the fitness of f(p) for each chromosome p in the population P(x).
3. Outline selection probabilities for each chromosome p in P(x), in general Fitness Proportionate selection is used. This involves a fitter chromosome having proportionally a higher probability of being selected, than a chromosome of lower fitness.
4. Produce P(x+1) by selecting chromosomes, using probabilities defined earlier, in order to produce chromosomes for the next generation, using genetic operations, such as crossover and mutation. They are explained in Section 2.2.4.
5. Continue loop until satisfactory solution obtained.

2.2.2 Genetic Programming (GP) Introduction

Genetic Programming is an extension of the original Genetic Algorithm. Genetic Programming uses tree representations for manipulating populations as opposed to strings.

Genetic programming is a method that genetically produces a population of trees representing simple computer programs to solve a problem. A genetic program step by step transforms a population of computer programs (also known as “chromosomes”) into a new generation of computer programs by applying genetic operations.

The main stages in a genetic program are:

1. Random Initialisation
2. Fitness Function
3. Selection
4. Recombination
5. Replacement

A genetic program starts with an initial population of small programs that consist of functions and terminals that are appropriate to the problem. Usually, the programs are different from each other in terms of shape and size where they contain a different amount of functions and terminals and arrangement. Each individual small program in the population is executed by the fitness function. Each little program is given a fitness score which is stored somewhere. The fitness score is a way of comparing each program in terms of how well they perform the task they are required to do.

In the beginning the production of the initial random population is a very random exploration of the search space of the problem. The initial random population is known as generation 0. In general the individual programs in generation 0 have a poor fitness. Although, obviously, some programs have a better fitness than others do. This difference in fitness between the individual programs is exploited by genetic programming, where genetic programming applies Darwinian selection (“Survival of the Fittest”) and genetic

operations to produce a new population of individual programs from the current population.

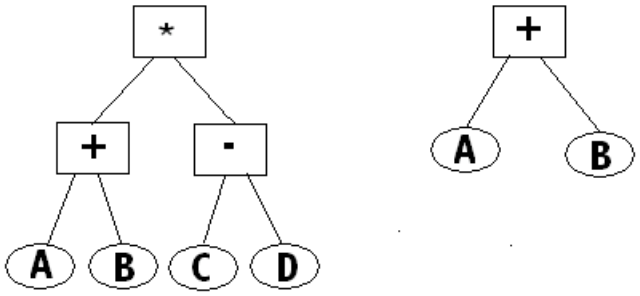
The genetic operations that are used are selection, recombination (crossover), mutation and replacement, and then the alteration of the population. Selection is used first; individual programs are probabilistically selected from the population based on fitness. In general, when using a probabilistic selection process, better individuals are favoured over inferior individuals. However, this does not mean the best individual is necessarily selected and the worst individual is ignored. In addition to selection, elitism is often used to ensure that the fittest individual(s) survive(s) into the next generation.

I will describe all of these genetic operations in more detail later on. After they are performed on the current population, the population of offspring replaces the current population. This iterative process is performed over many generations and the programmer can specify the maximum repetitions there will be. The genetic programming run terminates when a satisfying solution is obtained or the maximum number of repetitions is reached. The best chromosome (individual program) ever encountered during the run is typically designated as the result of the run.

A fundamental factor of genetic programming is that every program in the initial random population is a syntactically valid and executable program. This is to ensure that when genetic operations are performed during the run, offspring is created that is syntactically valid and executable.

2.2.3 The Production of the Initial Programs

Genetic Programming starts with a large amount of randomly generated computer programs. The randomly created programs have different sizes and shapes. These programs can be expressed as trees, as Figure 2.2.3 depicts.



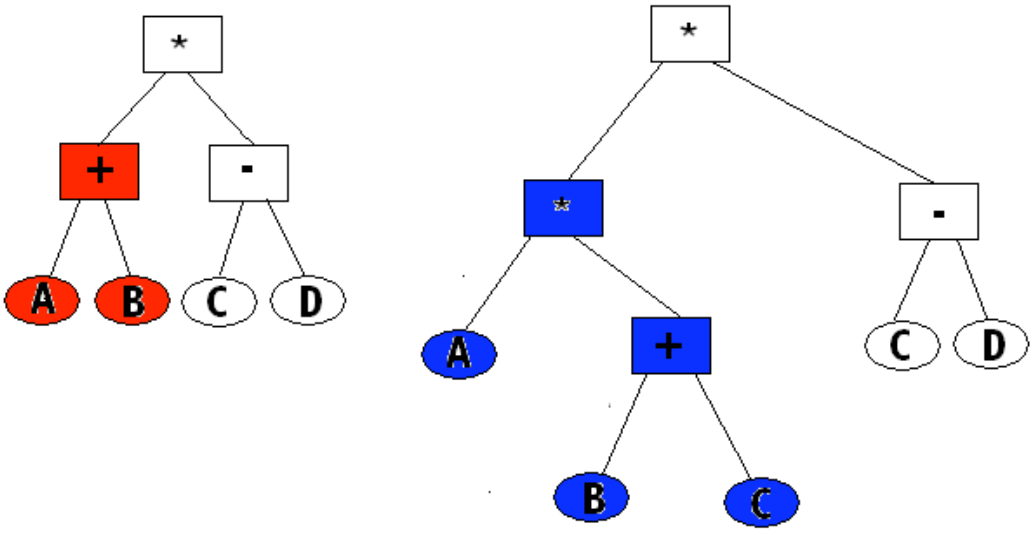
A+B * C-D A+B

Figure 2.2.3 expresses programs as trees. Both have different sizes.

2.2.4 Genetic Operations

2.2.4.1 Mutation Operation

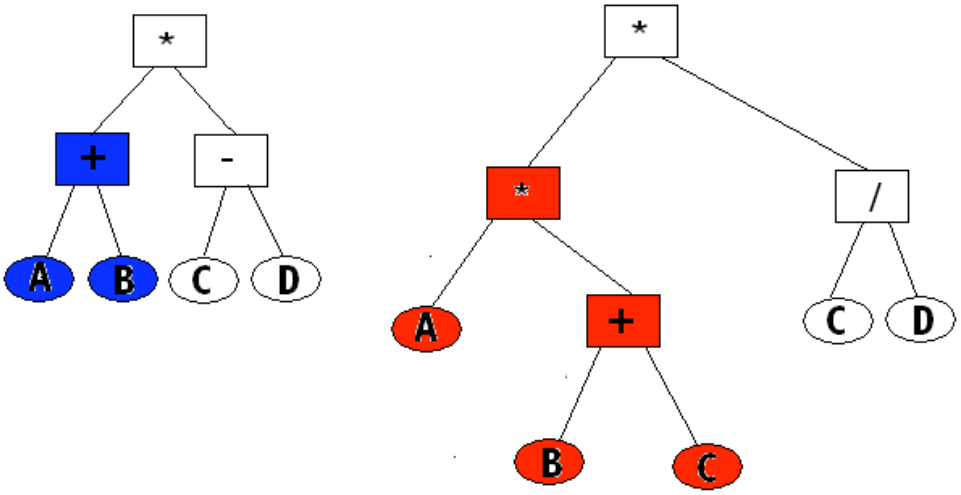
A mutation operation is performed on a single program that has been selected probabilistically from the population based on fitness. A point in the program, which is known as the mutation point, is selected randomly and the subtree at this point is deleted, being replaced by a new subtree. This new subtree is created using the same process that is used to generate the initial population (completely random subtree). The probability of mutation is 1% during each generation of the run. An example of a mutation operation is shown below:



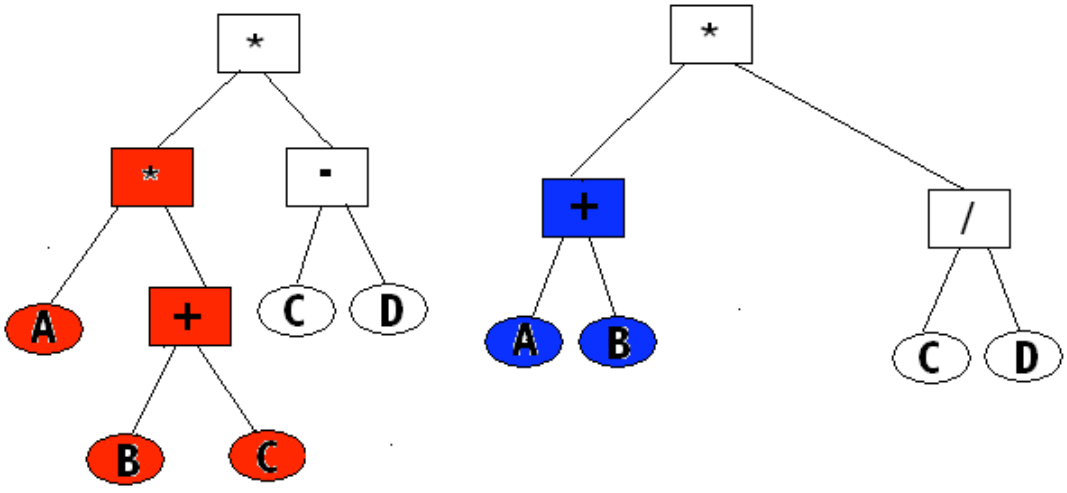
(A+B)* (C-D) becomes (A*(B+C)) *(C-D)

2.2.4.2 Crossover Operation

The crossover operation is also known as sexual recombination. Two programs are chosen probabilistically depending on their fitness, from the population. During the crossover operation it is very possible that two programs are selected which contain a different size and shape. Each program is known as a parent in crossover. Similar to mutation, a crossover point is randomly chosen, but here, a crossover point is randomly chosen in the first program, and a crossover point is chosen in the second program. In crossover, the subtree at each of the two parents is switched to create two new offspring.



These two parent programs, above, crossover, to become:



2.2.4.3 Reproduction Operation

This operation simply copies an individual program that has been selected probabilistically based on fitness into the next generation of the population.

2.2.5 Other Techniques for generating Non-Linear Factor Models

This project has used Genetic Programming to generate non-linear factor models. However, there are other options that could have been used to generate non-linear factor models. Here, I will explain the other options and why a Genetic Programming System was employed in order to develop a workable solution.

The three main search methods that can be used in order to determine non-linear relationships between factors are:

- *Calculus-based methods*: These methods work by finding local maxima from looking for the gradient from all directions leading from the current search point [4]. This method has many drawbacks. The main drawback is that this method concentrates on finding the local optima, but does not concentrate on the global anatomy of the search space and as a result often misses the global optima. This method is in essence very exploitative in its local environment.
- *Enumerative approaches*: A subset of enumerative approaches is dynamic programming. Examples include breadth-first and depth-first searches. All parts of the search space are acknowledged in order to find the optimum solution. This kind of approach is regarded as a very exploratory search as all points in the search space are visited in order to find a solution. It is very inefficient and breaks down for even modestly sized search spaces [4].
- *Random algorithms*: Certain random algorithms exist that support a very exploitative search through a search space. An example of a random algorithm is a Genetic Algorithm.

I have chosen Genetic Programming, which is an extension of Genetic Algorithms, because the selection, recombination (crossover) and mutation methods allow a very efficient balance between exploration and exploitation. The selection process accelerates

the search towards the fittest chromosomes, providing a form of exploitation. The mutation operator modifies certain chromosomes randomly in order to introduce extra diversity into the population. The mutation operator is the exploration operator due to the extra diversity that may have been lost during the selection phase. Recombination changes the context of information that is already present and provides a different form of exploration.

3. Design

This chapter explains in-depth the design of the investment simulator and the genetic program. The design of the historical data used in the project is also included.

3.1 Definition of Problem

The main target of this project is to produce a Genetic Programming System that searches for a nearly optimal non-linear formula that implements a multi-factor investment model. The fittest formula in the population will be used to make trading decisions when managing a real portfolio.

In order to find this formula the system undergoes training; in the Training Period. An investment simulator is incorporated into the system to model how a real portfolio manager would perform in real-life. I am using real fundamental and technical data for 80 UK equities from the FTSE 100 index. Based on each company's fundamental and technical values, investment decisions will be made for each company.

