Hedge Fund Selection Using Genetic Algorithms

Nicky Cheung Ho Tsang
University College London
BSc Computer Science Project 2004/2005
Project Supervisor: Chris Clack
Submission Date: 29 April 2005

"Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand "

John H. Holland
(1975)

Abstract

Genetic Algorithm was proposed by Holland J.H [9] in 1975, a programming technique based on evolutionary computing. I am going to use this technique to optimize strategies used in hedge funds. Using historical equity price and indicators, returns in term of total portfolio value are obtained and it is used to assess the risk, hence used as the fitness function for the set of strategies. Using the system, a set of integers will be generated and these integers are use to control the predefined investment strategies. The system will improve the strategies in terms of its control integers until the best set of integers is found, hence the best set of strategies for the given data.

Acknowledgement

This project would not have been completed without the generous help and assistance from many people and I would like to give my profound thanks to all of them.

I would like to take this opportunity to thank my project supervisor Chris Clack for his guidance and his valuable advice, my parents for their continued support and encouragement. I would also like to thank Wei Yan for her advice in the genetic algorithms.

My gratitude is particularly directed to Michelle Leung, Oscar Tsui, Karen Wang, Jaymen Leung and Emily Chiu for their support, knowledge and friendship. Their considerable advices and influences have helped me to get through non-technical problems painlessly and made this long process a little easier for me.

I would especially like to thank Reuter Group for allowing me to access their valuable information in the stock market for my research. Without them the experiment part of the project will not be achieved.

Finally I would like to thank my girlfriend Jena for her understanding and long-lasting support throughout the long enduring project.

Table of Content

Chapter 1

1. Introduction

Genetic Algorithms (GA), a type of evolutionary computing, are search algorithms based on the ideas of natural selection and inheritance. The concept of GA is designed to mimic processes in natural systems necessary for evolution, specifically those that follow the principles of "survival of the fittest" suggested by Charles Darwin [5]. As such they represent an intelligent operation of a search within a predefined data space to solve problems.

GA was first developed by John Holland [9] in the 1970's. GA has been widely studied, experimented and applied in many fields in computing. GA provides substitute methods for finding solutions in complex problems and its performance is generally better than traditional methods. Many of the real world problems involved finding optimal parameters, which might be proven difficult for traditional methods but ideal for GA. It can also be optimized to give a better performance by customizing the algorithm to different problem sets. This project applies the GA technique to the optimization problem of hedge fund strategies selection. The outcome of this project will be a system which takes in sets of historical data for training and produce a set of strategies that can performs well in the stock market. The system will then be used to test various time period and markets to show the adaptability and robustness of GA in the stock market.

1.1. Motivation

1.1.1. Hedge Fund

Hedge Funds are a collective of rich individuals investing into a large portfolio containing a large number of investment instruments. The main objective of a Hedge Fund is to ensure a profitable return for its partners. To achieve that the fund managers make decisions based on common investment strategies such as *Short-selling*, *Arbitrage* and *Hedging* [4] involving a combination of instruments. These clever techniques are often used collectively rather than independently to reduce risk by sacrificing some profit. But investment decisions are made only based on the fund manager's experience, so there is a chance of mangers being too conservative or the manager's speculations were incorrect, these human-caused errors can severely damage the profits in the hedge fund. I believe using evolutionary computing can reduce the chance of these errors, because of its ability to optimize solutions.

1.1.2. GA Technology

Genetic Algorithms can be use in many applications to solve all kind of problems providing the users can encode the problems into a form that the GA can accept, usually

in a form of chromosomes[1]. Using the chromosomes, the GA compares the relative fitness and uses various GA operations such as recombination and selection to improve the solution after each generation. To ensure success of GA, an effective representation of chromosomes and meaningful fitness evaluation are essential. Generally, Genetic Algorithms are useful when:

- The search space is large, multidimensional, and complex or has a high level of uncertainty.
- Domain knowledge is limited or narrowing the search space is difficult
- No mathematical analysis is available.
- Traditional search methods fail or have a poor performance.

Even though standard GA is very efficient when optimizing a complex problem, it is heuristic and it will only give an approximate static solution and will not further evolve in a dynamic environment. A problem of using GA was proposed by J.E. Baker in 1985 [1]. In his study, he suggested that premature convergence can occur in the population of chromosomes. This will stop the evolutional process in the system before the solution has optimistically evolved. In the next chapter, I will further explain Genetic Algorithms and its usage.

1.2. Objective

Following the idea of evolutionary computing and properties of GA, I have decided to design and implement a hedge fund selection system based on the genetic algorithm. Given the historical assert price data and set of customized strategies, the system would be able to manage a portfolio by applying the set of customized strategies based on the chromosomes generated by the GA. The GA will improve the performance of the portfolio after each generation and eventually a set of integers that is very close to the best possible solution will be found.

1.2.1. Technical Objectives

There are some technical objectives that the system must achieve. To optimize the GA so that it can find the best possible solution (thought it can't guarantee to find the global optimum). The system must be able to find the best possible chromosome within an acceptable complexity and how I define the best chromosome will be explain in chapter 4. The system must be able to simulate the various types of trading in the stock market. The system must be upgradeable which means investing instruments and strategies can be added without changing a lot of code. The system must also be able to separate the investment simulator from the GA modules, so that the simulator can be run on the best chromosome found in training without using the GA.

[1] Chromosome is a technical word used in biology. It contains the genetic information of the species. In Genetic Algorithms, it represents the candidate solution of the encoded problem. Chromosomes will be further explained in section 2.1.3.

1.2.2. Experimental Objectives

After design and implementing the system, I am going to use the system to experiment with selected real world's asset price data from different time period and markets in different countries to examine the following research questions:

- *Can the trained GA gives highest possible portfolio return when it is used in other time periods? For example, training the GA with the assert price data in the 1995-1996, it should be appropriate to use it in the stock market in 2005.*
- *Can the trained GA gives highest possible portfolio return when it is used in different stock markets? For example, training the GA using price data from London exchange market, the trained GA should also do well in the Hong Kong exchange market.*
- *Can the trained GA adapt to big changes in asset price? For example, Wall Street crash, dotcom bubble burst and etc.*

1.3. Structure of Report

Chapter 1 gives an introduction to my project that includes an explanation on the motivation behind this project, my objectives and a brief description on GA.

Chapter 2 has two sections. The first section will give detailed explanations on evolutionary computation: a more detailed explanation of standard genetic algorithms and the analysis of techniques for implementing genetic algorithms. The second section will explain the financial model used in the project which includes: investment strategies and instruments, technical analysis and indicators.

Chapter 3 will outline the system architecture and the design process. It will also give details on the software development tools and the difficulties encountered when designing the system.

Chapter 4 will present the process of implementation and testing of the system. This chapter will explain the how functionalities are implemented and tested. I will also discuss the difficulties encountered during the implementation phase. At the end of the chapter, I will review the system with the technical objective presented in chapter 1.

Chapter 5 explains how experiments and benchmarking are carried out and how the asset price data was selected. In this chapter, the results and findings of the experiments will be presented. Using the results, the hypotheses illustrated in chapter 1 will also be examined here.

Chapter 6 will conclude the report by commenting and evaluating critically on the project and using the findings from the experiments, I will suggest improvements and ideas for further developments.

Chapter 2

2. Background and Research Information

This chapter gives the background information on Evolutionary computing and Hedge Fund in order to give the reader some understandings for this report. This chapter consists of two sections. The first section will give detail information on evolutionary computing and describe techniques such as genetic algorithms and genetic programming. The second section will give information about the stock market, hedge fund portfolio and strategies.

2.1. Evolutionary computing (EC) – *Genetic Algorithms*

Evolutionary Computation is a computational technique that coalesces inspiration from biology and techniques of computer science to produce artificial intelligence. EC often provides robust methods to solve complex problems. It mimics many processes in the natural world, which will be discussed later on in this section. The main techniques in evolutionary computing are genetic algorithms and genetic programming but for the purpose of my project I will concentrate on explaining genetic algorithms (GA).

2.1.1. A Brief History and Applications of GA

GA, a class of adaptive stochastic optimization algorithms, was first developed by John H. Holland and it was published in his book "*Adaptation in natural and artificial system"* in 1975 [9]. He created an electronic chromosome in a form of a binary string and he selects chromosomes for reproduction using its relative fitness based on the principles of genetics. This will allow the program to narrow the search space in order to find optimized solutions and using the optimized solutions as the input for the next generation. So the population of solutions improves after each generation. John H. Holland using GA has laid the first milestone for EC.

GA is used in many applications in today's world and in various industries, for example: scheduling rosters for flight attendants, the GA incorporates all attendants' constraints and objectives into a single optimized roster. Another example is using GA in the medical field, it is used to help developing treatment programs and optimize drug formulae. These applications use GA because these problems are very complex, have many possible solutions and mathematical analyses are very limited.

2.1.2. Standard GA

Standard GA can be separate into five phases: Initialization, Fitness function, Chromosome Selection, Recombination and Mutation. A standard GA has the phases arrange in the order as shown in the diagram:

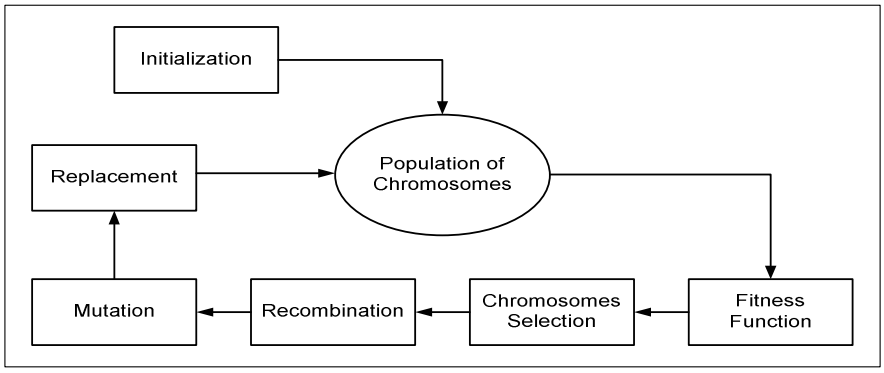


Figure 1: Standard GA Model

This standard model suggested by Melanie Mitchell [11] based on the theory of evolution has the basic function of solving simple problems. The genetic operators in this model have very basic functionalities and they mimic natural genetic operators almost directly. For example, the recombination operator has only one point of crossover, it splits the first parent's chromosome into two sections, and then it splits the second parent's chromosome at the same point. At the end, the combiner cross combines the sections to produce children chromosomes. The functionalities and customizations of each genetic operator will be described in the next section. This model has become very influential in the field of EC, many today's customized GA system which are capable of solving more complex problems are designed initially base on this standard GA model.

2.1.3. GA Operators and features

2.1.3.1. Chromosomes

Chromosomes represents the candidate solutions for the problem that the GA will perform fitness test on. Chromosomes initially proposed by John H Holland was a string of binary digits then different implementations arise as more researchers are involved with genetic algorithm research. Janikow and Michalewicz in 1991 [10] suggested different implementation such as using integers, floating points, character and even small sections of code[2]. Chromosomes can also be variable size, so the length of chromosome changes in length. The chromosome I am going to use in the system is a fixed size chromosome with integer in genes. Below shows a graphical view of the type of chromosomes I am going to use in my system. Technical terms that I will be using a lot in this report are also shown in the diagram:

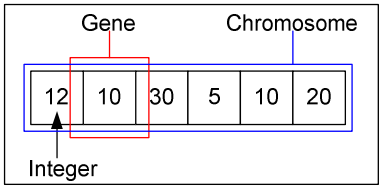


Figure 2: Chromosome

[2] Using code as chromosomes is mostly used in Genetic programming.

2.1.3.2. Initialization

The main function of this operator is to generate an initial population of chromosomes with predefined length. These generated chromosomes act as candidate solutions to the problem. These chromosomes are often randomly generated and distributed about the solution space because it can provide the GA with a diverse range of possible solutions. The size of population varies to different type of problems, because it can directly affect the time complexity of the system and in standard GA, there is a trade off between complexity of the system and the chance of the system to stop evolving before finding the best solution. This dilemma can be avoided by adding special functions or operators to the system, it will be discuss later on in this chapter.

2.1.3.3. Fitness Function (simulator) and Chromosome Representations

Fitness represents how well the chromosome solves the problem and the fitness function evaluates the chromosomes by simulating the chromosome's integers and output a fitness value. Simulation such as optimizing schedules can be very complex and this can create a bottle neck in the complexity of the system. The evaluation of fitness is very closely related to the representation of chromosomes. Chromosomes representation is a very important issue in GA, it can directly influent the performance and the correctness of the system. When designing the representation of chromosome, designers must understand the problem domain in order to encode the candidate solutions into a common form of string representation (chromosomes). The simulator is used to decode these chromosomes and extract relevant integers to determine the fitness based on user-inputted training data and objective assessments.

2.1.3.4. Chromosomes Selection

This operator act a survival selector in the natural world, either part of the chromosome survives or the whole chromosome to the next generation. It selects chromosomes with good fitness value in the population to become parent chromosomes for the next generation. This operator may also select the best chromosomes in the population to be placed in the next generation (See *Elitism* section in this chapter). There are many selecting mechanisms in EC, one of which is the roulette wheel sampling suggested by Goldberg [7] and I have applied this mechanism in my GA system because of its efficiency and it reduces the possibility of losing excellent genes embedded in a poor performing chromosome. I am going to explain it using a diagram:

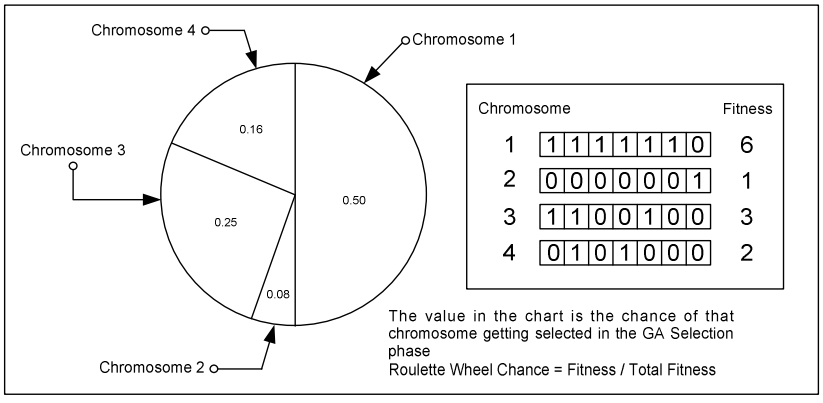


Figure 3: Roulette Wheel Selection

The above figure shows how the chance is derived from the fitness of chromosomes, roulette wheel chance is calculated by dividing the fitness of each individual chromosome by the sum of all chromosomes' fitness in the population. This selection method is fitness proportionate, a higher fitness chromosomes will have a higher chance of getting selected. The advantage of this mechanism is that the chromosomes with low fitness will not get eliminated completely, it will only get a lower chance of getting selected. So it is not guaranteed that the fittest chromosome goes to the next generation. But the disadvantage of roulette wheel selection is when the best chromosome is dominating the population (e.g. 95%) then the worse chromosome will have a tiny chance of being selected, hence accelerate the problem of premature convergence of population. There are other selection mechanisms proposed by T. Bäck and F. Hoffmeister in 1991 [3].

2.1.3.5. Recombination (Crossover)

The process of recombination of chromosome in GA is very much the same in the natural world. It has a point of crossover which can be randomly picked or predefined and the parent chromosomes are split into two segments at that point. A segment is taken out from each parent chromosome and then recombining the segments to form children chromosomes as shown in the diagram below:

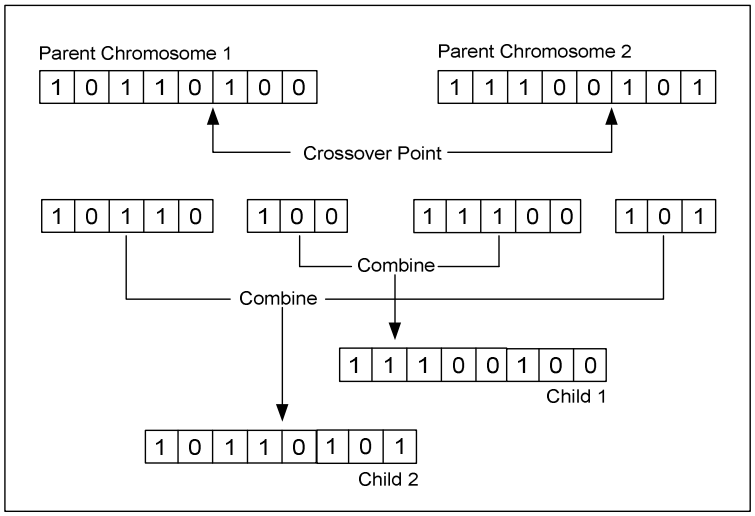


Figure 4: Single Point Crossover

There are many variations in crossover mechanism such as multi point crossover, some mechanisms are more suitable than other in different problem, offering different performance and rate of evolution.

2.1.3.6. Mutation

This operator randomly with a very small probability replaces some of the integers in chromosomes. For example, the chromosome (12,14,15,12,15,20,23) might be mutated to the chromosome (12,14,18,12,15,20,23). The reason for this operation is to ensure that if the initial population does not contain the best solution, some random candidate solution will be imported to prevent premature convergence of the population of chromosomes.

2.1.3.7. Elitism

This idea was introduced by Kenneth De Jong [6] in 1975. Elitism is an additional chromosomes selecting method that enhances the performance of GA. It copies the best chromosomes from the current population to the new population in the next generation and the rest of the populations are chosen in the conventional way. Elitism can increase the performance of GA, because it prevents losing the best chromosomes during the crossover phase. There are variations with this mechanism. For example, it can remove the same number of worst chromosomes as the number of good chromosomes that have been copied to the next generation.

2.2. Stock Market and Hedge Fund

Stock market is a place where companies issue stock to raise money and stock also known as equity represents the ownership of the company. Companies use this money to finance expansion, funding projects etc. This has created opportunities for investors to make money in the stock market by buying and selling equities. Investors buy and sell usually based on speculations or technical analysis. I am going to explain these terms later in this section.

Hedge Funds are a collective of investors financing a single fund, a fund manger is employed to manage the fund. He will guarantee the investors a profitable return. Due to the fund has a large amount of money, it allows fund managers to use aggressive strategies that are not available for mutual funds. I will explain these strategies later on in this section.

2.2.1. Instruments

Instruments are tools that represent monetary value in a stock market usually in a form of contract. There are many types of instruments such as stock, bonds and there are derivatives of equities such as *options* and *futures.* These instruments can be traded between investors. I will discuss the instruments that I used in the GA system in this section.

2.2.1.1. Equity

Equity (stock) signifies an ownership position in a corporation and represents a claim on its proportional share in the corporation's assets and profits. The value of equity also known as equity price is in term of the currency of the market that the company is listed in. It is one of the favorite instruments that investors use to trade in order to gain profits. In fluctuating markets, equities prices can vary very much within a very short period of time. Investors use various methodologies to predict the price and to trade in order to gain maximum profits.

2.2.1.2. Bonds

Bonds are in the Fixed Income category of instruments, the return is usually worse comparing to equities but they are more predictable and profit return is very stable. Bonds are basically loans, a bondholder has given the issuer a sum of money and the issuer can be a government, a corporation and etc. In return the issuer will pay interest to

the bondholder over a time period and after a predefined time period, the initial loaned amount will be returned. Each bond issuer has a rating and this indicates how reliable the issuer is and this allows investors to invest based on this rating. Bonds can also be traded between investors, because speculations can cause fluctuation in interest rate and the value of bonds. But the market of trading bonds is very small because fluctuation is very small especially if the bond's rating is high. Below shows the technical terms used in Bonds:

Technical terms in Bonds:	
Coupons	They are interest that the bond issuer pays to bondholder
Coupon Payments Frequency	This indicates how frequent the coupons are pay to bondholders

2.2.2. Market Indicators

The purpose of Market Indicators is to give investors indication on the overall performance of the market. This is very useful for investors to apply technical analysis when investing. In this section, I am going to explain some of market indicators I used.

2.2.2.1. Market Index

Market Index is a statistical indicator which represents the value of equities that constitute it. The index constituents are selected by analyst in the stock market board and these constituents must be able to represent the overall movement of the market, the index provides an indication of the current market's overall performance for all investors. For example, FTSE 100 index, it is composed of 100 representing equities from the UK stock market such as Lloyds TSB, Vodafone Group, etc.

2.2.2.2. Base Rate and Bank's Average Interest Rate

Base Rate is an interest rate set by national banks, it is a rate that commercial and personal banks set interest rate for their customer based on. For example, in UK, the Bank of England sets the base rate and banks such as HSBC and Barclays set their interest rate using the base rate. The bank's average interest rate shows the average of all the banks' interest rates, it gives an idea of how all banks set their interest relative to the base rate.

2.2.3. Technical Analysis

This is a method of evaluating equity that mainly uses historical asset price data to predict future market trends. It is also use to assess the risk when investing in the market. There are many technical indicators which measure the performance of the portfolios with risks associating to it.

2.2.3.1. Investment strategies based on Technical Analysis

Momentum Investment Strategy
This strategy assumes when price of equities is moving up, it will continue to move up for a period of time and on the other hand, if the price of equities are dropping, it will continue to drop for a period of time. Investors who use this strategy in their investment will buy the equities when the price starts to move up and sell the equities when the price starts falling.

Moving Average Investment Strategy
This strategy produce trading decision based on the historical price of the equity. It calculates the average price over a period of time usually 50-100 days depending on how sensitive the investor wants to know about the price movement. Moving averages are used to find the trend of price movement by smoothing out small fluctuations in price that can confuse interpretation. Investors compare the current price with the moving average indicator and make buy or sell decision. There are variations in the moving average technique such as the exponential moving average and simple moving average.

2.2.3.2. Technical Indicators
The system I am going to develop uses technical indicator as the fitness function for the GA. The indicator I am going to use is the Sharpe Ratio because it will measure risk-to-reward level for a given portfolio using the average return. The graph below will describe how the Sharpe Ratio works.

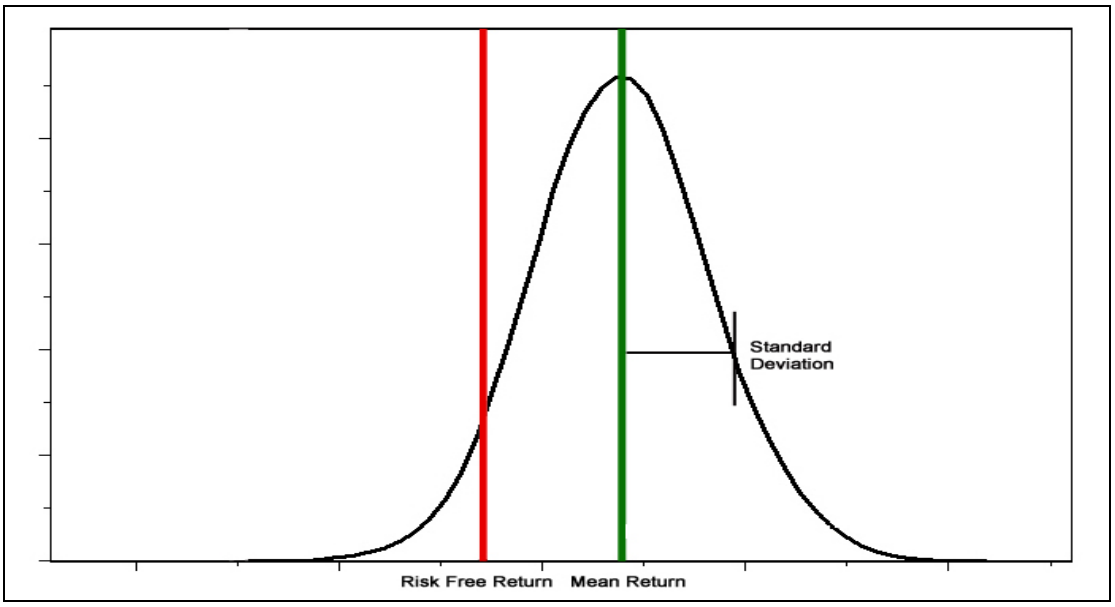


Figure 5: Sharpe Ratio Graph

Using the returns of portfolio, I can calculate the mean return and it is used to compare it will the risk free return, this will indicate the performance of the portfolio. The standard deviation of the curve is use to calculate the risk of making that return. The higher the standard deviation (a wider curve) will have a greater risk, because there is a higher

chance of getting a return that is further away from the mean, hence a very high return or a very low return is possible. So a low standard deviation with a high mean return is preferred. The Sharpe Ratio formula is shown below:

$$Sharpe\ Ratio = \frac{Mean \text{ Return - Risk Free Return}}{\text{Standard Deviation}}$$

Chapter 3

3. Design

This chapter will give the design detail for the system and the system consists of two sections: Genetic Algorithm (GA) and Investment Simulator (IS). Firstly I will refine the technical objectives suggested in chapter 1. Following that I will talk about the experimental design I did and then I will explain the design of functions in the GA and the IS.

3.1. Objectives

Genetic Algorithm (GA)
To customize the standard GA so that it will find the best possible solution (solution that is very close to the global optimum) given a predefine environment. The GA must find the best possible solution within an acceptable complexity and it must also be able prevent premature convergence of the population of Chromosomes.

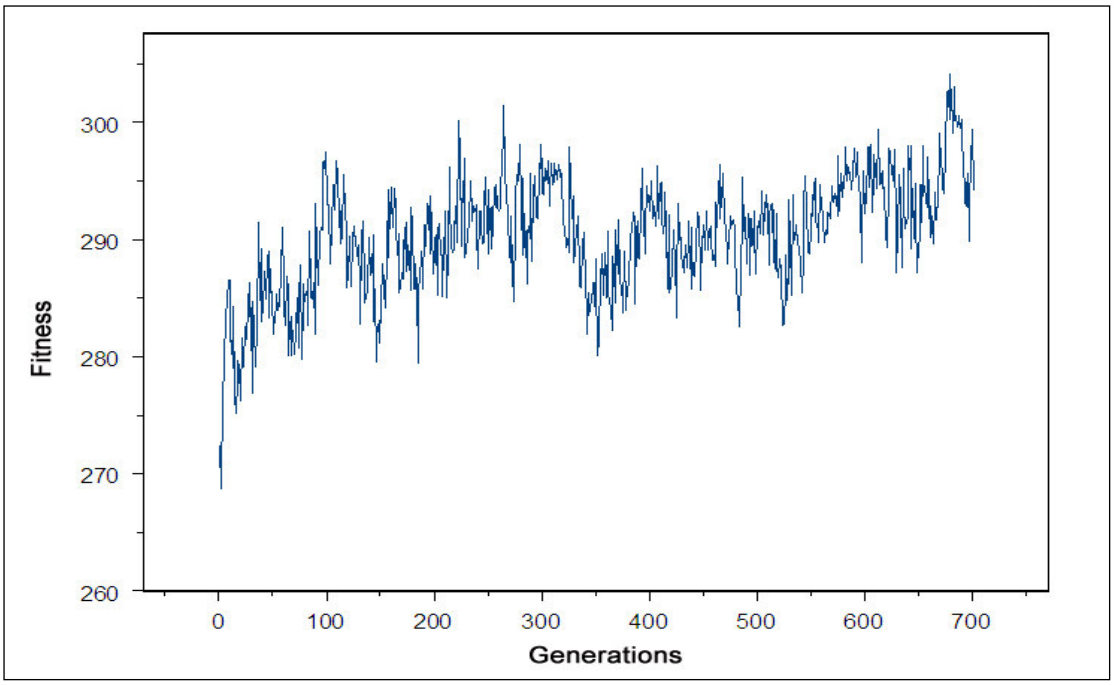
Investment Simulator (IS)
To design a simulator that can simulate various types of trading in the stock market. It must be able to apply investment strategies when trading and return a fitness for the strategies control integers (chromosomes).