In order to differentiate between good formulae and bad formulae, a form of evaluation will be incorporated into this system. After the investment simulator has been used for a particular formula, a Sharpe Ratio is acquired for that formula. The Sharpe Ratio is a measure of risk-adjusted return on investment, which will be explained in Section 3.5.4. The formula with the highest Sharpe Ratio is the fittest chromosome in the population. The Genetic Programming System will run for 50 generations and the fitter chromosomes from one generation will have a greater chance of surviving into the next generation population. After 50 generations the formula with the greatest Sharpe Ratio, which is the fittest chromosome, will be collected from the system.

The system design should primarily aim to achieve the best possible formula, but also needs to address issues of performance.

3.2 The Design of Data

A very important part of my system is to define how I am going to collect and manage the historical data for my system. The data must be split into:

- in-sample training data (The Training Period), to generate the fittest chromosome.
- out-of-sample test data, to validate the fittest chromosome on data it hasn't seen before.

The time period used for the Training Period is:

02 January 2004 - 30 January 2004

This eliminates all weekends and bank holidays between this period as the London Stock Exchange is closed for trading at these times. Therefore this period constitutes 21 days of trading. The time period between **02 February 2004 – 10 February 2006** will be used to test the fitness of the chromosome acquired from the Training Period.

As this project is on the FTSE 100 stock index, data for a large amount of stocks is required. I have decided on collecting data on 80 of the stocks from the FTSE 100. In general the FTSE 100 stock exchange tends to have around 100 stocks. However, due to the FTSE 100 reshuffle (occurs quarterly) which means certain stocks leave and others join the stock index, it is not feasible for me to include all stocks. Also, certain stocks are members of the FTSE 100 but have not been floating on the stock exchange for more than two years. Therefore, not enough data is available for these stocks. I have defined the stocks that I am going to use for the experiment, in the appendix.

For each stock a certain amount of fundamental and technical data is required as discussed earlier in Section 2.1. The 9 factor models (terminals) I have decided to use in my experiment are:

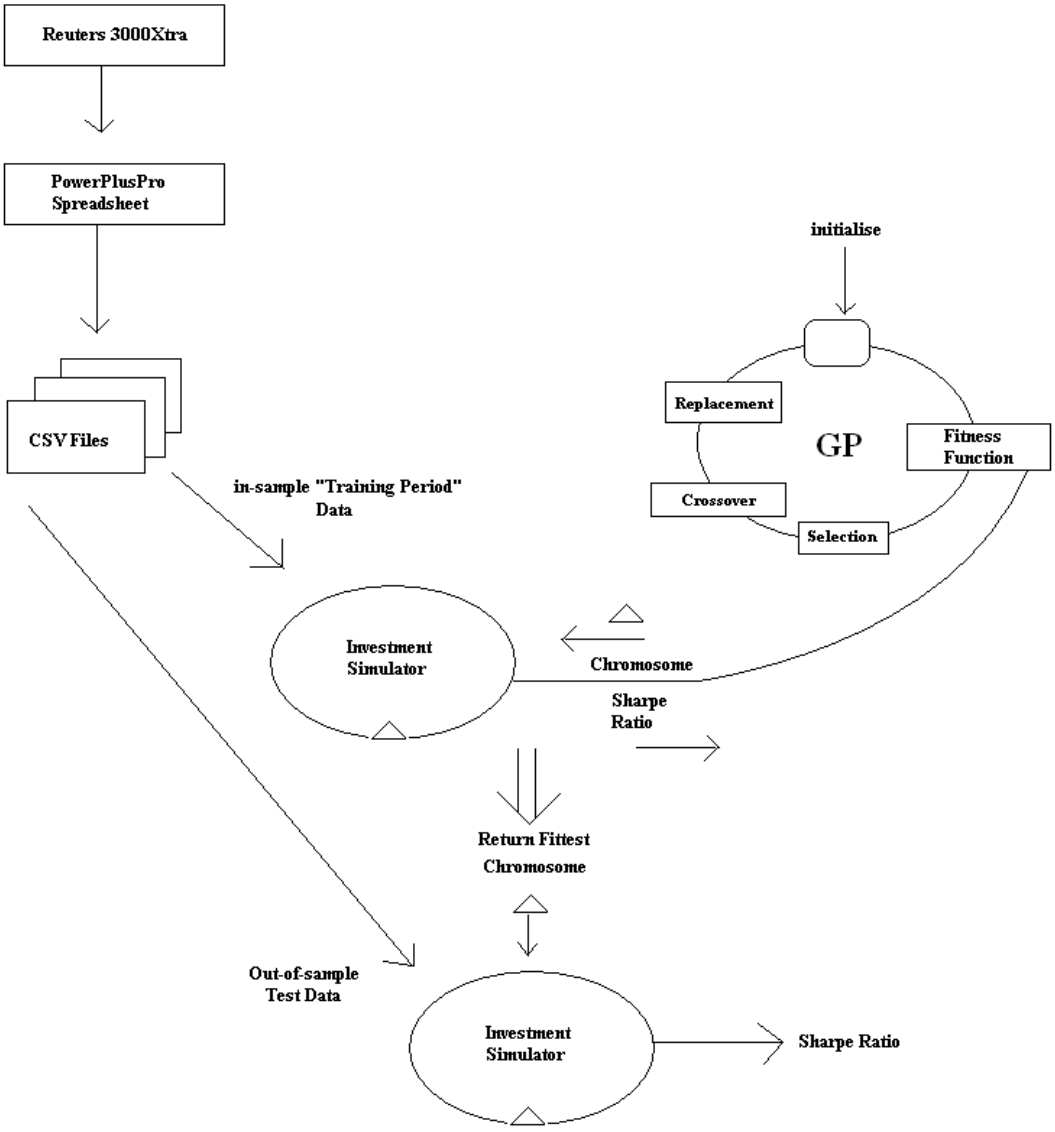
1. **Moving Average Convergence Divergence (MACD)** – See Section 2.1.2 for an explanation.
2. **Price** – The closing price of a stock on a given day.
3. **Book Value Per Share (BVPS)** – A measure to determine the level of safety associated with each individual share after all debts are paid. In essence it represents the amount of money that the holder of a share would receive if the stock was liquidated.
4. **Volume** – The number of shares traded in a stock on a given day.
5. **Earnings Per Share (EPS)** – The portion of a company's profit allocated to each outstanding share of common stock.
6. **Return On Equity (ROE)** – Indicates how much profit a company generates with the money shareholders have invested in it.
7. **One Month Price Momentum** – The rate of acceleration of a stock's price over the previous month.
8. **One Year Price Momentum** – The rate of acceleration of a stock's price over the previous year.
9. **Dividend** – Indicates the portion of a company's earnings that are distributed to its shareholders.

Each one of these factor models will be a terminal in the GP System (refer to Section 3.4.2).

There is likely to be a degree of complexity in obtaining the data. I have access to Reuters 3000Xtra, which uses PowerPlusPro a spreadsheet application. This application will allow the collection of all the necessary data for the stocks. However, there is going to be a need to program a spreadsheet in order to collect the data.

3.3 Overall System Diagram

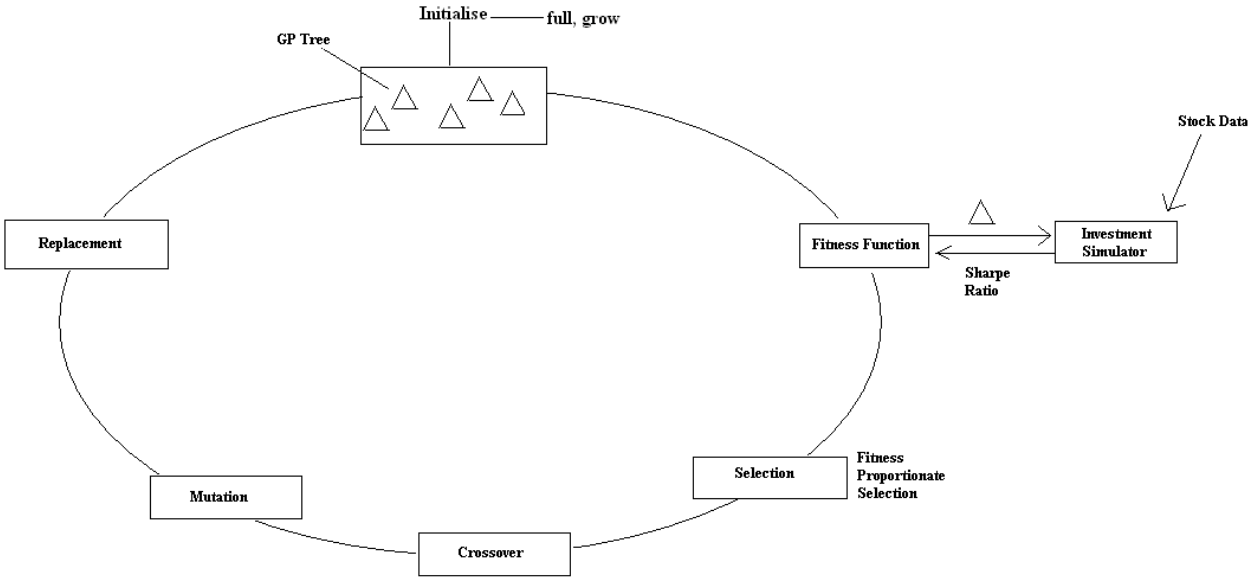
This diagram provides an abstract representation on the design of my system.



1. I will produce a Reuters Spreadsheet with stock data. From this I will produce a CSV file for each stock in sample.
2. I will produce an investment simulator to operate on this stock data.
3. A GP will generate chromosomes (Non Linear Factor Models) to be used by the investment simulator. The investment simulator will return the Sharpe Ratio for each. This part is known as the Training Period.
4. The fittest chromosome will be returned from the Training Period and tested on out-of-sample data. A Sharpe Ratio will be generated for this chromosome.

3.4 GP System Design

The diagram below describes how I have decided to design my GP system in order to develop a workable solution:



3.4.1 Representation

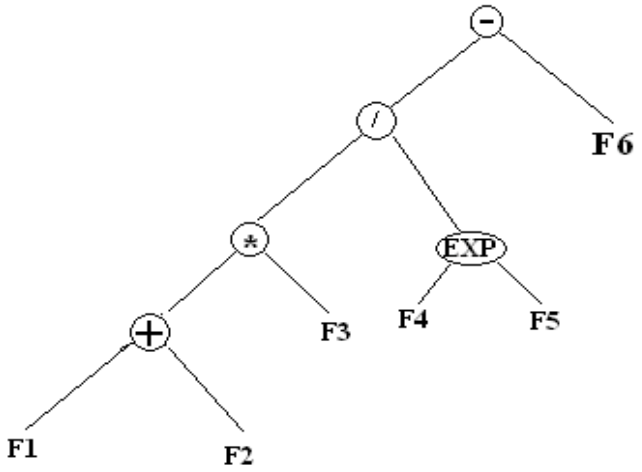
In my GP system a single GP chromosome is a tree that represents a non-linear equation combining factors. An example of a factor is "Price" (the closing price for a stock on a given day). Factors are represented as terminals in my program.

A single GP chromosome takes the factors for a single stock, uses the tree to combine them in a non-linear way, and produces a number that tells the portfolio manager whether this stock's price is likely to increase (Buy long) or the opposite (Sell short). A single GP chromosome (tree) does not need to use all the factors.

An example of a non-linear equation:

$$(\frac{(F1+F2)\times F3}{F4EXPF5})-F6$$

Functions used are +, -, *, /, EXP. The linear equation produces the following tree structure (a chromosome):



3.4.2 Terminals

The terminals to be used in the system are the 9 factor models that have been stated in section 3.2. All these terminal values will be obtained for each stock in the sample. They will be obtained using Reuters 3000Xtra. There is no requirement for all terminals to be used in a chromosome. One terminal on its own is sufficient to constitute a chromosome. Also, a terminal can be repeated more than once in a chromosome. Each terminal points to a value. This is because each stock has a terminal value for each day of trading.

3.4.3 Functions

These are arithmetic operations that are used to compute the terminals. The arithmetic operations that I will be using in my system are: +, *, -, / and EXP.

3.4.4 Chromosome Creation

The genetic program begins with the creation of an initial population. The initial population consists of simple programs represented as expression trees comprising functions and terminals. The creation of the initial population is a random production of

chromosomes from the search space. It is the basis of future chromosomes, as all chromosomes are adapted from these ones. In general most chromosomes in generation 0 will have a poor fitness. The GP System will use a population of 50 chromosomes. Each one of these 50 chromosomes will be executed in generation 0, and all 50 chromosomes will be executed each generation thereafter.