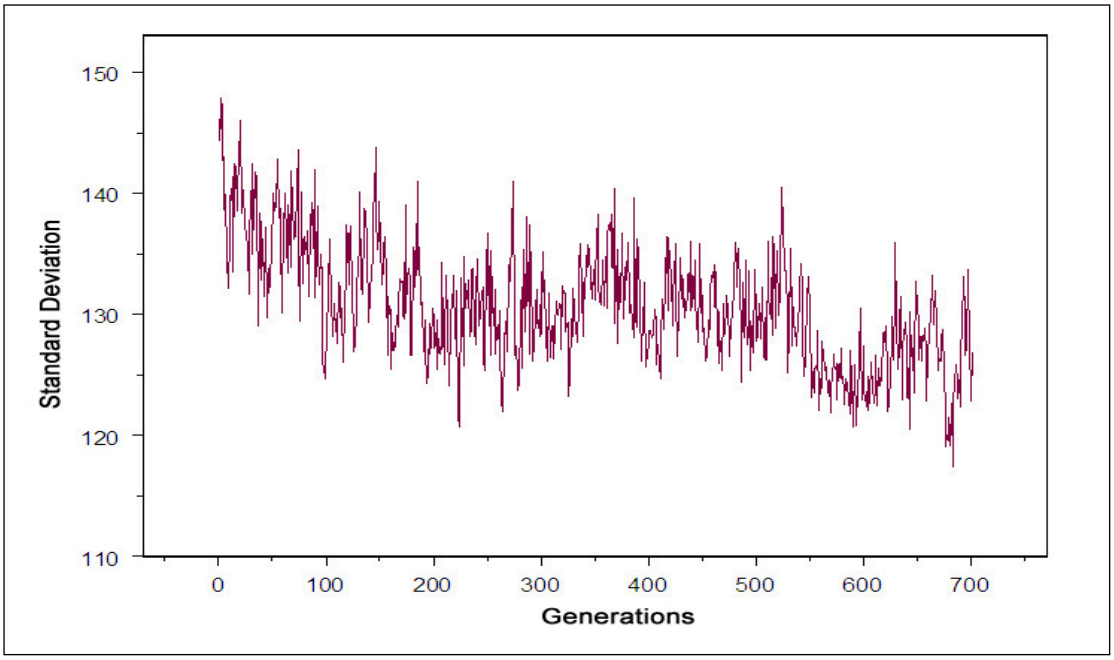
3.2. Experimental Design

Before the primary design stage some experimental designs and preliminary testing were done, it was mainly on the genetic algorithm. The genetic algorithm was designed using a dummy simulator. It will take in a chromosome which has 5 numbers, each number is randomly chosen between 0-100, the fitness value will be the sum of the 5 numbers[3] hence the highest fitness value will be 500.

Experiment Setting
Each GA will run for 700 generations and the best chromosome's fitness will be recorded. The experiment will be repeated for 20 times. For each generation, the mean and standard deviation of the best chromosome's fitness will be calculated using the data obtained from the 20 separate repeat runs. They will then be plot on the graph to show the performance of the GA (the graphs with blue lines are the mean fitness of the best fitness and the graphs with the purple lines are the standard deviation of the best fitness):

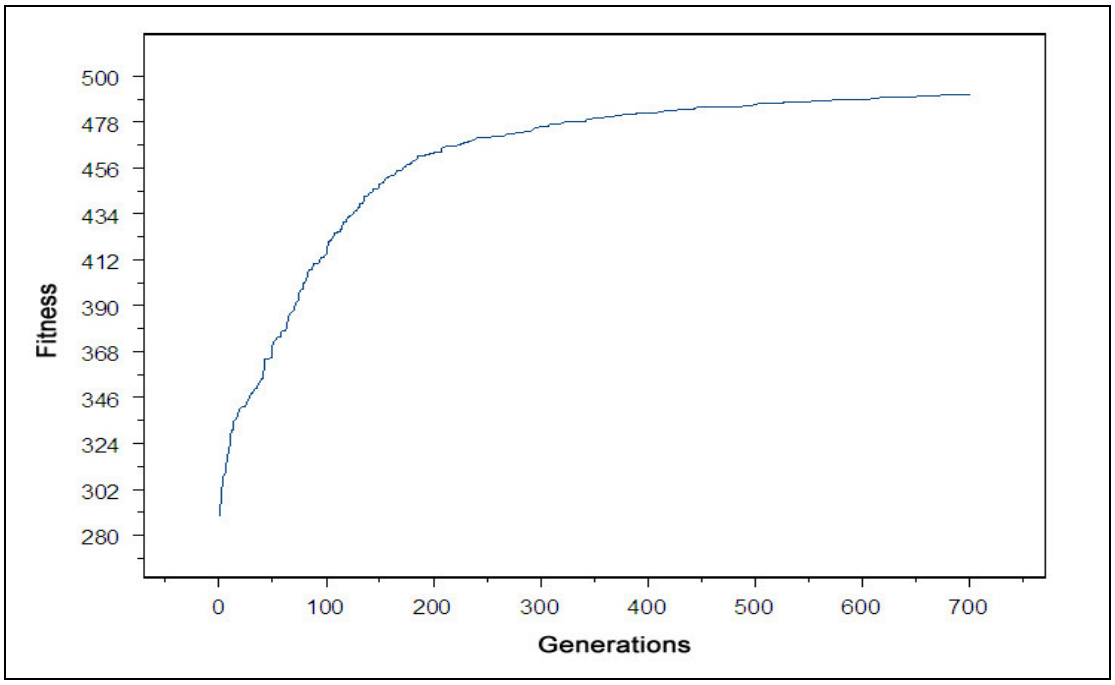
[3] For example, a chromosome of [12][13][15][80][57], the fitness of this chromosome would be 177.

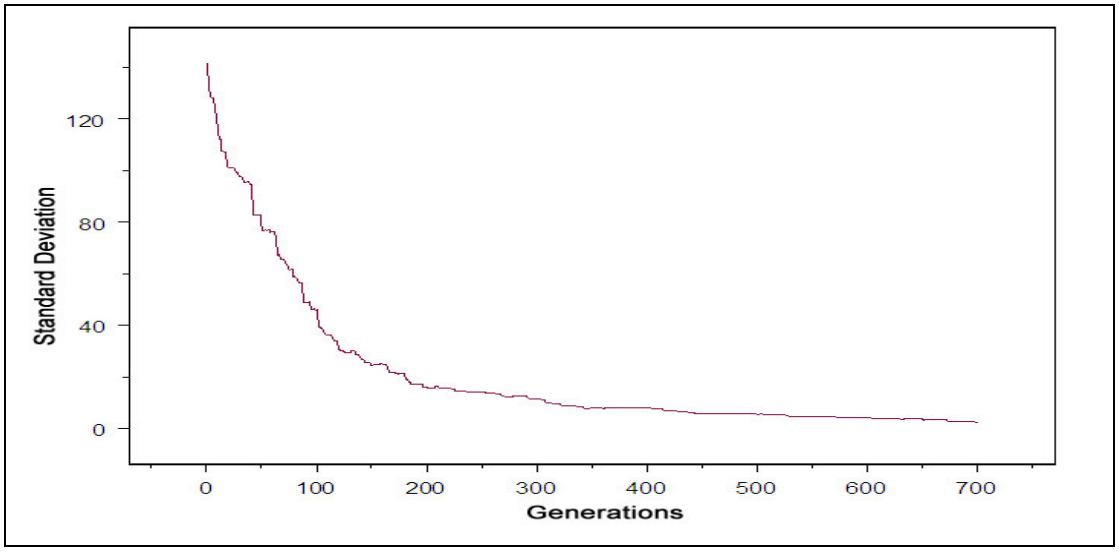




The graph above shows the performance of the *Exp GA 1* displaying the mean best fitness and standard deviation of population in every generation.
The Specification of GA:

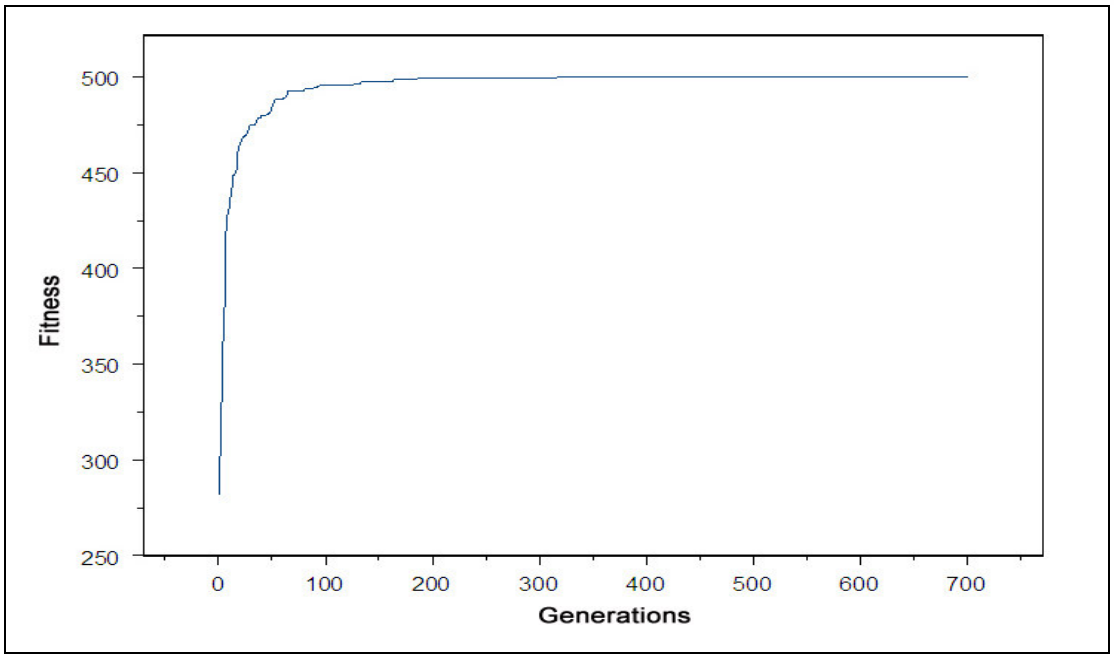
- *Single fixed point of crossover.*
- *Elitism not applied.*
- *Population size: 400*
- *Roulette Wheel selection applied.*
- *Mutation Rate: 1%*

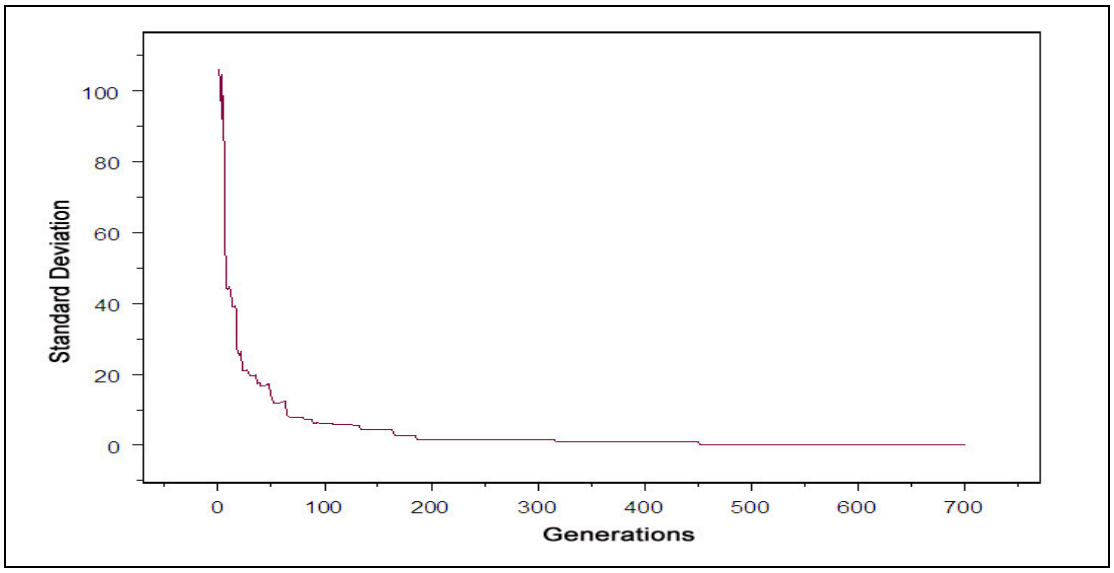




The graph above shows the performance of the *Exp GA 2* displaying the mean best fitness and standard deviation of population in every generation.
The Specification of GA:

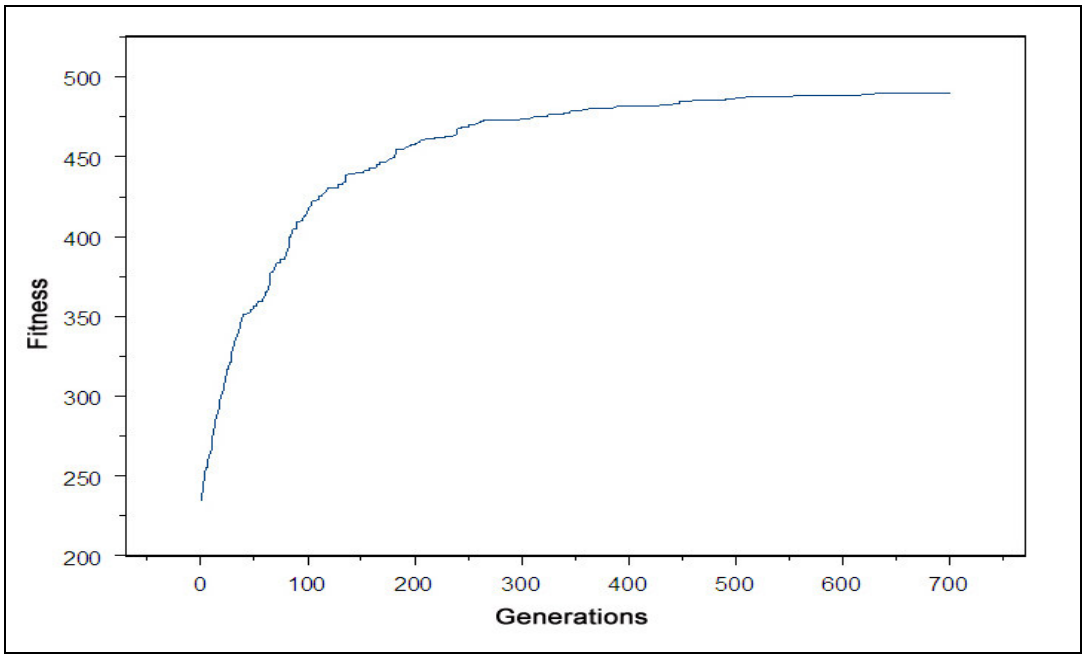
- *Single fixed point of crossover*
- *Elitism applied.*
- *Population size: 400*
- *Roulette Wheel selection applied*
- *Mutation Rate: 1%*

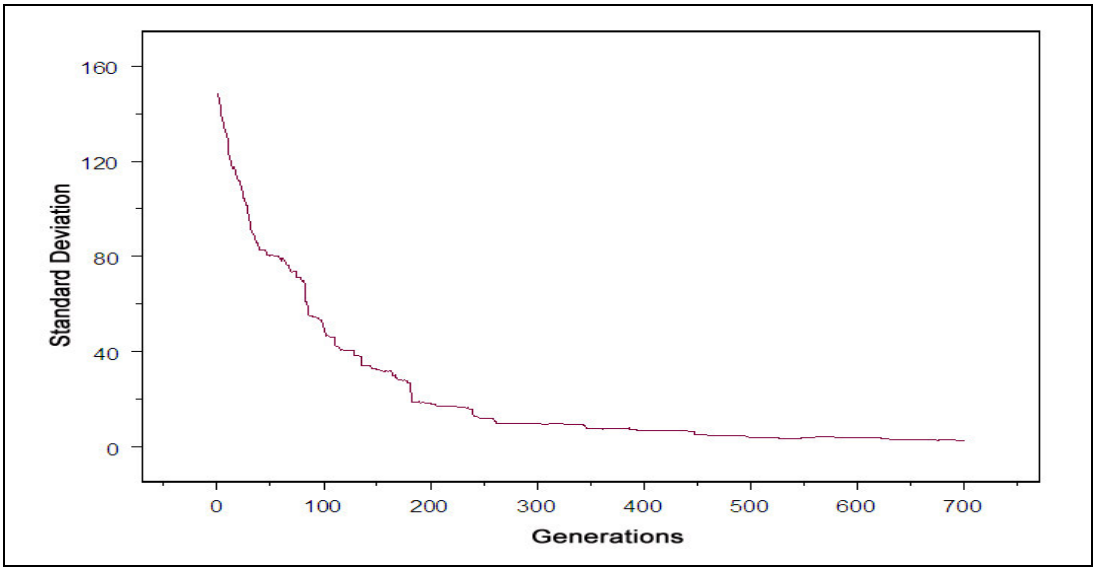




The graph above shows the performance of the *Exp GA 3* displaying the mean best fitness and standard deviation of population in every generation.
The Specification of GA:

- *Single fixed point of crossover*
- *Elitism applied.*
- *Population size: 400*
- *Roulette Wheel selection applied*
- *Mutation Rate: 1%*
- *50 randomly generated chromosomes added*





The graph above shows the performance of the *Exp GA 4* displaying the mean best fitness and standard deviation of population in every generation.
The Specification of GA:

- *Single random point of crossover*
- *Elitism applied.*
- *Population size: 400*
- *Roulette Wheel selection applied*
- *Mutation Rate: 1%*

3.2.1. Evaluation of Experimental GAs

Exp GA 1
From the graph, there are evidences that suggest some good chromosomes are destroyed from generation to generation. The sudden rises and drops in the mean fitness show that in the crossover phase, excellent parent chromosomes are destroyed after recombination and their offspring have lower fitness value, so they do not perform well in this given environment but they are carrier of good genes. The graph also shows that the rate of evolution is slow, because the after 700 generation, the fitness on average is around 280 which is still not close to the best possible fitness of 500. The standard deviation of fitness is fluctuating vigorously, which suggests that the GA is evolving unstably.

Exp GA 2
In this GA design, I have included *Elitism*. It has solved the problem of destroying excellent parent chromosomes during the recombination phase and the graph shows that the GA can achieve a fitness of 490 by the 580^{th} generation, this suggests that the rate of evolution has improved. The non-fluctuating mean fitness and standard deviation shows that the GA is evolving steadily. But in the period of 600^{th} generation to 700^{th} generation, it shows that the fitness is increasing very slowly, this indicates that premature convergence is occurring and evolution in that period is only powered by the mutation.

Exp GA 3
This GA is a customized version of *Exp GA 2*. Exp GA 3 has an extra phase in the GA cycle. After the mutation phase, a set of 50 randomly generated chromosomes are add to the population for the next GA cycle. This is a different form of mutation that applies to the whole population instead of a gene in a chromosome. The mean fitness graph shows that the GA has evolved very quickly to the best possible fitness (500) after 150 generations, this also suggests that premature convergence did not occurred. But this method forces the GA to do more random search than evolving the current population, if the size of randomly generated chromosome is large this can result in increasing the time complexity of the overall system. Another method for preventing premature convergence was suggested by Schaffer, J. D. and Eshelman, L. J. in 1991 [15], they suggested that it can be prevented by disallowing incest in crossover, this means it will not allow chromosome with similar fitness to crossover. But in this system, the problem of premature convergence only occurs when the fitness is very close to the best fitness, so this method was not used. This idea of adding random chromosomes is derived from macro-mutation first proposed by T. Jones in 1995 [18] and it was experimented in L. Angeline's study in 1997 [20] and Huelsbergen's study in 1998 [19]. In Huelsbergen's study, he designed a genetic programming system (GP) which uses the idea of adding random chromosomes and it was compared with a standard GP. He has found that the customized GP with the adding random chromosomes feature has a better performance out of the two.

Exp GA 4
This GA uses random point crossover instead of the fixed point crossover. The mean fitness graph shows that the GA has a reasonable rate of evolution and it is very similar to

the rate of evolution of Exp GA 2, which suggests that the two different crossover mechanisms (fixed point and random point) have very similar effect on the rate of evolution in this problem, more test has to be carried out ensure this for more complex problems.

3.3. System Architecture

This section will outline the final design of the system, the design will mainly be separated into two parts: Genetic Algorithm (GA) and Investment Simulator (IS). Firstly I will describe overall structure of the system. The UML diagrams for the system are given in Appendix D. The diagram below shows the overall structure of the system:

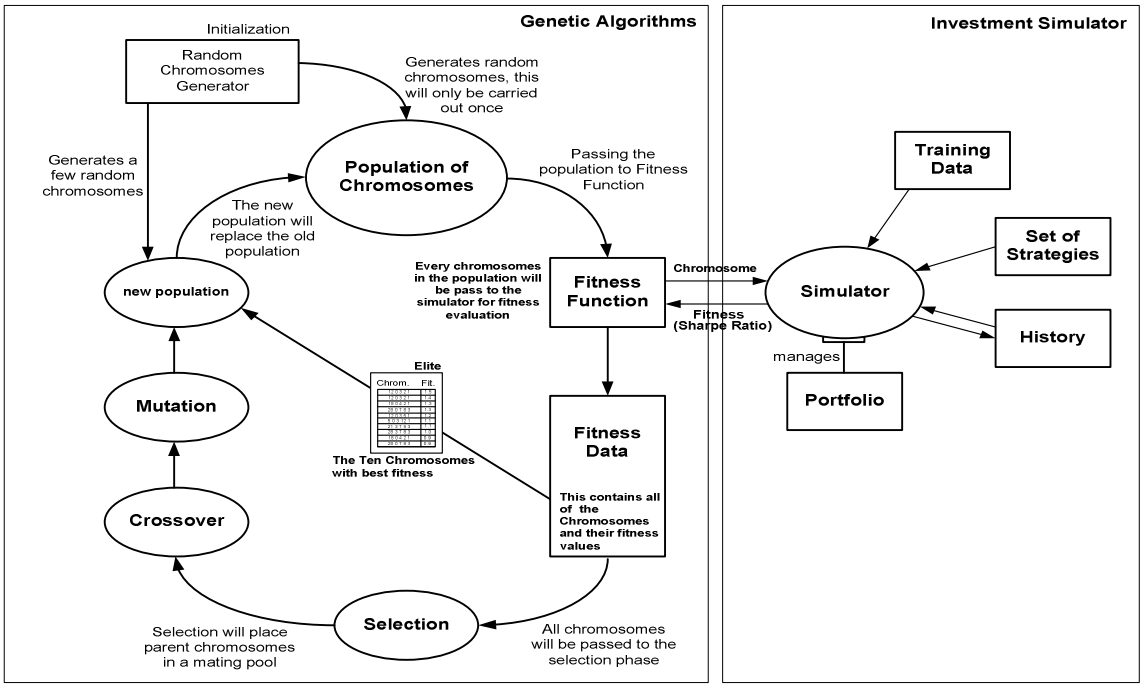


Figure 6: The overall Structure

3.3.1. Description of the cycle

The system initially starts at the GA, the GA randomly generates a set of chromosomes, and then the set is passed to the fitness functions. In the fitness function, the chromosomes are simulated one by one using the investment simulator. The investment simulator manages a portfolio that the instruments are selected by users. It makes trading decisions based on information such as historical training data and strategies. Historical training data is in the form of time series, so the IS cycle through the training data and make trading decisions. Further explanation will be given in section 3.3.3. After the simulations have been carried out on the chromosome, a fitness value will be returned and stored with the chromosome itself. This process will continue until all chromosomes in the population have been simulated and then the set of chromosomes are sorted in descending order of fitness, the top chromosomes are copied and stored as the elitisms and they will be place into the population for the next generation. After the

elitism process, the set of chromosomes with the fitness will then be passed to the selection phase of GA in which roulette wheel selection will be use to select the parents for the crossover phase. In crossover phase, two parents are selected and recombined to form offspring chromosomes. This process will repeat until the maximum size of population[4] has been reached. After that the mutation phase will start, and then the elitisms and the randomly generated chromosomes will be added to the population for the next generation. The GA cycle will start again and it will repeat until the best solution is found. The design of each functions in GA and IS will be discus in detail in sections 3.3.2 and 3.3.3.

3.3.2. Genetic Algorithm (GA)

3.3.2.1. Chromosomes

It is a user defined type. It contains the following variable types:

Variable Name	Explanations
Fitness	This variable has the type *double*, it stores the value of fitness that is calculated in the fitness function (Investment Simulator).
Roulette selection chance	This variable has the type *double*, it stores the probability of getting selected in the roulette wheel, it is relative to the fitness of this chromosome.
Genes	This variable has the type *Array*, it stores a integer in each index of array, and the length of chromosome can be set by setting the length of array.

Figure 7: Chromosome's Data Type

The chromosome class contains setter and getter methods for each of the variables, they are used throughout the GA module. When a chromosome is created, variables Fitness and Roulette selection chance are null, a value will be assigned after the chromosome have been simulated in the simulator.

3.3.2.2. Initialization

Initiation phase starts the whole GA cycle and it will only be used once. The purpose of this phase is to randomly generate candidate chromosomes for the population. The mechanism is show in the diagram below:

[4] Maximum size of pop. = Original size of pop. – (size of elitism + size of random generated pop.)