Each chromosome in the program is given a value by running the investment simulator. The investment simulator will return a Sharpe Ratio, which is the value given to the chromosome.

3.4.5 Fitness Function

As stated before when a chromosome has passed through the investment simulator, it is given a value, which is known as the Sharpe Ratio. The Sharpe Ratio calculation is explained in section 3.5.4. The higher the Sharpe Ratio the fitter the chromosome. Then the Sharpe Ratio is returned to the fitness function. The chromosome with the highest Sharpe Ratio will be returned at the end, as this is the fittest chromosome generated. As each generation is executed, chromosomes are likely to become fitter. Therefore, the fittest chromosome is more likely to come from a later generation. However, the fittest chromosome can be derived from any generation.

3.4.6 Selection

The selection method that is going to be used is Fitness Proportionate Selection. This means that the fitter a chromosome the higher the probability that a chromosome has of being selected for a genetic operation, such as crossover or mutation. Fitness Proportionate Selection allows for chromosomes that have a low fitness to be selected, but ensures that the fitter chromosomes are selected on a more frequent basis.

3.4.7 Crossover

As explained earlier crossover is a genetic operation. Crossover is applied to chromosomes in the population. The crossover operation is performed in the same way as explained in the background section. Two chromosomes are selected. A subtree from each chromosome is selected and swapped over. As each subtree contains terminals and functions; terminals and functions from each chromosome is swapped. For example:

Chromosome 1 = (Price * (MACD-Volume));

Chromosome 2 = EPS – BVPS;

Performing crossover on these two chromosomes could produce:

Chromosome 1 = Price * BVPS;

Chromosome 2 = EPS – (MACD-Volume);

This is just a simple example, and in the system the crossover will be of a greater complexity, where more terminals and functions are likely to be exchanged.

3.4.8 Mutation

This is similar to crossover except it is an asexual operation. When a chromosome has been selected a subtree will be chosen within the chromosome. This subtree will be replaced with a new random subtree, therefore producing a new chromosome that will be replaced back into the population. Mutation must be performed in a way that the original structure of the chromosome is preserved and only the subtree that is replaced changes the chromosome in any possible way. The replacement subtree can be a combination of any of the terminals and functions from the search space.

3.4.9 Reproduction

Reproduction involves a chromosome being put back into the population for the next generation. For example:

Chromosome 1 = Volume;

If reproduction is performed on this chromosome then it will simply be replaced back into the next generation.

3.4.10 Replacement

After the genetic operations have been performed on the current population, the old chromosomes are replaced with the new chromosomes. There are 50 chromosomes in a population. Therefore the 50 chromosomes in generation 0 will be replaced by a new set of chromosomes after the genetic operations have been performed. These new 50 chromosomes will provide the new population for generation 1. The replacement function may also incorporate elitism. This involves the fittest chromosome(s) from a generation being replaced into the next generation.

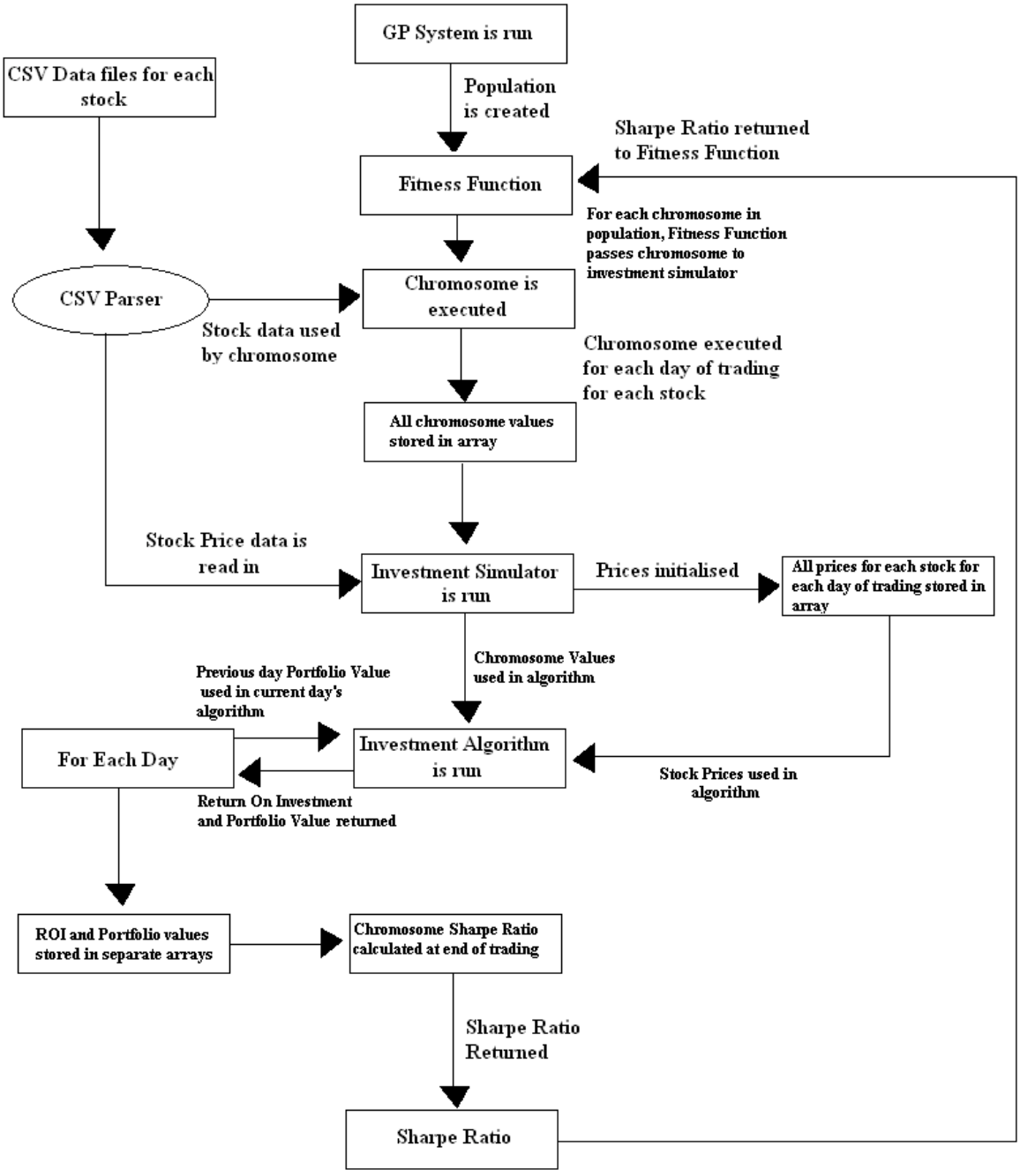
3.4.11 Program Termination

I am running this system for 50 generations. This ensures that after this amount of generations, there will be a chromosome acquired that is the fittest chromosome in the population. The chromosome received is the fittest chromosome from any generation. I have decided to specify the number of generations in order to ensure that I receive a chromosome within a certain time frame. An alternative would be to stop when a chromosome reaches a certain level of fitness. However, it is not known how long this could take, and computer resources are limited.

3.5 Investment Simulator Design

3.5.1 Investment Simulator Design Structure

The diagram below depicts the Investment Simulator design structure and how it incorporates with the GP System fitness function.



3.5.2 Investment Simulator Summary

Before explaining the design of the Investment Simulator algorithm in detail, I am going to provide a summary of what it essentially does, when a chromosome is sent to it:

1. The algorithm will run for the duration of 21 days of trading.
2. The algorithm will be executed for 80 stocks from the FTSE 100 stock index.
3. The portfolio value at the start is £100,000.

Every Day (Repeat for 21 days):

At the start of the day:

4. 80 chromosome values will be extracted (one value for each stock and values are different every day).
5. These 80 values are sorted in descending order; the highest value is the maximum and the lowest value is the minimum. Based on these 80 values buy and sell decisions are made.
6. Counter = Number of stocks with values in top quartile + Number of stocks with values in bottom quartile (Top quartile and bottom quartile of the 80 values)
7. The cost of each stock to buy long or sell short is the purchase amount.
   Purchase Amount = Portfolio Value (at close previous day)/Counter.
8. Those stocks with chromosome values in the top quartile are bought long.
9. Those stocks with chromosome values in the bottom quartile are sold short.

At the end of the day:

10. All stock positions closed at the end of the day and profit/loss calculated for each stock.
11. New Portfolio Value returned, except on day 0 = 100,000.
12. Return On Investment (ROI) = Portfolio Value (today) – Portfolio Value (yesterday). ROI = 0 on the first day of trading (known as day 0). ROI stored in array.

After 21 days:

13. Using the Return on Investment for every day (use array), the Sharpe Ratio is calculated and returned.

3.5.3 Investment Simulator Algorithm

Generating Chromosomes:

When the GP System runs, an initial population is created. A chromosome is selected from the population. A chromosome is a multi-factor expression that calculates a value for a given stock on a given day (for example **Price + Volume)**; ChromosomeValues[stock][day] stores those values. "Stock" depicts the FTSE 100 company the chromosome value has been calculated for. "Stock" has a size of 80, as there are 80 companies used in this sample. "Day" depicts the day the stock value has been calculated for, and has a size of 21 (21 days used in the sample). Hence, ChromosomeValues[0][4] is the calculated chromosome value for Anglo American (stock 0 in the sample) on day 4 of trading, for a particular multi-factor expression. This array is initialised while the chromosome is being executed.

Price Array:

Due to the number of stocks and days of trading, reading stock files while running the investment simulator will slow down the whole system. Therefore, a two-dimensional price array is calculated in the same way as the ChromosomeValues array. The price array is used to store each stock's price for each day of trading:

Price[stock][day]

Hence, while the investment algorithm is running, when a stock's price is required for a day, it is retrieved from the array as opposed to reading from that stock's CSV file.

The Investment Algorithm:

The explanation of the Investment Simulator as outlined in Section 3.5.2 is provided in more detail by the following pseudocode:

```
PortfolioValue = £100,000(for Day 0)
For Every Day:
    Read ChromosomeValues[0-79][currentDay]
    Store today's 80 values in a new single array: StoreArray[80]
    Put these 80 values in descending order
    StoreArray[0] = max
    StoreArray[79] = min
```

Counter = (number of StoreArray values > (min+(0.75*(max-min)))) + (number of StoreArray values < (min+(0.25 *(max-min))))

For Every Stock:

if stock's chromosome value > (min+(0.75*(max-min))) then BUY

StockValue (PurchaseAmount) = Yesterday's PortfolioValue/Counter;
Retrieve this stock's closing price for yesterday and today, from PriceArray.
StockValue = ((todayPrice – yestPrice)/yestPrice)*StockValue
Return StockValue

ELSE

if stock's chromosome value < (min+(0.25 *(max-min))) then SELL Short

StockValue (PurchaseAmount) = Yesterday's PortfolioValue/Counter;
Retrieve this stock's closing price for yesterday and today, from PriceArray.
StockValue = ((yestPrice – todayPrice)/yestPrice)*StockValue
Return StockValue

Today's PortfolioValue = sum of all company StockValues (all 80, stocks not bought/sold will have a value of 0)

PortfolioValue stored in array (day 0 = 100,000 always)

ReturnOnInvestment = Today's PortfolioValue – Yesterday's PortfolioValue

ReturnOnInvestment stored in array.

Fitness of chromosome is calculated (Sharpe Ratio) and returned.

Initialisation:

The system will ignore all transaction costs (and dividends). Therefore, as an optimisation, the Investment Simulator will close out all long (stocks owned) and short (stocks owing) positions at the end of each day. At the start of day 0, the value of the Portfolio is £100,000. At the start of each day the Portfolio has a cash value only (because all stock positions were closed at the end of the previous day).

PortfolioValue[day] is an array that records the value of the portfolio at the end of each day, after all stock positions have been closed (i.e. long positions sold and short positions bought). Day 0 = 100,000, effectively no stocks are bought or sold on this day.

Buying (Long) and Selling (Short) strategy:

At the start of each day, the chromosome values for that day are inspected. The investment strategy is to buy shares for all stocks whose ChromosomeValues[stock][today] is in the top quartile (i.e. the top 25% of all chromosome values for that day), and to sell shares for all stocks whose

ChromosomeValues[stock][today] is in the bottom quartile (i.e. the bottom 25% of all chromosome values for that day) - since positions are closed out at the end of every day, this will always be selling "short" (i.e. selling shares that are not currently owned).

Calculating Profit:

The funds available for buying stocks at the start of each day are given by the PortfolioValue of the end of the previous day (i.e. PortfolioValue[day-1]). This amount is divided by the number of stocks to be purchased (the "buy" stocks), and an equal cash amount is available to purchase shares in each of these buy stocks. Thus, if the number of buy stocks is called BuyCounter, then the cash amount available to purchase each buy stock is:

PortfolioValue[day-1]/BuyCounter

Shares are bought at the start of the day (at yesterday's price) and sold at the end of the day (at today's price) when all positions are closed. The profit (or loss) made during the day on each buy stock is calculated as the amount invested multiplied by the percentage change in the price of that stock:

((PortfolioValue[day-1]/BuyCounter)*((Price[x][day]-Price[x][day-1])/Price[x][day-1]))

Since all sales are "short", sold at the start of the day (at yesterday's price) and then bought back at the end of the day (at today's price), the profit calculation can be simplified by treating them as "buy" stocks and then negating the change in price when the position is closed at the end of the day. However, in order to guard against the Portfolio Value becoming negative due to short selling, we wish to limit the amount of shares sold short and also hold back some cash in reserve. This is achieved very simply by using Counter (the total number of stocks in both the top and bottom quartile - i.e. the total number of buy long stocks and sell short stocks). This is used instead of BuyCounter (the number of buy stocks only) in the above equation. This equation is used to buy stocks and sell short stocks:

Profit for a single buy stock x = ((PortfolioValue[day-1]/Counter)*((Prices[x][day] - Prices[x][day-1]) / Prices[x][day-1]))

Profit for a single sell short stock x = ((PortfolioValue[day-1]/Counter)*((Prices[x][day-1] - Prices[x][day]) / Prices[x][day-1]))

Portfolio Value and Return On Investment Calculations:

The new PortfolioValue at the end of a day (PortfolioValue[day]) is:

PortfolioValue[day-1] + (profit/loss for all buy stocks) + (profit/loss for all sell short stocks)

ReturnOnInvestment[day] is an array that records changes in the portfolio value from day to day:

ReturnOnInvestment[day] = PortfolioValue[day] - PortfolioValue[day-1]

An exception is that ReturnOnInvestment[0] = 0. The Return on Investment array is used to calculate the Sharpe Ratio after the 21 days have executed.

3.5.4 Sharpe Ratio

$$SharpeRatio = \frac{\overline{ROI} - RFR}{\sigma}$$

Figure 3.5.4 Sharpe Ratio Formula

The Sharpe Ratio is a measure of risk-adjusted performance of an investment asset [5]. The ROI (Return On Investment) in the investment simulator measures the change in value of the portfolio over a day. The ROI is measured every day. To measure the Sharpe Ratio the mean ROI is required. Therefore, a mean ROI for all trading is calculated. However as stated earlier, for day 0 the ROI is 0, and this day is not included in the calculation of the mean. A sum of the ROI's from day 1 to day 21 is returned. This value is then divided by the sample (because day 0 is ignored, therefore only 20 days used in sample) to return the mean.

$$\overline{ROI} = \frac{(ROI[1] + ROI[2] + ... + ROI[20])}{20}$$

The standard deviation refers to the standard deviation of the ROI. To calculate the standard deviation, the variance is computed first. For every day in the sample the mean is subtracted from the ROI. This value is then multiplied by itself (the power of 2) to return a variance value for a particular day. Again the ROI for day 0 is ignored (only day 1 to day 21 used).

$$Variance[day] = (ROI[day] - \overline{ROI})^2$$

The sum of all variances is returned to give the variance of the sample. This variance value is then square rooted to give the standard deviation.

$$Variance = Variance[1] + Variance[2] + ... + Variance[20]$$

$$\sigma = \sqrt{Variance}$$

RFR is the Risk Free Rate. This is the rate of return someone would expect when putting his or her money in a bank instead of investing. The value used for RFR is 4% per annum 0.04/365 per day.

$$\textbf{RFR} = 0.04/365$$

To return the Sharpe Ratio for a chromosome the final calculation is computed (Figure 3.5.4). The R.F.R is subtracted from the mean ROI. This value is then divided by the standard deviation to deliver the Sharpe Ratio. The Sharpe Ratio is returned into the fitness function. The higher the Sharpe ratio the fitter the chromosome. Once the Sharpe Ratio is returned the next chromosome is executed. 50 chromosomes are executed per generation.

3.6 Design Limitations

There are obviously many limitations to the design of this system. The investment simulator is based on actual trading techniques employed by professionals but in the process of modelling these professional trading techniques it is very improbable that the system will be as accurate as real trading systems. The following limitations were imposed:

1. There were only 80 companies from the FTSE 100 stock index used in the sample. Whereas the FTSE 100 contains around 100 of the largest companies in the London Stock Exchange. This was mainly because companies move in and out of the FTSE 100 during the FTSE100 reshuffle, which occurs four times a year (quarterly).
2. I only included end of day closing prices for each stock. In the London Stock Exchange stock prices are changing every second when the stock market is open for trading. However, to employ historical prices for each company for every second of trading over the past few years was not feasible. Therefore, even though, professionals buy long and sell short on stocks in less than a second of trading, my system only allows this option once a day per stock.
3. I have also ignored transaction costs. For every transaction made there is a broker fee. The size of this charge depends on the stockbroker.
4. Professionals who work in investment banks or other financial firms believe in diversifying their portfolios. Therefore, they prefer to have stocks from many different sectors. This ensures that if one sector such as oil performs very badly then there will be other sectors that will consolidate the portfolio. In my portfolio I have not allowed for this. I believe that this slightly limits the results that can be achieved in the system.
5. I am also only using 21 days of data for training, which constitutes a month of trading. This is due to a limit in computer resources. The computational cost of the Training Period is very high.

4. Implementation

This chapter characterises how the design was implemented. Therefore, the implementation of the data, investment simulator and the GP system is included.

4.1 Introduction

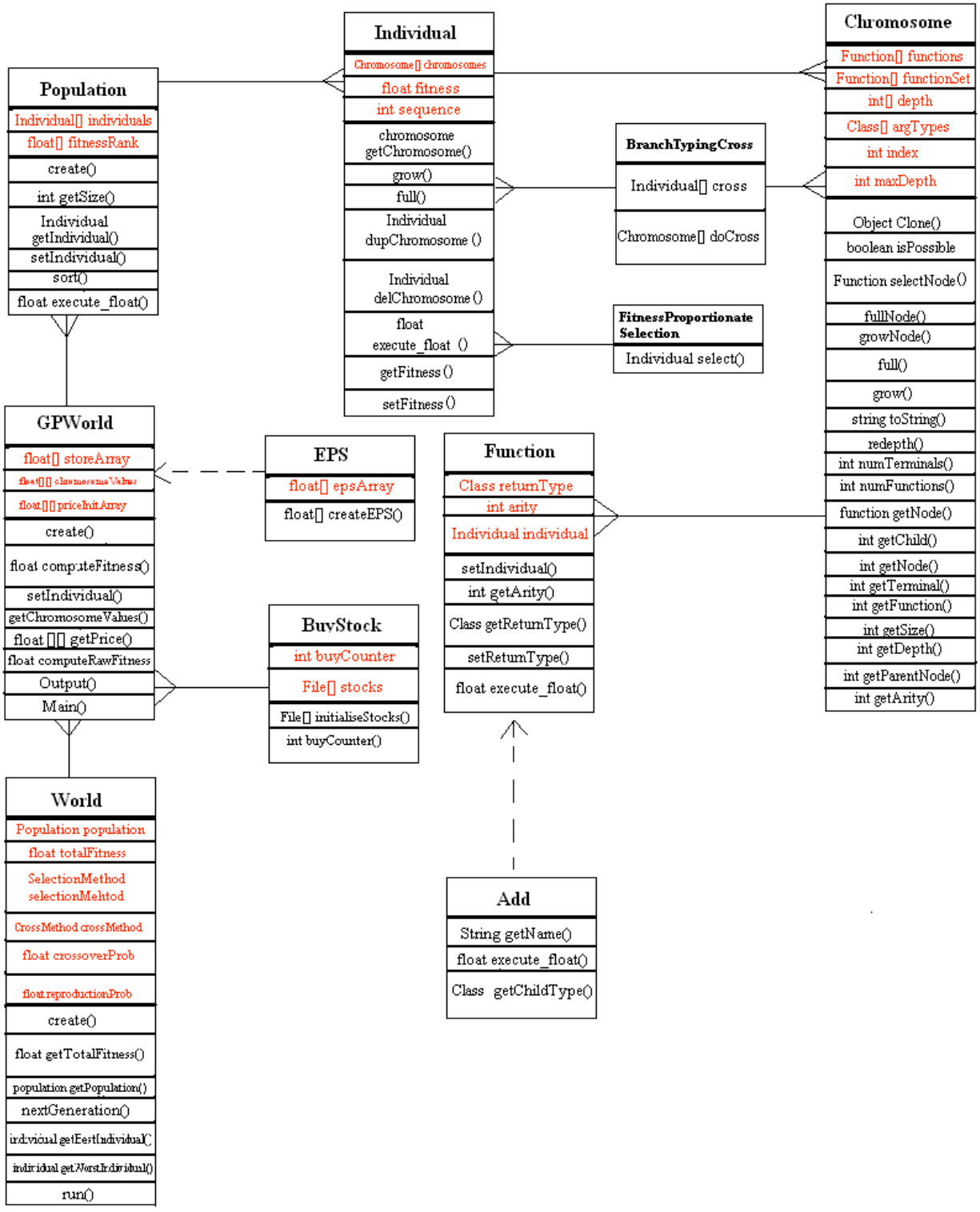
The GP System I am using is written in Java. It is based on jgprog, version 1.4, by Robert Baruch (2000). However, the program has been heavily modified in order to provide the correct implementation. I have customised the tree representation and the fitness function. I have produced an Investment Simulator (IS), which has been incorporated in the fitness function. The IS has also been combined with a CSV parser in order to read in stock data.

In this chapter I will discuss how the system is implemented, the structure of the classes and the testing of the system.

4.2 Development Tools

The Genetic Programming System was written in the Java programming language along with the IS that was incorporated with the GP System. I chose Java because a GP System was already available which I could extend. The encapsulation provided by the class mechanism would help organise the merging of the IS with the GP System. I have two years of experience of coding in Java. Other options that were available to me, I have little or no experience in the use of. I used Reuters 3000Xtra in order to acquire data on the stocks used in the Investment Simulator. The data was obtained in spreadsheet format and the spreadsheet used was PowerPlusPro.

4.3 Class Diagram



The diagram above is a class diagram of the whole system, depicting the main classes.

4.3.1 Class Explanation

1. The World class is an abstract class that defines all the main running characteristics of the Genetic Program System.
2. The GPWorld class extends the World class. The main purpose of the GPWorld class is to return fitness values. Therefore, the GPWorld class has the investment simulator implemented inside it. The GPWorld class is the main class that creates the population.
3. EPS is a terminal in the system. This class is created to retrieve all EPS data. All terminals have a separate class that retrieves data. The GPWorld class calls the terminal classes to retrieve data while running the investment simulator.
4. Add is one of the functions in the system. All arithmetic operations have a class that extends the function class.
5. Function is an abstract class that represents all functions.
6. The Population class depicts the population of individuals in the system.
7. The Individual class represents a GP Individual containing a GP Chromosome. This class is used to perform the crossover operation on chromosomes.
8. The Chromosome class represents a Non-Linear Factor Model (GP Chromosome).

4.4 Genetic Program

4.4.1 GPWorld Class

The GPWorld class is an extension of the World class that defines all the main characteristics of the program. Firstly within the GPWorld class I have defined all 9 terminals. In this class the number of generations and the number of chromosomes per generation is set. When the program is run the population of 50 chromosomes is created and a chromosome is randomly selected. This chromosome is computed by returning a value for each day of trading for each stock and storing in an array (**ChromosomeValues** array). A price array is created in the same way. These two arrays are used in the investment simulator in order to return a Sharpe Ratio for that chromosome. As stated before, the Sharpe Ratio represents the fitness of a chromosome. Therefore, once a Sharpe Ratio has been returned for a chromosome it is returned to the fitness function.