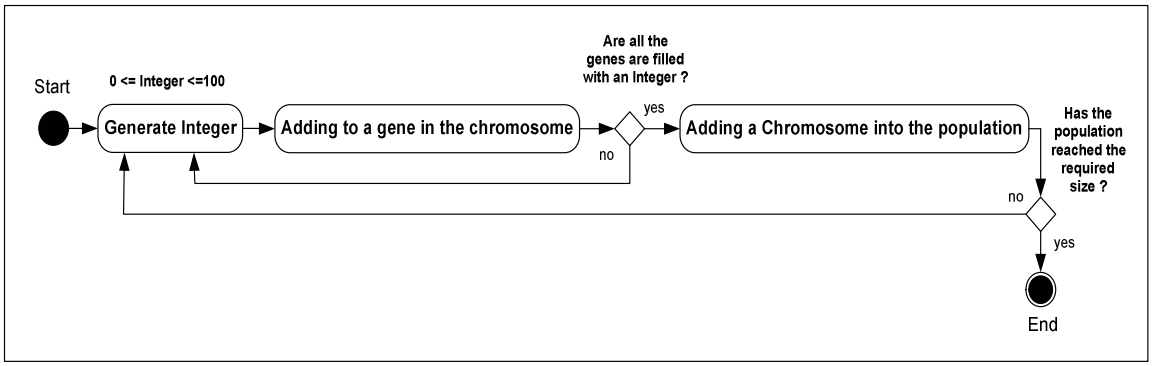


Figure 8: Initialization phase

It randomly generates an integer between 0 and 100. The integer is then added to one of the gene in the chromosome, this process is repeated until all genes in the chromosome are filled. After a chromosome is created, if the population has not reach the required size, the whole process is repeated to create a new chromosome.

3.3.2.3. Fitness Function

This phase will pass the chromosomes one by one to the Investment Simulator (IS), IS will calculate and return the fitness of the chromosomes. The fitness function will write the fitness into the chromosome's variable.

3.3.2.4. Selection

Selection phase will start after all chromosomes have been simulated for fitness. It will use the fitness of chromosomes to calculate a roulette selection chance. It is calculated using the formula below:

$$\text{RouletteSelectingChance} = \frac{\text{Fitness}}{\sum \text{Fitness}_n}$$

Where n = number of chromosomes

Using the roulette selecting chance, the system selects the chromosome by randomly indexing into the array of chromosomes (population) and then a random number is generated and it is use to compare with the Roulette Selecting Chance. If the random number is less than the roulette selecting chance, the chromosome will be selected as a parent for the crossover process. This process will repeat until the number of parents required has been attained. These parent chromosomes will be store in the mating pool[5].

3.3.2.5. Recombination

In Recombination phase, the system will select two parents from the mating pool randomly. Recombine them using and storing the offspring into the children pool[6]. From the results of the preliminary testing, it shows that fixed point and random point crossover have similar effect on the rate of evolution. In this system I am going to apply the fixed point crossover to experiment the effect of this mechanism in the system on the stock market.

[5] Mating Pool – An array of chromosomes that are waiting for recombination.
[6] Children Pool – An array of offspring that was created by cross over of parents.

3.3.2.6. Mutation

Mutation phase will cycle through every gene in every chromosome in the population. For every gene in the chromosome, a randomly number between 0 and 1 will be generated and if the number is less then the mutation rate in decimal number(1% = 0.01), another random number between 0 and 100 will be generated and replace the number in the gene.

3.3.2.7. Elitism

In the selection phase, after all chromosomes have been simulated for its fitness, the chromosomes will be sorted in descending order of fitness and the top chromosomes will be copied into a new array. The top chromosome array will be attached to the population array after the mutation phase. Elitism act as a shortcut for the best chromosomes but these elites are not eliminated from the original population, they are still available for recombination.

3.3.2.8. Adding Random chromosomes to population

This mechanism is derived from the idea of macro mutation suggested by T. Jones in 1995 [18] and it was used by Huelsbergen in 1998 [19]. This process introduces greater level of random search in the GA system. It occurs after the elites have been added to the population. Chromosomes are generated using the same mechanism in initialization. Users can define the number of chromosomes to be generated.

3.3.3. Investment Simulator (IS)

The IS has a complex architecture, so I will separate it into 5 parts: Formation of Investment strategies, making trade decisions using historical data, Executing decisions and updating portfolio, storing portfolio returns into history and calculating fitness. In the end I will integrate these parts to give an overall design.

3.3.3.1. Formation of Investment Strategies

In IS, Investment Strategies are predefined by users and these investment strategies are controlled by chromosomes, so integers from the chromosome must be extracted and applied to these strategies before they can be use to make trading decisions. The diagram below will show the process of applying control integers to strategies.

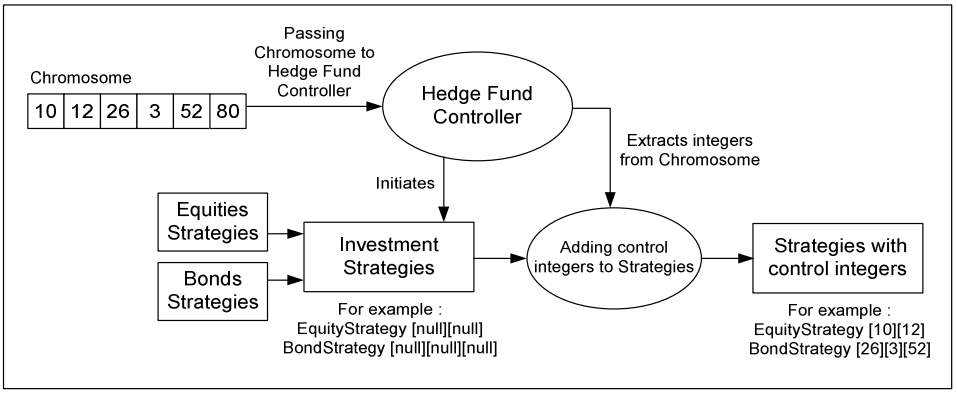


Figure 9: formation of Strategies

Chromosomes are broken down into control integers and an instance of the strategies will be initiated. Control integers are then added to the strategies and the investment strategies will be ready for simulation. The allocation of genes to strategies will be explained in chapter 4 (section 4.2.1.1)

3.3.3.2. Making Trade Decisions

The diagram below shows the process of make decision based on price and index data with the strategies:

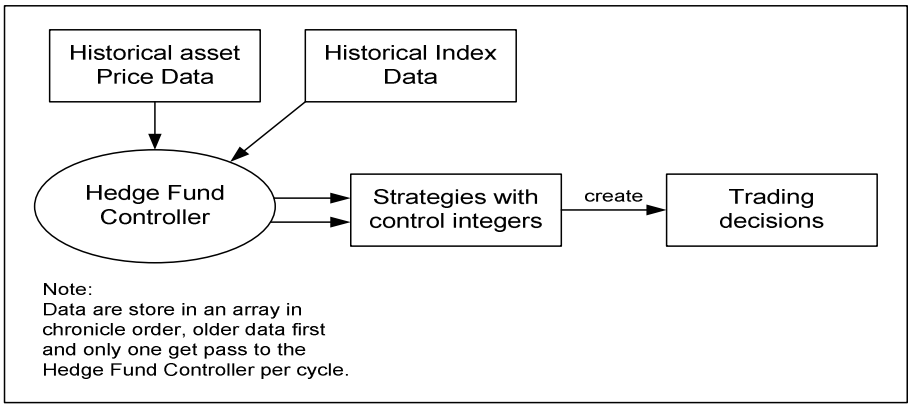


Figure 10: Making Trade Decision

All market information is stored in an array which is originally imported from files. Only one set of data is sent to the Hedge Fund Controller per trading week[7] and after making the trading decisions, they are executed and the portfolio will be updated. The cycle will start again with the second set of data. The process will terminate when all data in the array have been simulated.

3.3.3.3. Executing Decisions and portfolio update

This section describes the execution of decisions and portfolio updates, the process is shown in the diagram below:

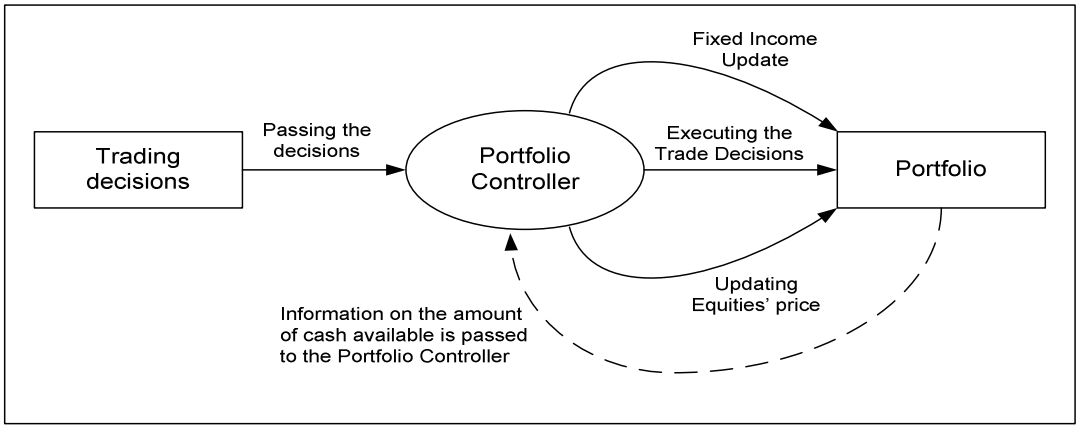


Figure 11: Executing Decisions and Portfolio Updates

[7] If the system takes in weekly data over a time period of one year, there will be 52 cycle in the system.

Trading decisions are passed to the portfolio controller and these decisions get executed if there is enough cash in the portfolio. After executing decisions, the portfolio gets updated by the asset price data and then the portfolio controller will update all the fixed income instruments such as cash[8] and bonds.

3.3.3.4. *Storing Portfolio returns in History*

The purpose of keeping a record of all Portfolio returns is that the fitness of chromosomes can be calculated using this information. Portfolio returns are written into history when all trading decisions have been executed and the portfolio has been updated.

3.3.3.5. Fitness Calculation

The fitness for chromosomes in Investment Simulator is the Sharpe ratio. A high value represents the portfolio is achieving a high profit with low risk, so higher value represents the portfolio is performing well. It is calculated using the portfolio's return per time unit (week) and the risk-free return. The Sharpe's ratio formula is shown below:

$$\text{Sharpe Ratio} = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i - x_{risk-free}}{\sigma_{\text{return}}}$$

$x = \text{Portfolio Return}$

$x_{Risk-Free} = \text{Risk Free Return of Portfolio}$

$\sigma_{\text{return}} = \text{Portfolio Return's Standard Deviation}$

The expected return and the standard deviation of return can be calculated using the set of portfolio return stored in the history. The risk free return can be calculated using average bank deposit interest rate. The compound interest is used as the risk free return and it can be calculated using the formula below:

$$Compound\ Interest = P(1+r)^n$$

$P = \text{Initial Cash Amount}$

$r = \text{Interest Rate in decmial}$

$n = \text{Number of Interest paid}$

[8] Reasons for declaring cash as a fixed income instrument will be explained in section 4.2.2.1

3.3.3.6. Overall Workflow of IS

The five parts explained before are integrated to complete the IS. The procedures are shown in the sequence diagram below:

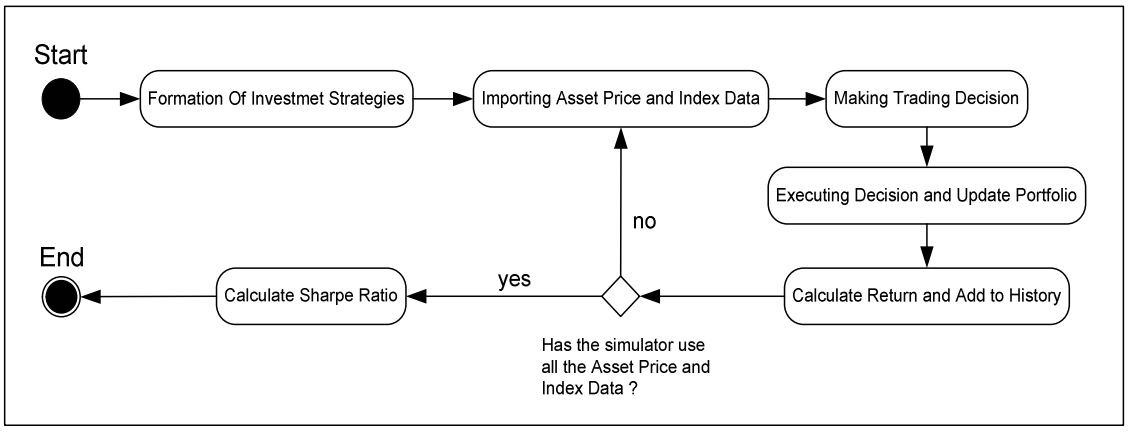


Figure 12: Overall workflow IS

3.3.4. Other Operations

There are other operations that do not belong to GA or IS, but they are essential. I am going to explain them in this section.

3.3.4.1. Fitness logging

This is used to write data such as best fitness or average fitness of the chromosome population into a separate file for fitness analysis, this will allow fitness to be exported to Excel for graph drawing and other experimental work.

3.3.4.2. System Timer

This is used to measure the time complexity for actions in the system, the timer starts in the initialization phase of GA and stops when the optimized solution is found.

3.3.4.3. GA Configuration and IS Configuration

This allow user to change the setting of the GA and IS. Users can change the population size, number of generations to be run, length of chromosomes, size of the elitism and number of parent chromosomes for GA and provide the initial environment for the IS. e.g. Setting the initial portfolio, specifying the file that contains data.

3.3.4.4. Data Entry

This will import all data that the investment simulator requires, the user inputs the filename of the file that contains data, the user also inputs the type and the size of data and then it will import these data into an array for IS to simulate.

3.3.5. Design Features

In IS, I applied the strategy pattern when design investment strategies, I defined a family of investment strategies and encapsulating each variation of the strategy into a separate class. This allows strategies to be interchangeable and also adding strategies become easy for further development.