Then the next chromosome is computed. This is repeated for all 50 chromosomes in the generation. Then the next generation is computed. This is repeated until all 50 generations are completed. Each fitness that is calculated is stored in an array that has been created in the Population class. The array is known as fitnessRank and stores all the float fitness values. This array is sorted in the population class, and at the end of a generation the best chromosome is returned. I have modified the World class in order for it to display the best chromosome for every generation at the end of the running of the program. Also the best chromosome from all of the generations is displayed as the fittest chromosome at the end.

4.4.2 GP Implementation

Genetic Program Functions	Modifications
Initialise Population	The initialisation of the population was created in "jgprog" and left without modification.
Fitness Function	The fitness function was modified and extended.
Selection	The selection method was created by "jgprog".
Crossover	The crossover method was created by "jgprog".
Replacement	This was created in "jgprog".
Terminals	I implemented all terminals.
Functions	All functions were already implemented in "jgprog".

Initialise Population: This area was not modified. The population was created in "jgprog" using the ramp half-and-half method. This is a combination of the "full" and "grow" method. When creating chromosomes the "full" method works by selecting only functions until a node is at a specified depth, at which time only terminals are selected. Therefore, every branch reaches a maximum depth. The "grow" method works by randomly selecting functions and terminals until a terminal node is reached. This terminates the branch.

Fitness Function: I created the fitness function within the GPWorld class. The Investment Simulator is run on each chromosome and a Sharpe ratio is returned into the fitness function. When all generations have been executed, the fittest chromosome is returned. This is executed by storing the fittest chromosome from each generation in an

array. After the final generation the array is sorted in descending order with the fittest chromosome at position 0.

Selection: The selection method was created in "jgprog". The selection method used is Fitness Proportionate Selection. This form of selection selects chromosomes proportionally according to their adjusted fitness. The fitter a chromosome the greater the chance the chromosome has of being selected for a genetic operation. Elitism is not supported in "jgprog", therefore, it was not implemented in the system.

Crossover: The crossover method was created in "jgprog". The crossover method used is called "BranchTypingCross". This works by crossing two chromosomes. A random chromosome is chosen. A node is chosen from this chromosome, with 90% probability it will be a function and 10% probability it will be a terminal. A random-point is chosen in the second chromosome is chosen with the same probability distribution, but must be of the same type as the first chromosome. If a suitable point cannot be found in the second chromosome then they are not crossed. If a resulting chromosome's depth is larger than the maximum crossover depth then that chromosome is simply copied from the original rather than crossed.

Replacement: The replacement method was created in "jgprog". The type of replacement used is called "generational". The population is evolved every generation. The system probabilistically reproduces and crosses chromosomes in the population, to create a new population, which then overwrites the original population.

Terminals: I implemented all the terminals myself. Each factor model in the system was represented as a terminal. All terminals have been created as variables in the system. Each variable has a float implementation. Therefore, all execution on terminals should return float values. I created a class for each terminal. This provided a class for the Investment Simulator to access when reading in data of a particular factor model (terminal). I have explained in Section 3.2 the factor models chosen to be terminals.

Functions: All functions used were created in "jgprog". A class has been implemented for each function that has been used in the system. I am only using five functions in my system. Section 3.4.3 states the functions used.

4.4.3 GP Parameters

The parameters declared below are the stated parameters in my genetic program.

Selection Method = Fitness Proportionate Selection
Cross Method = Branch Typing Cross
Crossover Probability = 0.9 (90%)
Reproduction Probability = 0.1(10%)
Maximum Crossover Depth = 17
Maximum Initialisation Depth = 6
Mutation Probability = 0

When a chromosome is selected it has a 90% probability of crossover with another chromosome and a 10% probability of being asexually reproduced. This means that the same chromosome is placed into the next generation. The maximum depth of a chromosome resulting from crossover is 17. The maximum depth of a chromosome during population creation is 6 (generation 0). Mutation was not implemented in the "jgprog" system.

4.5 Investment Simulator Implementation

4.5.1 Investment Algorithm Representation

Investment Simulator Implemented in GPWorld Class:

I created the Investment Simulator algorithm in the main GPWorld class. The GPWorld class extends the World class (which was created in "jgprog"). Therefore, due to the hierarchical structure of "jgprog" it was very difficult to put the Investment Simulator

(IS) in a different class. This is because it would have meant that each chromosome would have to be called in a different class to GPWorld (Each chromosome is created in GPWorld). The GPWorld class extends the World class, in order to provide an implementation of the fitness function. Essentially the IS is the fitness function. This is because the IS returns a Sharpe Ratio for each chromosome. The Sharpe Ratio represents the fitness of a chromosome. Therefore, to simplify the problem the IS is implemented in the GPWorld class.

Investment Simulator Method Implementation:

The method that implemented the Investment Simulator is known as: **computeRawFitness(Individual ind)**. The value returned from this method is a float. This float value returned is the Sharpe Ratio of the chromosome (represented as Individual) being executed. Two arrays created at the beginning of the method are **ReturnOnInvestment** and **PortfolioValue**. The **PortfolioValue[0]** is initialised to 100,000 because the value of the portfolio at day 0 is £100,000. Also **ReturnOnInvestment[0]** is initialised to 0 because there is no return on investment on day 0.

The "getPrice()" method:

The first method called by the Investment Simulator is **getPrice()**. This is used to return an array with all the stock prices, for each stock, for each day of trading in the sample. This method is called every time the investment simulator is computed for a chromosome. This **getPrice()** method is purely employed to reduce the computation time of the IS during runtime. Prices are required in the IS to calculate how much profit is made on stock purchases. As the stock prices are stored in CSV files, continuously accessing these files will slow down the IS. Therefore, employing the **getPrice()** method to store stock prices in an array, allows the IS to continuously access the array as opposed to the CSV files, making it more efficient. The array returned from the **getPrice()** method is initialised as **priceInitArray[80][21]**. This is because there are 80 stocks used in the sample and 21 days of trading. Therefore, the IS can access any price for any stock in the sample.

The "getChromosomeValues" method:

The second method called by the Investment Simulator is called **getChromosomeValues()** and is computed each time **computeRawFitness()** (the Investment Simulator) is called. A two-dimensional array is returned with values computed for the chromosome that is being executed. As stated in the Design section a chromosome is a multi-factor expression that calculates a value for a given stock on a given day. Each terminal has its own class. Each terminal class is used to return an array of terminal values for each stock. Each stock has 21 terminal values (because 21 days are used in the sample). Therefore each terminal array is initialised as **Terminal[21]**. For example Book Value Per Share is initialised as **BVPS[21]**. All terminal values are retrieved from the stock CSV files and stored in their arrays at runtime. When a chromosome is computed, only the arrays for the terminals (factors) that are used in the multi-factor expression are returned. The terminal arrays are returned for each stock in the sample. To make this clearer, assume the chromosome is **BVPS + EPS**. Stock 0 is computed first. An array for **BVPS** and an array for **EPS** is returned (The EPS array contains 21 EPS values for stock 0, and the BVPS array likewise). For each of the 21 days the corresponding values for **EPS** and **BVPS** are added together. These values are stored in **ChromosomeValues[0][day]**. This is repeated for the remaining 79 stocks in the sample. This gives the array **ChromosomeValues[80][21]**.

Strategy for buying (long) and selling (short) stocks:

When in the **computeRawFitness()** (Investment Simulator) method, the values computed for each chromosome is initialised to a new array in the following way:

storeArray[stock] = ChromosomeValues[stock][today]

For every day of trading this initialisation takes place. 80 stocks are stored in **storeArray.** This array is then sorted into descending order with **storeArray[0]** holding the stock with the highest (**maximum**) chromosome value and **storeArray[79]** holding the stock with smallest (**minimum**) chromosome value. Using this array it can be decided which stocks should be bought long and sold short.

If a stock's chromosome value is in the top quartile then buy the stock long on that day.

Stock's chromosome value > (minimum + (0.75*(maximum – minimum)))

If the stock's chromosome value is in the bottom quartile then sell short.

Stock's chromosome value < (minimum + (0.25*(maximum – minimum)))

If a stock's chromosome value is not in the bottom or top quartile it is not bought long or sold short.

Calculating the amount spent on a stock (purchase amount):

The number of stocks that will be bought long or sold short is calculated before any transactions are carried out on a particular day. The **ChromosomeValues[80][today]** is sent to the **BuyStock** class. The **Counter()** method in this class returns an integer counter, to the **computeRawFitness()** class. This counter denotes the number of stocks with a chromosome value in the top quartile plus the number of stocks in the lower quartile. This is in order to calculate how much will be spent on each stock. The amount spent on a transaction for a stock is **PortfolioValue[day-1]/Counter**. Therefore, if the value of the portfolio yesterday is £120,000 and 20 stock transactions are to be completed today then 120,000/20 = 6,000. Therefore £6,000 will be spent on each stock. The amount spent on a stock is called the Purchase Amount.

Buying (long) and Selling (short) stocks:

Once the cost of a stock (purchase amount) has been established, on a particular day, stock transactions are carried out (stocks are bought long, sold short or no transaction performed on them). In order to do this, the simulator loops through all (80) stocks in the array. For stock 0 we check whether its chromosome value is in the top quartile of all stock chromosome values. If it is in the top quartile the stock is bought long, for the purchase amount. If the stock's chromosome value is in the bottom quartile of chromosome values it is sold short, for the purchase amount. If the stock is in neither the top or bottom quartile it is not bought long or sold short. This is repeated for the remaining 79 stocks in the sample.

Calculating Stock Profit/Loss:

Whenever a stock is bought long or sold short, today and yesterday's closing price for that stock is extracted from the **PriceInitArray**. Whenever a stock transaction is made the Purchase Amount is denoted as the current stock value. Therefore, if £6,000 is spent on each stock (purchase amount), at the start of the day the stock's value is £6,000. At the end of the day the stock's value depends completely on that stock's price movement (on that particular day). The stock value calculation for buying a stock long is different to selling a stock short. The calculation for buying long is:

PriceChange = ((Today's closing price – Yesterday's closing price)/Yesterday's close)

NewStockValue = PurchaseAmount*PriceChange

The calculation for selling short is:

PriceChange = ((Yesterday's closing price – Today's closing price)/Yesterday's close)

NewStockValue = PurchaseAmount*PriceChange

The **PriceChange** calculation is different when buying long to selling short. This is because essentially when a stock is sold short, profit is made if its value decreases. When buying a stock long, profit is made if its value increases.

Calculating the Portfolio Value every day:

The new **PortfolioValue[day]** is calculate incrementally. When the **NewStockValue** is returned for a stock it is added to **PortfolioValue[day]**. Once the Investment Simulator has looped through all 80 stocks in the sample, all **NewStockValue** will have been summed together and the **PortfolioValue[day]** is returned for that particular day. On day 0 the **PortfolioValue[0]** = 100,000, therefore no trading occurs on this day.

Calculating the Return On Investment (ROI):

Unless it is day 0 the new **ReturnOnInvestment[day]** can be calculated.

ReturnOnInvestment[day] = PortfolioValue[day] – PortfolioValue[day-1].

Whatever the value of the ROI it is stored in **ReturnOnInvestment[day]**. Using the **ReturnOnInvestment** array, the Sharpe Ratio is calculated and sent to the fitness function. When the Investment Simulator finishes execution a new chromosome is executed, and the process is repeated for this chromosome.

4.5.2 Sharpe Ratio Code

As stated before Sharpe Ratio = (mean – RFR)/Standard Deviation:

The pseudocode below is analogous to the code used in the GPWorld class to calculate the Sharpe Ratio for a chromosome.

Access ReturnOnInvestment Array
Calculate sum of all ReturnOnInvestments by traversing through array:
int size = 21;
While (int x=1; x<size; x++){
Sum = Sum + ReturnOnInvestment[x];
}
Calculate mean: mean = Sum/(size-1);
Calculate Standard Deviation of ReturnOnInvestment array
Risk Free Rate = 0.04/365
Return Sharpe Ratio

It must be mentioned that ReturnOnInvestment[0] is not used in the sample because on day 0 the Return on investment is always 0. The Sharpe Ratio calculation is explained in greater detail in Section 3.5.4.