When designing investment instruments, I used the decorator pattern. Instruments are separated into different categories and each instrument has its only class. All the instruments are in a hierarchical structure, the common functions are moved up in the hierarchy and the less common functions will remain in the subclass.

When integrating the GA package and the IS package, I applied the mediator pattern. I created an interface class (*fitnessFunction*) in the IS package. A concrete class *Simulator* implements the interface and creates an instance of *HedgeFundController* with information from the IS configuration (IS configuration (IS_Conf) is a user defined data type in a different package) and starts the simulation process. Another advantage for using the mediator pattern is that it encapsulates classes from other classes and prevents interactions between classes that are not supposed to communicate. It also makes the GA more pluggable because other simulators can be used in the system as long as they implement the *fitnessFunction*.

The class Portfolio in IS can be serialized to an object (an array of bytes). This allows user defined objects to be passed by value instead of passing the reference only. This is very important because during simulation phase, the population of chromosomes must be simulated using the same initial portfolio. But if the portfolio is passed to the simulator by reference, only the reference is updated and when the test has completed, the reference of the end portfolio gets send to the next chromosome test and it will be used as the initial portfolio of second chromosome's test which will give an incorrect fitness. On the other hand, if the portfolio is passed to the simulator by value, every time a new test is initiated, a new copy of portfolio is passed to the simulator, hence the original copy (serialized object) remains unchanged and be available for the next test.

Chapter 4

4. Implementation and Testing

This chapter will describe the implementation of the design and testing of the implemented system. This will start by stating all the applications used in developing the system. It will follow discussing the implementation issues in GA and IS such as Representation of Chromosomes, Investment Strategies implementation, implementation of GA operators and etc. After that I will explain the difficulties encountered during implementation and finally it will give details on testing done on the system.

4.1. Software used in Development

Java 2 Standard Edition JDK 5.0
The system is written in Java programming language and Sun Java compiler was used to compile the system. J2SE runtime environment was installed to run Java applications. Java was chosen as the programming language for the system because the Object Orientated feature in Java will be more efficiency when designing IS, but the trade off is that the system will have a higher time complexity, since Java is considered slower comparing to other languages such as C.

Eclipse SDK 3.02
This is an Integrated Development Environment (IDE) for Java development, it provides an area for writing Java programming language and it will also auto compile the source and it will also provide a debug function to remove bugs in the system. This is use to increase the efficiency of implementation of design.

Omondo EclipseUML Studio 2.0.0
This is an external plugin for *Eclipse*, it allows users to create UML diagrams in *Eclipse,* and generates the basic UML diagrams from source code. It also provides a low level of MDA (Model Driven Architecture) development.

S-Plus 6.2 Student Edition
This software is use for data analysis, it also provides spreadsheets for calculating data and drawing graphs such as 3D landscape chart, scatter diagram and etc. This software can also find correlations and apply statistical functions in data.

JUnit
This is a external library for Java and it can be used in eclipse, it provides a testing framework for unit testing. Methods for assessing the output of a function are provided in the JUnit library

4.2. Implementation Issues

4.2.1. Genetic Algorithm

4.2.1.1. Chromosome Representation

The chromosome representation is directly related to how fitness is evaluated using the user configured simulator. In the design the fitness is evaluated by applying the user customized strategies on historical training data and the simulator would monitor the rate of return (ROR) in the portfolio and comparing it with the risk free return, hence give an indication on how well the customized strategies are performing in the environment (The Stock Market). This has defined the solution domain for the system, hence the representation of chromosome will be a set of integer values and each of the value will be control integers for one or more investment strategies. For this system, I defined the representation of chromosome as shown below:

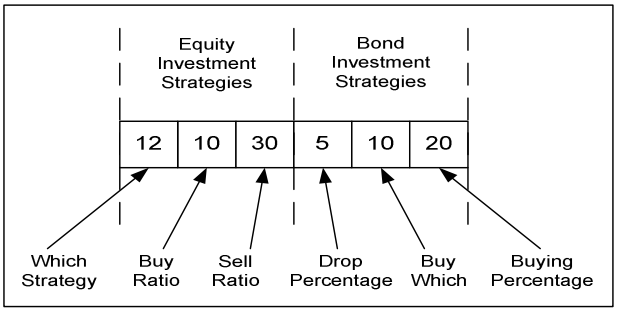


Figure 15: Chromosome Representation

Figure 15 shows the representation of chromosomes in the system: Gene 0, 1, 2 in the chromosome are reserved for Equity Investment Strategies. Gene 3, 4, 5 are reserved for Bond Investment Strategies. *Which Strategy* will select which equity investment strategy to be used, *Buy Ratio* and *Sell Ratio* in the strategies will affect the trade amount of equity, *Drop Percentage* represents the percentage drop in a market index (specified by the strategy) before buying bonds, *Buying Which* will influence the strategy's choice on buying a bond from a list of bonds and *Buying percentage* will decide the amount of bond that the strategy will buy. This idea of grouping genes in the chromosome was suggested by E. Falkenauer in 1994 [8]. Understanding of integers varies from strategies, so I will further explain the representation of these integers corresponding to each strategy in section 4.2.2.3.

4.2.1.2. GA Parameters

The system has a seven control parameters: number of generations, chromosome length, population size, elite size, mating pool size, random chromosomes addition size[9] and mutation rate. These parameters are closely related to each other, De Jong [7] suggested in his study on Genetic Algorithm in 1975 that the mutation rate should be inversely proportion to the population size. I am going to test these parameters by changing one of the parameter while keeping all other parameter the same. Firstly I will find the best

[9] This is the number of randomly generated chromosomes added to the population after each generation.

solution using high values for the parameters and repeat it for a 20 times to ensure the solution I obtain is the best solution. The best solution can be used to compare with solutions obtained by systems that are configured differently. The table below shows the configuration and the results 3 separate runs of the system. This test was carried out using the investment simulator (IS) with 3 equities and 3 bonds.

	Generation	Pop size	Elite size	Mating pool size	Random size	Mutate Rate (%)	Time	Best Fitness
1	1000	500	20	250	100	1	1516	0.653
2	1000	500	20	250	100	1	1480	0.658
3	1000	500	20	250	100	1	1778	0.654

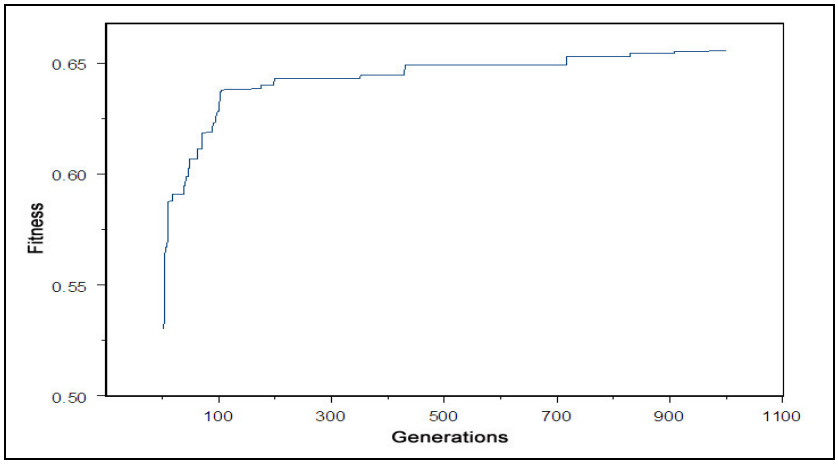


Figure 16: 3 repeat run of the over configured System

The parameters of the system is set relative to the population size: Elite size is 4% of the population size, mating pool size is 50% of the population size, random chromosome addition size is 20% of the population and mutation rate is 0.2% of population size. The graph above shows the mean fitness for every generation of 20 separate runs of the system and the final solution they found has a very similar fitness. The small difference occurs in the third decimal place and Sharpe ratio is usually rounded to two decimal places. This solution can be declared as the best solution because the number of generation set for the system was excessively high. This configuration for the system performs well, but the time complexity is too high (finding the best solution in an average time of 1591.3 seconds (26 minutes)). I am going to improve the time complexity by altering the configuration while retaining the accuracy of the best solution. I am going to set this fitness (0.65) as the target for other configuration to achieve. Firstly I am going to try lowering the population size to improve the complexity.

In Schaffer, Caruana, Eshelman and Das's [16] study and C.R. Reeves' study in 1993 [14], they suggested that a small population size would give the most optimizing but Goldberg [9] has argued that a larger population size would give a better performance. GA experts have different estimation on the population size but all the population size that they suggested only covers a very small proportion of the solution domain[10]. To find

[10] Population size of 100 in a solution domain of size 100^6 is only covering $1*10^{-1}$ % of all solutions.

the best population size for the system, I will draw a graph using fitness of best solution suggested by the system and find the general trend when I increase the population size. I can use the trend to estimate a population size that will give the best performance in term of time complexity and accuracy of the best solution. The following graph shows the trend of the population size affecting the best solution found by systems.

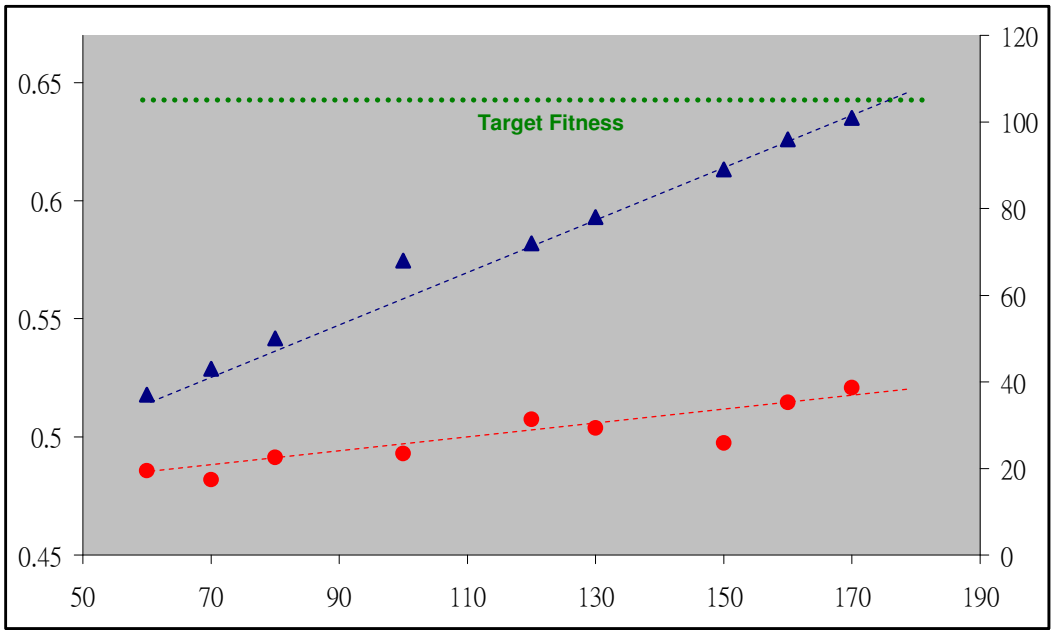


Figure 17: The Trend of Population Size affecting the best Solution found

The left Y-axis represents the fitness value of the best solution, the right Y-axis represents the time complexity and the X-axis is the population size. The configuration of system for this test is shown in appendix A. Parameters for the configuration are set relative to the population size[11]. The blue line shows the trend of time complexity as the population size increase. The red line shows the trend population size affecting the best solution's fitness. The green line is the best solution's fitness.

Using the trends, I can mathematically estimate the minimum population size and its estimated time complexity that can find the best solution. It is because all the points on the graph are found using the system with a predefined configuration as shown in the table below:

Generation	Pop size	Elite size	mating pool	Random size	Mutate Rate	Time
250	60	2	30	12	0.12	37
Run 1	Run 2	Run 3	Run 4	Run 5	AVG	
0.4713691	0.468727	0.49394	0.498605	0.49602768	0.485734	

[11] Elite size is 4% of the population size, mating pool size is 50% of the population size, random chromosome addition size is 20% of the population and mutation rate is 0.2% of population size.

Generation	Pop size	Elite size	Mating pool	Random size	Mutate Rate	Time
250	70	2	35	14	0.14	43
Run 1	Run 2	Run 3	Run 4	Run 5	AVG	
0.4775445	0.490567	0.47928	0.472468	0.49001488	0.481975	

Generation	Pop size	Elite size	mating pool	Random size	Mutate Rate	Time
250	80	3	40	16	0.16	50
Run 1	Run 2	Run 3	Run 4	Run 5	AVG	
0.4841067	0.496013	0.475989	0.497255	0.50345066	0.491363	

Not all tables are shown here, please look in Appendix A for the whole list of tables

Figure 18: Tables of test runs and results

Each parameter is set relative to the population size except the number of generations to be run by the system and the number of generations is constant (250). This mean if I extend the red best fit line in the graph until it crosses with target fitness line, the x value (population size) of cross point will be the minimum population size that the system will find the best solution within the predefine number of generations (250). To do that, I select two points on the best fit line, using those two points I calculated the gradient of the line and then I derived the equation of the best fit line by selecting two points on the line. The equations of best fit lines on the graph are shown below:

$$y = 0.0003x - 0.571 \qquad (1)$$

$$z = 0.6x - 156 \qquad (2)$$

(1) is the equation for the best fit line of the scatter diagram *population size* against *fitness*. Where x = population size and y = the average best fitness found by the system
(2) is the equation for the best fit line of the scatter diagram *population size* against time *complexity*. Where x = population size and y = time complexity.
For the population size test, if I set y = the target fitness for equation (1), x will be the population size that the system will find the target fitness on average run and if I use it for the x value in equation (2), I can find the time complexity that the system can achieve the target fitness. Using the figures from the population size test, the system with the predefined parameters can find the best solution in 250 generations if the population size is 4070 and the time complexity of the process will be 2286 second. This shows if decreasing the number of generations by ¾, it will required the population size to increase by 8 times in order for the system to find the best solution. This also shows increasing population size will have a greater increase in time complexity, so increasing the number of generations is more favorable which agrees with J.J. Grenfenstette and J.M. Fitzpatrick's study in 1985 [10].

Modifying other parameters in the configuration can also improve the time complexity, because each parameter will have an effect of the gradient of the best fit line.

The higher the gradient, the smaller the population size will be required. This idea was suggested in M. F. Bramlette's study in 1991 [5]. In the last test, parameters are set relative to the population size:

Elite size is 4% of the population size
Mating pool size is 50% of the population size
Random chromosome addition size is 20% of the population size
Mutation rate is 0.2% of population size

I examined the *Random chromosome addition size*, because it is the main function to prevent premature convergence of the population. I examined using the same method as the last test, but changing the relative percentage of *Random chromosome addition size*. The test percentages are 10%, 40%, 60%, 70% and 80%. The result is shown in the graph below:

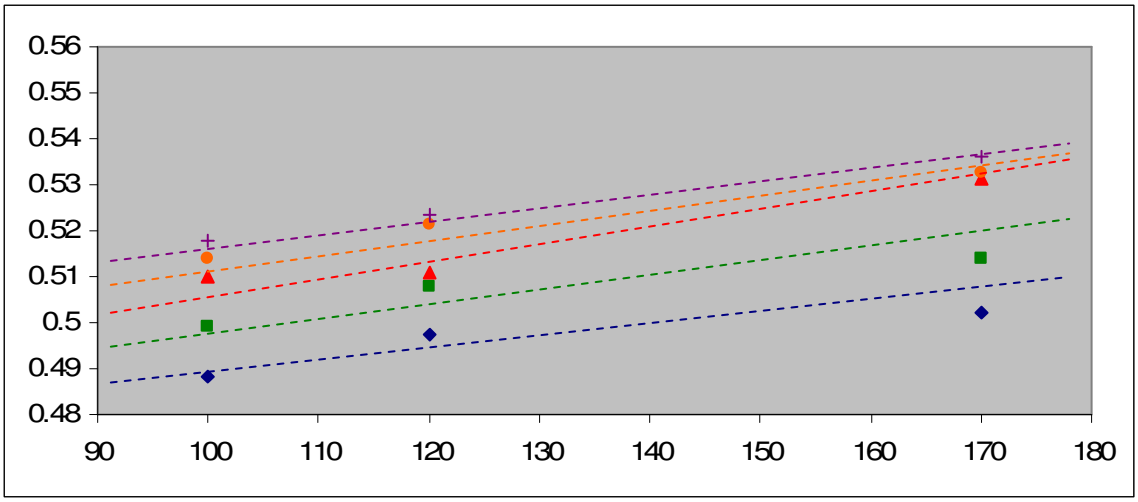


Figure 19: The Trend of Population Size with 5 different random chromosomes addition percentage
Parameters for the configuration of the system is shown in appendix B

X axis: Population size
Y axis: Best solution's fitness value

Red Line: best fit for the system with a random chromosomes addition size of 60% of population size.
Green Line: best fit for the system with a random chromosomes addition size of 40% of population size.
Blue Line: best fit for the system with a random chromosomes addition size of 20% of population size.
Orange Line: best fit for the system with a random chromosomes addition size of 70% of population size.
Purple Line: best fit for the system with a random chromosomes addition size of 80% of population size.

The graph shows as the size of random chromosomes addition increase, the gradient of the curve increases until the percentage of added chromosomes is about 70%. Which means the line of best fit can reach the best fitness earlier as the percentage of added random chromosomes increases in the graph; hence a smaller population size will be needed and this improves the time complexity. But as the percentage of added random chromosomes is near 70%, the increase in gradient stops because it forces the system to do more random search and comparatively less evolutionary computing, so it creates a heavier load on the system by increasing the size of population for the next generation, hence more simulations have to be carried out and the improvement in time complexity would decrease.

Summary on GA Parameters

The results from test shows that these parameters are very influential, they have a big effect on the time complexity. To find the best set of parameter for the system is very difficult because effectiveness of parameters is problem orientated. Using an example in this system, if there are a lot of investment instruments in the portfolio, a large population and small number of generation will have a better time complexity because the system would simulate a lot of chromosome in one simulation phase rather them having a lot of simulation phase and simulate a small population at a time. These parameters in my system are not fixed and user can set the parameters according to their problem, but for recommendation, I would suggest a medium population size (300), a high number of generations (600), mating pool size – 50% of population size (150), random chromosomes addition size – 50 % of population size and a mutation rate of 1% of population size. I would also suggest running the system a number of times (3-5), this will ensure the solution found by the system is very close to the best solution. Even though the mutation rate I suggested is small comparing to the size of adding random chromosomes (Macro-Mutation operator), but it is essential for mutation to be carried out because it provides mutations in the genes of the chromosomes with good performance whereas the macro mutation operator provides mutations to the population as a whole.

4.2.2. Investment Simulator

4.2.2.1. Investment Instruments

Investment instruments are implemented as user defined data types in Java, the main function for this class is to extract common functions and variables to hold object orientated structure of the system. Different types of instruments such as fixed income instruments inherit methods from this class. In the instrument class, it contains standard variables such as the name of the instrument, price and type, it also provides standard methods such as displaying instruments and calculating value of the instrument to its sub class. The diagram below shows the tree of instruments and I will further explain each concrete implementation of instruments:

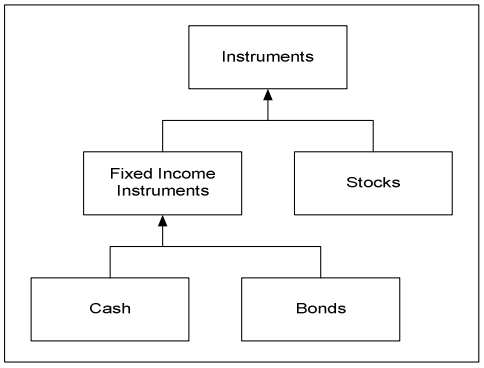


Figure 20: Tree of Instruments

Fixed Income Instruments

This is another higher level class, it extracts all common variable and methods from all instruments that have fixed income features in them. All the fixed income instruments must extend this class before can be use in the system and its methods can be overridden by concrete classes. This class has fixed income methods such as counting down to interest delivery date.

Stock

This class is the implementation of stocks, it extends the class instrument and inherits all the methods from instrument. It is use to trade in the system and price of the stock can be update in the system.

Cash

This class is the implementation of cash and it is an extension of the fixed income instrument because it has some fixed income behaviors. Cash in the system is assumed that all cash is placed in the bank and interest is given to user by the bank at a regular time period, so this is very similar to the idea of fixed income, hence it extends the fixed income instrument. The cash class overrides some of methods in instruments such as the show() method, because when displaying the cash, the information displayed is very different from other instruments.

Bonds
This class is the implementation of bonds, it is an extension of fixed income instrument and it overrides some of the methods in fixed income instrument such as showing the instrument. It also has some special methods, for example, converting the bond object into a byte array. This method is used to serialize bond objects and in the system it is used to make a copy of the object for passing it to function by value.

4.2.2.2. Portfolio
Portfolio is implemented using an Arraylist. Arraylist is a Java data type, it is an array with basic array function already implemented. The Arraylist is use to hold the instrument objects. The instruments in the portfolio arraylist can be updated using methods in portfolio control. The portfolio also has the variable *marketIndex*, it is use to store the current market index for the stock market. This variable is use in some of the investment strategies. The portfolio object can be serialized because each simulation requires a deep copy[12] of portfolio. The portfolio is managed by a portfolio controller, the controller provides trading methods to buy and sell instruments.

4.2.2.3. Investment Strategy
Investment strategies are implemented in an inheritance tree like the way I implemented instrument. Investment strategies are separated into different categories, for the system I implemented, there are two categories: Equity investment strategies and Bond investment strategies. They both have a common interface, the structure is shown in the diagram below:

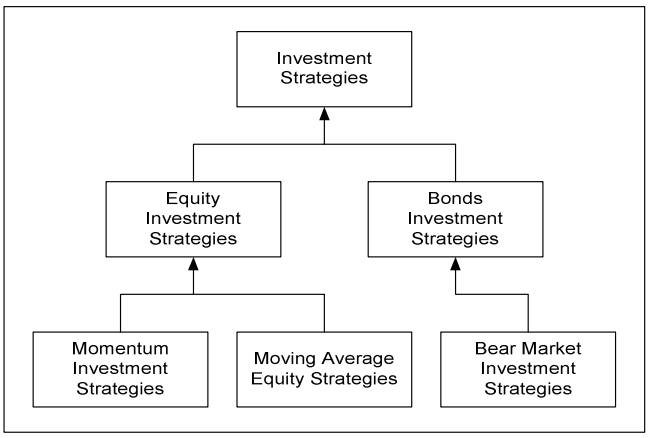


Figure 21: strategies structure

Interface Investment Strategies
This is the top interface of the strategy structure and this is a logical interface. This interface is an empty class, it doesn't have any methods or variables. The main purpose of this class is to provide a common interface for different type of strategies, this would make adding of new strategies and new category of strategies much easier and it also provide an easy method for the hedge fund controller to access the strategies.

Interface Equity investment strategies

[12] Deep copy in Java is a method to copy an object by value and not just copying a reference to the object.

This is an interface for all the equity investment strategies and it implements[13] investment strategies. It has a method called makeDecision(), this method must be overridden by a concrete class. Any new equity strategies must implements this interface and they must have the method makeDecision() in their class body.

Interface Bond investment strategies
This is an interface for all the bonds investment strategies and it implements investment strategies. It has a method called makeDecision() like the interface Equity investment strategies but this method takes in a different set of parameter, this method must be overridden by a concrete class. Any new bonds strategies must implements this interface and they must have the method makeDecision() in their class body.

Momentum Investment Strategy
This class implements the interface equity investment strategies. This strategy compares today's equity price with yesterday price, if the price has increased, a 'BUY' decision is created with the buy amount set as a attribute in the object and vice versa. The 'HOLD' decision is only used when the there is no change in the equity price. The trading amount is calculated by multiplying the buy or sell ratio depending on the trade decision from the chromosome with the percentage changed in equity price and the amount of equity in the portfolio.

Moving Average Equity Strategy
This class implements the interface equity investment strategies. This strategy calculates an average price of previous data, the number of previous data can be set by users. The strategy compare today's price data with the average, and then if the difference is positive, a 'BUY' decision is created with the buy amount set as a attribute in the object and vice versa. The 'HOLD' decision is only used when the there is no change in the equity price. The trading amount is calculated using the same method in momentum investment strategy.

Bear Market Investment Strategy
This class implements the interface bond investment strategies. This strategy monitors the market index value in the portfolio class object. The market index is then compared to the current market index and the percentage different is calculated. If the percentage different is negative, this shows the overall market is not performing well and the strategy will compare the percentage difference with its threshold that is set previously by the GA. If the percentage different is over the threshold, it will create a 'BUY' decision and the portfolio controller will buy bonds according to the attributes in the decision object.

4.2.2.4. History
Information on the portfolio is store in the History class. The History class has an array, each index of the array represents one time unit. This class also has methods to calculate statistical measurements such as mean, standard deviation and Sharpe ratio.

[13] Implements – a technical term used in Java programming language, it mean the class is a sub class of an interface. For example, X implements Y – that implies X has an interface Y.

4.3. Testing

Testing is done on the system to ensure the correctness of the functions in the system, each class is tested individually, the output of the functions are monitored to ensure the functions are operating as they should be. Testing is done very differently on GA and IS, testing on GA is done when the module have been implemented fully whereas IS is developed using a test-first approach. It is because the GA is a much smaller module, it only has 7 classes, bugs and error can be found and corrected easily. But the IS module consist of 19 classes, to find bugs and runtime error is very difficult, so a test-first approach was used. Below shows the important functions that testing is essential.

Genetic Algorithm

1. Initialization initializes the correct number of chromosomes in the population.
2. After the simulation, all chromosomes have a fitness value.
3. The population return the right number of chromosomes
4. when accessing a chromosome in the population, the right chromosome is return
5. Roulette wheel selector returns the right number of chromosomes.
6. Recombination of chromosomes is correct, the point of crossover is correct.
7. Random chromosomes addition adds the right number of chromosomes into the population.

Investment Simulator

1. The instruments are correctly initiated.
2. Instruments are added to the portfolio correctly.
3. Market Index can be set correctly in the portfolio.
4. The portfolio control can correctly perform trading functions in the portfolio.
5. Portfolio can correctly update instrument in the portfolio.
6. The portfolio can correctly de-serialize portfolio and bonds byte array
7. Decisions can be carried out correctly on the portfolio.
8. Investment strategies can create correct decisions.
9. Investment strategies are correctly initiated.
10. Bonds are correctly added to BondList and BondList correctly returns the bond.
11. Hedge Fund Controller is correctly initiated.
12. Hedge Fund Controller can correctly select strategies using the integers in the chromosome.
13. Hedge Fund Controller can correctly perform the simulation using imported data.
14. Portfolio return can be correctly added to history.
15. History class can correctly calculate the mean, standard deviation and Sharpe Ratio.
16. The correct fitness is return by the Hedge Fund Controller.

The tests are carried out using Unit Testing, the results of tests can be found by running JUnit testing in Eclipse, please look at the user manual in Appendix F for the method of running the tests.