4.6 Reuters Spreadsheet

I am receiving all my technical and fundamental data for the FTSE 100 stocks from Reuters 3000Xtra. Reuters 3000Xtra is a fast provider of a wide range of information for financial professionals. It delivers an efficient combination of real-time data with powerful analysis tools. The analysis tool that I have used is PowerPlusPro. This allows me to import Reuters real-time and historical data for stocks into a Microsoft Excel spreadsheet.

I have produced a spreadsheet containing a sheet for each FTSE 100 stock in the sample. I have also produced one sheet that contains all the FTSE 100 stocks in the sample displaying various factor models and calculations for each stock. The latter sheet is known as the "FTSE100Companies" sheet.

REUTERS	DSPLY_NAME	ASK	BID			
/.FTSE	FTSE 100 INDEX	0.00	0.00		Company Sector	Previous Day Price
/.AD.FTSE	FTSE 100 Index	#N/A unkno	#N/A unkno	Fully retrie	SECTOR_INDEX_NAME	PRICE_CLOSE
/AAL.L	ANGLO AMERICAN	1827.00	1826.00	AAL.L	FTSE All Share Mining Index	1979
/BT.L	BT GROUP	211.50	211.25	BT.L	FTSE All Share Fixed Line Telecommuniations Index	222.75
/HNS.L	HANSON	624.50	624.00	HNS.L	FTSE All Share Construction & Materials Index	639
/OML.L	OLD MUTUAL	157.25	157.00	OML.L	FTSE All Share Life Insurance Index	164.75
/SDRt.L	SCHRODERS NV	854.50	853.00	SDRt.L	FTSE All Share General Financial Index	895
/ABF.L	ASSOC.BR.FOODS	823.50	823.00	ABF.L	FTSE All Share Food Producers Index	839
/CBRY.L	CADBURY SCHWEPF	561.50	560.50	CBRY.L	FTSE All Share Food Producers Index	549.5
/HSBA.L	HSBC HOLDINGS	931.50	931.00	HSBA.L	FTSE All Share Banks Index	933
/OOM.L	O2	196.25	195.75	OOM.L	FTSE All Share Fixed Line Telecommuniations Index	197.75
/SGE.L	SAGE GROUP	242.50	242.25	SGE.L	FTSE All Share Software & Computer Services Index	258
/AL.L	ALLIANCE & LEICS	921.50	921.00	AL.L	FTSE All Share Banks Index	994
/CCL.L	CARNIVAL	3293.00	3289.00	CCL.L	FTSE All Share Travel & Leisure Index	3300
/ICI.L	ICI	330.25	330.00	ICI.L	FTSE All Share Chemicals Index	332
/PRTY.L	PARTYGAMING	118.25	118.00	PRTY.L	FTSE All Share Travel & Leisure Index	134
/SHP.L	SHIRE	724.50	724.00	SHP.L	FTSE All Share Pharmaceuticals & Biotechnology Index	744
/ANTO.L	ANTOFAGASTA	1727.00	1726.00	ANTO.L	FTSE All Share Mining Index	1869
/CNA.L	CENTRICA	232.50	232.25	CNA.L	FTSE All Share Gas Water & Multiutilities Index	254.75
/IHG.L	INTERCONT HOTEL	799.50	799.00	IHG.L	FTSE All Share Travel & Leisure Index	839.5
/PRU.L	PRUDENTIAL	528.00	527.50	PRU.L	FTSE All Share Life Insurance Index	550
/SMIN.L	SMITHS GROUP	990.50	989.00	SMIN.L	FTSE All Share Aerospace & Defense Index	1046
/AUN.L	ALLIANCE UNICHEM	748.00	746.50	AUN.L	FTSE All Share Pharmaceuticals & Biotechnology Index	800.5
/CNE.L	CAIRN ENERGY	1889.00	1887.00	CNE.L	FTSE All Share Oil & Gas Index	1920
/III.L	3i GROUP	867.00	866.00	III.L	FTSE All Share Equity Investment Instruments Index	847.5
/PSON.L	PEARSON	677.50	677.00	PSON.L	FTSE All Share Media Index	687.5
/SN.L	SMITH&NEPHEW	542.00	541.00	SN.L	FTSE All Share Health Care Equipment & Services Index	535.5
/AV.L	AVIVA	702.00	701.50	AV.L	FTSE All Share Life Insurance Index	705
/CPG.L	COMPASS GROUP	219.25	218.75	CPG.L	FTSE All Share Support Services Index	220.5
/IMT.L	IMPERIAL TOBACCO	1752.00	1751.00	IMT.L	FTSE All Share Tobacco Index	1737
/RB.L	RECKITT BENCKSR	1823.00	1820.00	RB.L	FTSE All Share Personal Goods Index	1920

Figure 4.6.1.1 displays a part of the "FTSE100Companies" sheet.

Book Value Per Share	Price/Earnings	52 Week High Price	52 Week Low Price	52 Week High PE	52 Week Low PE
BVPS	PE	HIGH_52W	LOW_52W	PE_HIGH_52W	PE_LOW_52W
16.73387645	16.23533905	1979	1130	16.23533905	9.589970443
0.458585035	9.9	235	196.5	12.87650602	9.044444444
3.772893446	18.46820809	640	451.5	25.68181818	15.23121387
0.793890917	6.079335793	165.25	115	93.5	4.769372694
15.33104043		899	630		
4.668587254	19.88151659	863	728	19.97630332	15.82608696
1.372021025	22.5204918	594	466.5	29.6	21.68032787
7.643646727	13.39477	949.5	825	15.46255644	12.52619167
1.172161581	41.19791667	209.25	114	59.78571429	26.51162791
0.569704634	23.07692308	259.5	192.25	23.21109123	18.84328358
3.956724274	11.25707814	994.5	814.5	11.26274066	8.538622129
74.15019443	20.23217082	3319	2637	24.16394805	17.00164672
0.605126199	11.36986301	336	231	22.56880734	9.87544484
-0.1012					
4.52615854	77.93598097	753	539	78.87875493	66.04655377
6.688598894	9.024420404	1869	1052	14.00913749	6.401927234
0.648920267	5.379089171	264.75	217.5	21.694714	4.835373583
4.56554208	34.28964785	842.5	634.979988	34.41218382	13.90906722
1.780591951	21.73825609	551.5	445	44.99539069	18.29965922
2.134267835	19.26335175	1052	813	26.4010989	16.32596685
2.961126783	13.99475524	875	720	19.60297767	12.89335664
2.699207647	157.635468	2018	1042	269.1983122	43.74475231
6.605131492	9.41670584	869	635.477122	35.68001751	8.738925243
3.234296302	9.978229318	694.5	608	60.22522523	8.933236575
0.773998138	32.45454545	558	451.5	41.63554892	28.24242424

Figure 4.6.1.2 displays another part of the "FTSE100Companies" sheet.

REUTERS	Reuters Group PLC
Announcement Date	Adjusted Dividend Value
26/07/2005	4.277778
16/02/2005	6.833333
27/07/2004	4.277778
17/02/2004	6.833333
22/07/2003	4.277778
18/02/2003	6.833333
23/07/2002	4.277778
12/02/2002	6.833333
24/07/2001	4.277778
13/02/2001	13.722222
25/07/2000	4.055556
08/02/2000	12.222222
20/07/1999	4.055556
09/02/1999	12.222222
22/07/1998	4.25
10/02/1998	12.375
23/07/1997	3.875
11/02/1997	11.25
24/07/1996	3.4375

REUTERS	Reuters Group PLC
Trade Date	Volume
03/01/2006	13344446
02/01/2006	#N/A ND
30/12/2005	3173426
29/12/2005	5519404
28/12/2005	7544635
27/12/2005	#N/A ND
26/12/2005	#N/A ND
23/12/2005	1248420
22/12/2005	5642407
21/12/2005	19087904
20/12/2005	6430825
19/12/2005	12463382
16/12/2005	16824168
15/12/2005	11666791
14/12/2005	13897488
13/12/2005	7476362
12/12/2005	6775346
09/12/2005	6298889
08/12/2005	7467301
07/12/2005	5323884
06/12/2005	6361584
05/12/2005	4402270

Figure 4.6.1.3 displays part of spreadsheet for a FTSE 100 stock.

MACD - RGO Simulation						Short Averaging Period		
Equity RIC:						n:	12	1-399
RTR.L		Fully retrieved at 00:02:10				Long Averaging Period		
EQUITY_PRICE_HISTORY		Fully retrieved at 23:08:27				n:	26	2-400
CONSOLIDATION:Daily;TRADE_DATE:T-20Y<T						Signal Averaging Period		
TRADE_DATE	CLOSE					n:	9	1-400
	Reuters Group PLC							
Trade Date	Close	SMA	EMA_1	EMA_2	SMA (MACD)	MACD	Signal	
3-Jan-06	434	418.9792	421.7718	411.8216	7.785429	9.950155	8.352055	
30-Dec-05	430.5	416.75	419.5485	410.0473	7.370745	9.501111	7.952531	
29-Dec-05	431	414.9792	417.5573	408.4111	7.02556	9.146133	7.565386	
28-Dec-05	431	413.1667	415.1131	406.604	6.782586	8.509109	7.170199	
23-Dec-05	422.25	410.7083	412.2246	404.6523	6.639666	7.572266	6.835471	
22-Dec-05	422.5	409.4583	410.4018	403.2445	6.629358	7.15728	6.651272	
21-Dec-05	420.5	408.3958	408.2021	401.7041	6.666845	6.498047	6.52477	
20-Dec-05	408.5	407.0208	405.9662	400.2004	6.767619	5.765744	6.531451	
19-Dec-05	407.75	406.8542	405.5055	399.5365	7.012755	5.969014	6.722878	
16-Dec-05	408.25	406.3958	405.0974	398.8794	7.237774	6.218	6.911344	
15-Dec-05	404	405.7917	404.5242	398.1297	7.397078	6.394445	7.08468	
14-Dec-05	407.5	405.8542	404.6195	397.6601	7.540895	6.959371	7.257238	
13-Dec-05	407.25	405.0417	404.0957	396.8729	7.586488	7.22283	7.331705	
12-Dec-05	409.25	404.125	403.5222	396.0427	7.593728	7.479495	7.358924	
9-Dec-05	409.25	402.8542	402.4808	394.9862	7.56322	7.494663	7.328781	
8-Dec-05	401.5	401.6458	401.2501	393.845	7.465181	7.405011	7.287311	
7-Dec-05	407.25	401.0208	401.2046	393.2327	7.361563	7.971963	7.257886	
6-Dec-05	409.75	400.2083	400.1055	392.1113	7.181625	7.99419	7.079367	
5-Dec-05	404	398.8542	398.3519	390.7002	6.997399	7.651735	6.850661	
2-Dec-05	406.5	397.6042	397.325	389.6362	6.830176	7.688795	6.650392	

RTR.L
EARNINGS_HISTORY
PERIOD_END_DATE:T-10Y<T
EPS_ANNOUNCE_DATE EPS_ADJ

REUTERS	Reuters Group PLC
EPS Announcement Date	Adjusted EPS
26/07/2005	9
16/02/2005	25.2
27/07/2004	20.4
17/02/2004	3.1
22/07/2003	0.3
18/02/2003	-26.3
23/07/2002	-5.2
12/02/2002	3.3
24/07/2001	19.1
13/02/2001	37.9
25/07/2000	25.2
08/02/2000	30.2
20/07/1999	14.3
09/02/1999	26.7
22/07/1998	13.3
10/02/1998	24
23/07/1997	14.2
11/02/1997	30.4
24/07/1996	14.6

Figure 4.6.1.4 displays another part of a spreadsheet for a FTSE 100 stock.

Figures 4.6.1.3 and 4.6.1.4 display a part of a sheet for a single FTSE 100 company. 80 sheets were created, as data was required for each company that is going to be used in the sample. Each individual sheet for each company simply imports a lot of data from the "FTSE100Companies" sheet. However, all the historical data for each company is imported from Reuters databases, because the "FTSE100Companies" sheet contains only some historical data. In order to read data into my GP system I used a CSV parser. Therefore, each company sheet was required to be in CSV format. As I decided on my factors in the design stage; a CSV sheet was created for each of the 80 stocks in my sample. Data was contained for 9 factors from between 02-January-2004 and 10-February-2006. A part of one of the sheets is displayed in figure 4.6.1.5. Each column represents a factor in order to simplify the way data was read into the Investment Simulator.

Date	Close	MACD	EPS	Dividend	Volume	One Month	One Year	ROE	BVPS
10-Feb-06	642	5.919394	29.1	10.22222	48697600	0.037157	0.081719	18.76328	2.698637
09-Feb-06	637.5	4.173649	29.1	10.22222	35214904	0.015127	0.073232	18.76328	2.698637
08-Feb-06	621	2.342917	29.1	10.22222	30116498	-0.01036	0.039331	18.76328	2.698637
07-Feb-06	620.5	1.652159	29.1	10.22222	27084332	-0.00161	0.045493	18.76328	2.698637
06-Feb-06	619	0.797378	29.1	10.22222	45383164	-0.01276	0.036851	18.76328	2.698637
03-Feb-06	614	-0.1525	29.1	10.22222	40708288	-0.00567	0.030201	18.76328	2.698637
02-Feb-06	611	-0.85359	29.1	10.22222	58021964	0	0.02862	18.76328	2.698637
01-Feb-06	614.5	-1.43056	29.1	10.22222	63828004	0.000814	0.028452	18.76328	2.698637
31-Jan-06	601	-2.50254	29.1	10.22222	42592196	-0.02117	0.032646	18.76328	2.698637
30-Jan-06	601	-2.45983	29.1	10.22222	27824740	-0.01958	0.047038	18.76328	2.698637
27-Jan-06	612.5	-2.36953	29.1	10.22222	42786504	-0.00082	0.056946	18.76328	2.698637
26-Jan-06	601	-3.39073	29.1	10.22222	52777704	-0.01556	0.047038	18.76328	2.698637
25-Jan-06	587.5	-3.47392	29.1	10.22222	54245832	-0.02651	0.005993	18.76328	2.698637
24-Jan-06	587.5	-2.15502	29.1	10.22222	60845204	-0.02165	0.013805	18.76328	2.698637
23-Jan-06	596	-0.44028	29.1	10.22222	30609184	-0.00667	0.018803	18.76328	2.698637
20-Jan-06	597.5	0.8893	29.1	10.22222	30429450	0.001676	0.018755	18.76328	2.698637
19-Jan-06	605	2.420665	29.1	10.22222	42019424	0.007494	0.026293	18.76328	2.698637
18-Jan-06	601.5	3.576397	29.1	10.22222	53202024	0.009228	0.022959	18.76328	2.698637
17-Jan-06	609	5.355875	29.1	10.22222	48098468	0.020956	0.029586	18.76328	2.698637
16-Jan-06	620.5	6.785595	29.1	10.22222	16519211	0.036759	0.043734	18.76328	2.698637
13-Jan-06	621	7.34083	29.1	10.22222	23793780	0.034138	0.052542	18.76328	2.698637
12-Jan-06	628.5	7.891827	29.1	10.22222	26583016	0.042289	0.064352	18.76328	2.698637
11-Jan-06	626.5	7.716673	29.1	10.22222	21458756	0.038111	0.052941	18.76328	2.698637
10-Jan-06	619	7.584506	29.1	10.22222	35432064	0.03772	0.025684	18.76328	2.698637
09-Jan-06	628	8.069129	29.1	10.22222	55198392	0.04754	0.051926	18.76328	2.698637
06-Jan-06	627.5	7.663952	29.1	10.22222	32667716	0.047679	0.068086	18.76328	2.698637

Figure 4.6.1.5 Part of a CSV file for a stock

4.6.1 Spreadsheet Calculations

To extract data from Reuters into PowerPlusPro was not a simple process. I was required to code many different formulas and Reuters' functions in order to obtain the data I required. The first thing I needed for each stock in my sample was to retrieve as much historical price information as possible. In order to do this I used a function known as **DeHistory**. The purpose of this function is that it retrieves stock historical data from a specified table in the Securities 3000 database (A database in Reuters). It functions as follows:

DeHistory(*Code, Table Name, FieldList, Destination Cell, Conditions, DeMode*)

The arguments are defined as follows:
Code: This is the RIC Code of the equity used, or the cell that contains this value. The RIC code is the Reuters Instrument Code. All equities have a RIC code. For The BT Group this is BT.L.
TableName: Name of the table, data is retrieved from.
FieldList: This is the array of fields to retrieve for this stock. For example closing price.
Destination Cell: Cell reference specifying the top-left corner of the destination cell, where data will be imported.
Conditions: Any kind of extra information required to retrieve data, depending on the provider and the kind of request made.
DeMode: This is used to specify the data source, to format the results and to specify whether data is refreshed automatically.

For example, the code entered for the equity BT Group was (RIC Code "BT.L"):

DeHistory(*BT.L, EQUITY_PRICE_HISTORY, TRADE_DATE:CLOSE, B8, CONSOLIDATION:Daily;TRADE_DATE:T-20Y<T, "LAY:HOR IDR:SKIP SOURCE:EQUITY RET:A5000")*

EQUITY_PRICE_ HISTORY: This is the name of the table that the historical price data was retrieved from. The choice of table is purely on account of the type of data required.
TRADE_DATE: This is the date of the closing price.

CLOSE: This is the closing price for the equity.

CONSOLIDATION:Daily;TRADE_DATE:T-20Y<T: This ensures that the daily closing price is retrieved for up-to the last 20 years of available data for that equity.

"LAY:HOR": This denotes the layout parameter for the array orientation. In this case it is a horizontal orientation.

"IDR:SKIP": This removes rows containing invalid data. For example "N/A".

"SOURCE:EQUITY": This retrieves equity data from the Reuters Securities 3000 database. Each database has many tables.

"RET:A5000": This returns an array size of up to 5000 rows.

I wanted the closing price to continue being retrieved during my project. Therefore this required me to be continuously updating the data in Reuters. In order to do this I was required to make use of a function known as **DeUpdate**. This function works in exactly the same way as **DeHistory**, except that *DeMode* is used in a completely different way. *DeMode* is used to update the current values in the next data engine session. Therefore, **DeUpdate** is used to update the **DeHistory** values.

Once I had retrieved all the historical closing prices for the equities in my sample I was able to use these values to calculate historical **MACD** values.

MACD(d) = EMAP12(d) – EMAP26(d)

EMAP12 = Exponential Moving Average for a 12-day Period

EMAP26 = Exponential Moving Average for a 26-day Period

The value d = today's date.

Close(d) = today's closing price for this particular company.

EMAP12(d) = ((2/13)*Close(d))+((11/13)*EMAP12(d-1))

EMAP26(d) = ((2/27)*Close(d))+((25/27)*EMAP26(d-1))

EMAP12(d-1): This denotes the previous day's EMAP12 value. In order to retrieve the first day's value (first day value in the whole sample, no EMAP12(d-1) value on this day) in the sheet, I used today's closing price. Therefore, for the first day in the sample:

EMAP12(d) = ((2/13)*Close(d))+((11/13)*Close(d))

This was also the case for EMAP26(d) for the first day's value in the sheet, where Close(d) was used twice.

The rest of my historical data was calculated in exactly the same way as the closing price was calculated. The data was calculated using **DeHistory** and **DeUpdate**. The following historical data was obtained:

- ***EPS:***
  1. Table Name: EARNINGS_HISTORY
  2. Time Period: Data is retrieved for the previous 10 years
  3. Frequency of Data: Quarterly data is retrieved, therefore, 4 EPS values are retrieved for each equity per year
- ***Dividend Value***
  1. Table Name: DIVIDEND_HISTORY
  2. Time Period: Data is retrieved for the previous 10 years
  3. Frequency of Data: Quarterly data is retrieved, therefore, 4 dividend values are retrieved for each equity per year.
- ***Volume of Trades***
  1. Table Name: INDEX_PRICE_HISTORY

2. Time Period: Data is retrieved for the previous 10 years
3. Frequency of Data: Daily data is retrieved

- ***Shares Outstanding***
  1. Table Name: SHARES_OUTSTANDING
  2. Time Period: Data is retrieved for the previous 10 years
  3. Frequency of Data: Variable, purely dependent on the equity. In general 20 values are retrieved annually
- ***Revenue, Long Term Debt, Net Attributable, Shareholder's Equity, Equity Issued***
  1. Table Name: COMPANY_REPORT
  2. Time Period: Data is retrieved for the previous 10 years
  3. Frequency of Data: One value is retrieved annually

One fact to point out from the above information is that each piece of data has a different frequency. For example ***Volume of Trades*** has daily values whereas ***EPS*** has quarterly values. In order to accommodate for this when producing my CSV files the most recent value is used. Therefore:

Today's Date = 23/02/06

Last EPS Announced Value(12/01/06) = 12.9

Today's EPS Value = 12.9

This process was incorporated for many factors, because some factors such as ***EPS*** do not change daily. They change when there is a new announced value. Not all the factors described on the previous page have been used in my CSV files. This is because many of them are used simply for calculations. ***Shareholder's Equity, Equity Issued*** and ***Net***

Attributable are used to calculate ***Book Value per Share*** and ***Return On Equity***. ***Long Term Debt, Revenue*** and ***Shares Outstanding*** were not used. This is simply because of the infrequency of this data. I believed they would not provide an accurate enough measure of a stock's performance due to their infrequency.

Book Value per Share = (Shareholder's Equity/Equity Issued)

Return On Equity = 100*(Net Attributable/Shareholder's Equity)

I also employed formulas to calculate the **One Month Price Momentum** and the **One Year Price Momentum.**

One Month Price Momentum = ((Today's Closing Price-Closing Price a month ago)/Closing Price a month ago)

One Year Price Momentum = ((Today's Closing Price-Closing Price a year ago)/Closing Price a year ago)

There were various other factor models I attempted to generate. However, I was unable to gain access to the relevant historical data on PowerPlusPro. The factors included:

- ***Relative Strength Index***
- ***2 Years Consensus Earnings***
- ***Rate of Reinvestment***

4.7 System Testing

The testing used assumes the form of verification and validation. The verification is used to test that the system is correct and the validation is used to test that everything functions in the way it has been programmed to do so.

4.7.1 Verification

Verification firstly checks that the program runs without any errors and all the classes compile correctly. However, other tests are required in order to ensure that the system does exactly what it is supposed to do. First I am going to perform robustness tests to check that out of range values are not accepted.

4.7.2 Robustness Tests

Two robustness tests were carried out to ensure that the system only allows certain values within a range. Any other values must not be accepted. The table below provides representation of the values that are accepted and the out-of range values that were tested on the system.

Test	**Values Accepted**	**Out of Range Test Values**
1. Crossover Probability	0.0>= x =<1.0	x = 1.5
2. Reproduction Probability	0.0>= x =<1.0	x = 1.5

4.7.3 Test Schedule

The schedule below is there to test the main parts of the system:

1. **Test Investment Simulator**: This test is there to test the general functionality of the Investment Simulator. It is there to ensure that the simulator can take in a set of values, and return a Sharpe Ratio.
2. **Test Chromosome**: This test is there to ensure that a chromosome can be generated and that all the values computed from this chromosome can be stored in an array. This is to ensure that for the 21 days of trading and for the 80 stocks, a chromosome's values are generated and then stored in a two dimensional array.
3. **Test getPrice() Method**: This is to test that for each stock and the 21 days of trading the prices that are generated can be stored in a two-dimensional array.
4. **Check the fitness:** This test is to ensure that the fittest chromosome is always returned.
5. **Check GP Chromosome:** Check that a GP individual cannot be initialised to a depth of greater than 6. Also its crossover depth cannot be greater than 17.

All tests were completed successfully and the results are shown in the appendix.

5. Experiment

This experiment outlines how the two experiments were set-up and the results of these two experiments.

5.1 Experiment Design

The main purpose of the experiment, as detailed in earlier chapters, is to generate a financial formula that can be used to make decisions on which stocks in the FTSE 100 to buy long or sell short. There will be two experiments. The first will be used to generate the fittest chromosome. The second experiment will be run to test the performance of this chromosome on separate data. The design of the first experiment (Training Period) will be as follows:

- The Genetic Program will run for 50 generations
- There will be 50 chromosomes in the population
- 80 equities from the FTSE 100 stock index will be used
- The data is from 02/01/04 until 30/01/04; this constitutes 21 days of trading

The design of the second experiment will be as follows:

- Only the fittest chromosome from the previous experiment will be used
- This chromosome will be tested on the same 80 equities from the FTSE 100 stock index as in the first experiment
- A different time period will be used however. The data is from 02/02/04 until 10/02/06
- There will be no running of the genetic program, only the Investment Simulator. This experiment is simply there to measure the performance of the chromosome on different data. It is not generating new chromosomes.

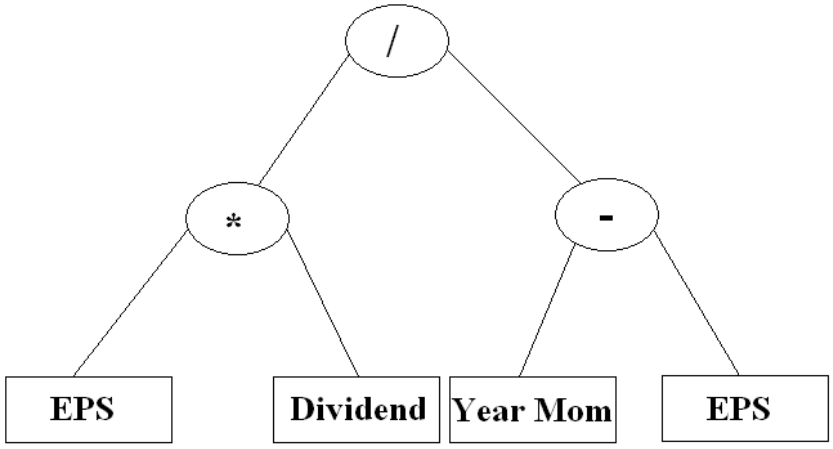
5.2 Training Period Results

The genetic program was run for 50 generations, and at the end of this training period the fittest chromosome was displayed along with the fittest chromosomes for each generation. The Sharpe Ratio, which constitutes the fitness of a chromosome was

displayed. The fittest chromosome was found in the 32nd generation. This chromosome was:

$$\frac{EPS \times Dividend}{Year\,\mathrm{Pr}\,iceMomentum - EPS}$$

The tree-structure of the chromosome is as follows:



The Sharpe Ratio of this chromosome was 0.8348981.

5.3 Genetic Program Results

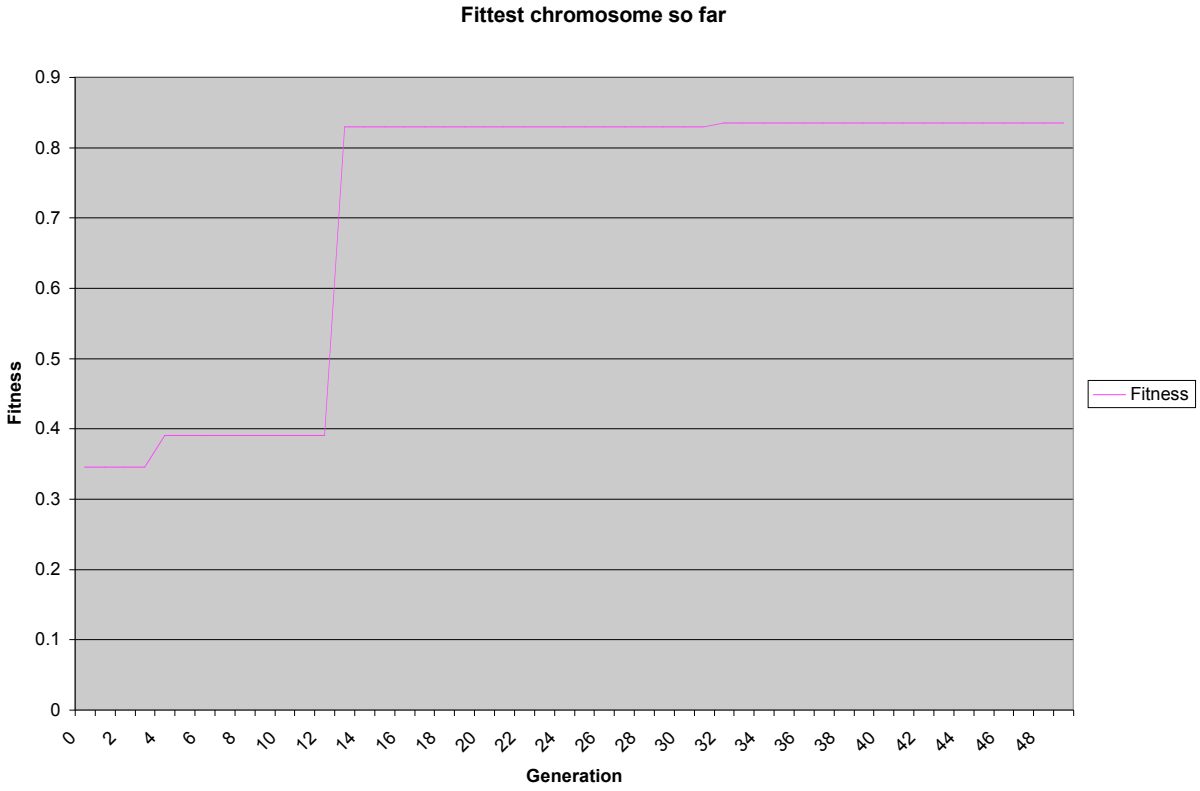


Figure 5.3 “Fittest chromosome so far” line graph representation

Figure 5.3 depicts the fitness of the best chromosome obtained so far in the run, at each generation. The highest point in the diagram is reached at generation 32, where the fittest chromosome was acquired. In general, as the number of generations increases, the fitness of chromosomes should increase. In the experiment certain generations achieve relatively high fitness, however, there is no evidence that the fitness of chromosomes is necessarily increasing every generation, the way a genetic program should. After generation 32 there is no increase in fitness. The first thing to remember is that a very fit chromosome is achieved, and that the main objective of this experiment is to acquire this chromosome. Whether this chromosome is obtained in generation 0 or generation 49 is not really important.

Figure 5.3 does display the fact that crossover assists the search for fitter chromosomes. This is because generation 0 contains the best chromosome of the original population.

Any changes to this population are because of the Crossover operation. Generation 0's best fitness is 0.345, and the fittest chromosome is 0.8348981. In genetic programming, a strong reason for the fitness always improving as generations increase is elitism. Genetic programs usually contain elitism. This means that the fittest chromosome from a generation is always put back into the next generation. However, the implementation that was used ("jgprog") did not provide an implementation of this. Fitter chromosomes were a lot more likely to be selected (in selection) and 90% of these were used in crossover, this meant that the same chromosome usually did not appear in the next generation (crossover changes the chromosome, as explained in Section 2.2.4). Even though crossover is used as a way of evolving a fitter chromosome, this does not necessarily mean that a fitter chromosome is always evolved. Low fitness children can be produced as well as high fitness children.

Overall, a very fit chromosome was achieved, and regardless of which generation it was acquired in, it will be tested on the out-of-sample data.

5.4 Portfolio Performance using Fittest Chromosome

The fittest chromosome achieved; that was tested with out-of-sample data was:

((EPS * Dividend)/(Year Price Momentum – EPS))

The data used was between the time period of 02/02/04 until 10/02/06. The portfolio value at the beginning of the simulation was *£100,000*. After completion the portfolio value was **£138,036.22**. This gives an annual return of 19% per annum. Over the same period the FTSE 100 stock index increased at a rate of 14.5% per annum. Therefore, this chromosome provided a greater rate of return than the FTSE 100 stock index did. In some ways, this does suggest that this chromosome has the capabilities of using various instructions and providing a profit on investment. The more important result of this experiment was the Sharpe Ratio and this was 0.110914335. This value is very different from the Sharpe Ratio of this chromosome during the Training Period, which was 0.8348981. One reason for this is that the Training Period was very short and this may not have been enough time to sustain a chromosome that could work under any trading conditions.

6. Summary & Conclusion

This section will allow me to summarise what was achieved, deduce a conclusion and provide a brief critical evaluation of the project. Ideas of how the project could be improved in the future are also provided.

6.1 Summary of the project

All of the project objectives were achieved:

1. PowerPlusPro sheets were programmed in order to capture stock data.
2. An investment simulator was designed and implemented that used a non-linear multi-factor equation (chromosome) to determine buy and sell decisions for a long/short equities hedge fund (based on FTSE 100 stocks).
3. An existing Genetic Program System was modified and extended, which called the Investment Simulator to determine the fitness of each chromosome.
4. Experiments were run to determine the efficacy of the Genetic Program and Investment Simulator system. The experiments were operated to monitor:
    i) The behaviour of the system during the Training Period.
    ii) The behaviour during the out-of-sample test.
5. A brief critical evaluation of the system and experimental results is provided in Section 6.2.

6.2 Critical Evaluation and Conclusion

The first thing to conclude is that my results certainly are promising and provide encouraging signs for future work. The Training Period was a very short time period and to achieve such high Sharpe Ratios was a success. The out-of-sample result was also encouraging to an extent. The chromosome that was acquired in the Training Period and subsequently tested on two years of data did return a profit. Therefore beginning with the belief that factor models can predict stock price movements, then this study certainly does combine enough real-life trading conditions to advocate future work.

However, even though, the Sharpe Ratio for the fittest chromosome was positive in both experiments, it was contrastingly very different. This firstly suggests that a longer time period was required for the Training Period. This could also imply that GP-evolved trading rules are ineffective when used with historical equity data. But the return of 19% per annum was good, so the problem is more likely to be because the Training Period length was too short. There is also a possibility that the factors used for this study were not effective, but others may have provided more favourable results.

As stated in the Background section, the fitness of chromosomes should improve as the generations increase. Therefore, the fittest chromosome should be achieved in the final or final few generations. The Training Period experiment ran for 50 generations and the fittest chromosome was only achieved in the 32^{nd} generation. The main reason for this occurring is because of the absence of elitism from the GP System, where the fittest chromosome from the previous generation is replaced into the next generation. However, a very important thing to point out is that a very fit chromosome was achieved, and that the main objective of this experiment was to acquire this chromosome. Whether this chromosome was obtained in generation 0 or generation 49 is not really important.

The results provide a significant amount of evidence that the crossover operation successfully provided a form of exploration. This is because generation 0 contains the fittest chromosome of the original population. Any changes to the population after generation 0 is because of the Crossover operation. The fittest chromosome in generation 0 has a fitness of 0.345, and the fittest chromosome obtained during the Training Period is 0.8348981.

Overall, the objectives of this project were all achieved (see Sections 1.1 and 6.1). A formula was obtained that when tested on two years of sample data provided a greater rate of return than the FTSE 100 stock index did over the same time period. I believe this project has been a success and certainly provides encouraging signs for future work.

6.3 Future Work

The following are suggestions for future work:

- A longer time period for the Training Period would certainly provide more accurate chromosomes that consider a greater amount of information and data.
- In this study equities were bought long and sold short similar to a hedge fund, however, there was no diversification. Diversification is where companies from different sectors are purchased in order to reduce risk. The idea behind this is that, if one sector performs very badly then this can be covered by other sectors that may have performed in a more profitable way. Therefore, in future studies I would propose the use of diversification.
- Another area that must be considered is how stocks are chosen to be bought long or sold short. A different algorithm could be used to select stocks as opposed to the one that has been employed in this study. Also an equal amount is invested on each stock on a particular day. This could be changed to reflect the stocks' chromosome values, and be proportional to this value.
- Elitism should be employed in the genetic program to ensure that the fittest chromosome is always re-placed into the next generation.
- In real life conditions for trading, transaction costs are applied to every investment made. In order to make the results more accurate these costs certainly should be applied.
- Only 80 equities were used from the FTSE 100 stock index. The FTSE 100 stock index has around 100 equities. Using more equities may produce more accurate results. Also using equities from a broader range of stock index such as the FTSE 250 would allow for this.

References

1: Michael Beckett, How the Stock Market Works (Kogan Page), 2004

2: John J Murphy, Technical Analysis of the Financial Markets (NYIF), 1999

3: John Koza, Genetic Programming: Vol 3 (Morgan Kauffmann), 1998

4: Wei Yan and Christopher D. Clack, Automatic Generation of Non-linear Factor Models for Long/Short Hedge Funds: A Genetic Programming Approach, Unpublished report, UCL Department Computer Science, 2005

5: William F. Sharpe, Investments (Pearson), 1998