4.3.1. Performance Testing

4.3.1.1. Performance test - different number of instrument

The system is tested using the configuration I suggested in section 4.2.1.2 and they uses the same set of data and same number investment instruments. The graph below shows the best solution's fitness value after each generation:

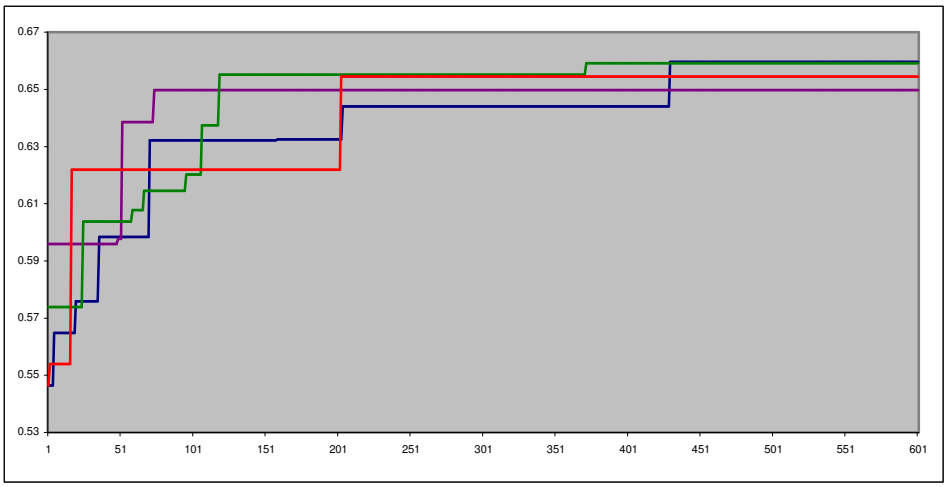


Figure 22: Four runs of the system with the same configuration

X-axis – Generation of the system
Y-axis – The fitness value of the best solution

The system have completed each run in within 720 seconds (12 minutes) and all four runs' best solutions tends to 0.65. This shows the best solution for this sets of data has a fitness value that is very close to 0.65. The investment simulator in this test has 5 investment instruments and I used weekly equity price for data.

Another test is done using an investment simulator that has 6 instruments and same GA configuration. The system has completed the process in 780 seconds. This shows increase the number of instrument by one will increase the time complexity by 8.3%, hence it shows the system is capable of completing a simulation with 10-20 instruments in an acceptable time and usually a hedge fund portfolio only have 8-14 instruments.

4.3.1.2. Performance test on different data size

The previous test was done using weekly equity prices and here I am going to test the performance of the system using daily prices. The system have complete the process in 2117 seconds (35 minutes) which has increase the time complexity by 270%.

4.3.1.3. Summary of Performance test

The system is capable to complete a simulation with 10-20 instruments in acceptable time but if the data size increases it will cause a heavy increase in time complexity of which is very difficult to run the system in standard home computers. So the experiments on the system in the next chapter will be carried out using weekly data.

Chapter 5

5. Experiments

This chapter will describe the experiments I did with the system. Firstly I will refine the research questions from chapter 1 to a more detailed hypothesis. Following that I will describe the source of real world stock market data. After that I will describe the experiments I did using the system and then finally I will presenting the results graphically and stating the findings in the experiments. All of the equities price graph will be shown in Appendix C

5.1. Hypothesis

The research questions in chapter1 I suggest are refined to detailed hypotheses and these hypotheses are mainly investigating how robust and adaptive is the system in different environments. These are the 2 main hypotheses I am going to test the system:

1. Training the system using the data of a predefined set of trading instruments in a specific time period in the history, can the chromosome it produced have the same level of performance in terms of portfolio return using the same set of trading instruments in today's market? And can the resulting chromosome adapt to the big changes in the stock market? This hypothesis will test how adaptive is the chromosome produced by the system in different time period of the stock market.

2. Training the system using data from a sector of the market, can chromosome it produced have the same level of performance in the same sector of the market but different stock in the same period of time? And in the same sector of the market but in a different country?

5.2. Source of real world data

Real world stock market data are obtained from various sources. The major source of equity and bonds price data are from the Reuter 3000Xtra online real time stock market monitor system, it provides all the information about different stock markets in different countries, it allows me to gather equities' prices, bonds' information and market indexes in different time periods. It has provided important information for all the experiments on the system. Another source of data is the bank of England website[14], it provides historical base rates and average bank deposit interest rate for the system to estimate the risk free return.

[14] http://www.bankofengland.co.uk/

5.3. Experiments to Test the Hypotheses

In this section, I am going to test the hypotheses by designing experiments for each hypothesis. This will involve selecting equities and bonds to be used in the experiments, selecting the time periods, selecting market indicators and multiple runs of the experiments.

5.3.1. Hypothesis One

5.3.1.1. Experiment Plan

I am going to select a few equities and select a time period as the training data. The time period length will be 1 year and then the chromosome produced by the system will be used to simulate trading for the following year, the previous year and a year that is further back in history.

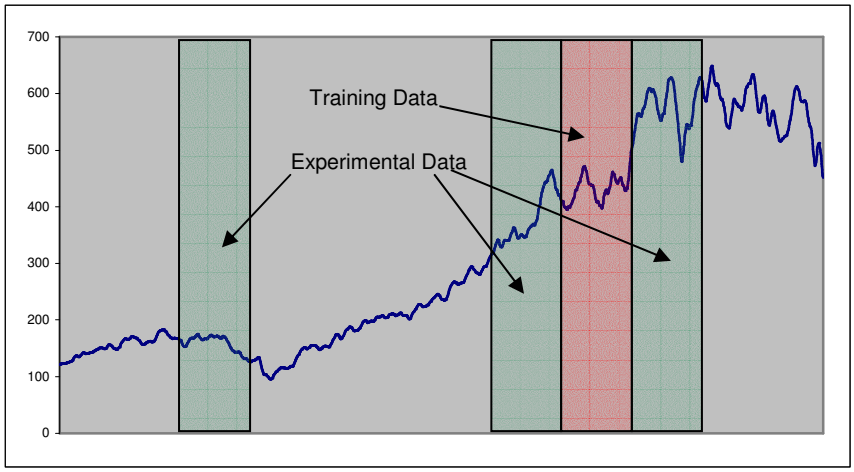


Figure 23: Graph showing the training and experiment data

I chose these time periods because investors usually apply technical analysis in the price data a short period of time before they invest in the market and the trend of equity price generally follow on for another period of time, this is the reason why I chose the year after the training period as experimental data. The other two period of time is chosen because it will demonstrate how adaptive is the chromosomes on historical data.

Methodology

The experiment will start by performing multiple runs of the system on the training data to produce a chromosome ($\mathbf{C_{training}}$) which has a fitness value ($\mathbf{F_{training}}$) that is very close the global optimum for this training environment (training data). The $\mathbf{C_{training}}$ will then be applied to obtain a fitness value ($\mathbf{F_{test}}$) for each experimental data (data from different time period) using the system with the GA module disabled.

For each experimental data, a full system run will be carried out to find the chromosome ($\mathbf{C_{best}}$) which has the fitness value ($\mathbf{F_{best}}$) that is close to the global optimum for this environment. The $\mathbf{C_{best}}$ and $\mathbf{F_{best}}$ of each experimental data are

compared with $\mathbf{C_{training}}$ and $\mathbf{F_{test}}$. The diagram below will give a clearer description of the experiment process:

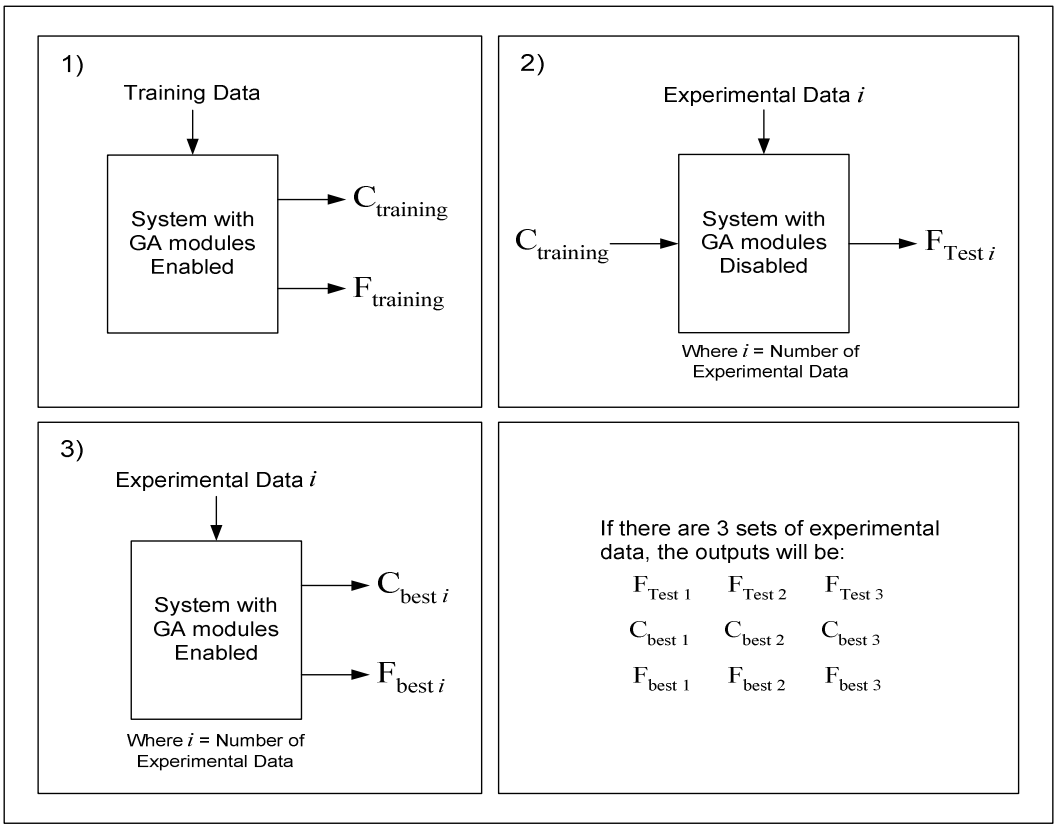


Figure 24: experiment process

Variables name	Description
$\mathbf{C_{training}}$	Chromosome found by the system with the GA module enabled using the training data.
$\mathbf{F_{training}}$	The fitness of $C_{training}$ when running the system using the training data.
$\mathbf{F_{test}}$	The fitness of $C_{training}$ when running the system using the experimental data.
$\mathbf{C_{best}}$	Chromosome found by the system with the GA module enabled using the experimental data.
$\mathbf{F_{best}}$	The fitness of C_{best} when running the system using the experimental data.

Selecting Equity

The equities I am going to choose will be the FSTE 100 index constituents, I select them because hedge fund managers usually select those equities for their portfolio due to the lower risk they have. I will select 3 equities for this experiment because the system has an acceptable time complexity. The following equities[15] are chosen for this experiment:

1. HSBC 2.Tesco 3.Hilton Group

[15] The price graphs for these equities are shown in appendix C.

Selecting Bonds

Selecting bonds is difficult in different time period, because bonds are issued in one time period, may not still be existed in another time period. For this experiment I will try selecting bonds that existed in all four time periods or selecting bonds that have similar interest rates and the coupon payments frequency is similar. I am going to choose 3 bonds for this experiment. The following Bonds are chosen for this experiment:

Issuer	**Industrial Class**	**Rating**	**Interest Rate**	**Coupon Payment Frequency**	**Issue Data**	**Mature Date**
GILT	Government	AAA	8.5%	Semi-Annual	16 -07-86	16 -07-07
Canary Wharf Finance PLC	Finance Service	AAA	7.23%	Quarterly	04-12-97	22-10-27
EIB	Supranational Organisation	AAA	8.75%	Annual	14-02-95	25-08-17

I chose these bonds because they have different interest rate and different coupon payments frequency, so that the system has a range of different bonds to select from. Even though bond with rating AAA have lower interest rate, but I still select bonds with that rating, because in the real world, hedge fund manager usually select bonds with rating AAA to reduce the risk of losing portfolio value.

Other settings for the experiment

Time periods	Currency	Initial Cash Amount in pounds	Bank's average Interest Rate	Frequency of banks paying interest	Data Type
2002-2003	GBP	100k	4%	4	Training
2001-2002	GBP	100k	5.2%	4	Experiment
2003-2004	GBP	100k	3.6%	4	Experiment
1997-1998	GBP	100k	6.8%	4	Experiment

Assumption made in the experiment

It is assumed that bonds' interest rate does not change throughout the year and bonds are not tradable in the system. It is also assumed that bank's interest rate is fixed throughout the year and an average interest rate is used to calculate the risk free return.

5.3.1.2. Training

The table below show the chromosome produced by the system in three separate run:

	Chromosome ($C_{training}$)	**Fitness** ($F_{training}$)
Run 1	64 7 43 0 36 18	0.7030565140792469
Run 2	86 9 36 0 40 17	0.7021727514995172
Run 3	81 7 44 0 60 17	0.7023268977713589

The system has suggested very similar results in all three runs, this shows that the chromosomes suggested by the system is the best chromosome or it is very close to the global optimum. By looking at the chromosome, the moving average equity strategy is more preferred in this year, a small buy ratio is used and a relatively higher sell ratio is used. This suggest that the overall movement of the 3 equities is downward, even though Tesco's and Hilton group's price have raised but the highest priced equity (HSBC) has an overall drop in price during the year. This has forced the system to produce a more conservative buy/sell ratio. The fourth gene in the chromosome has a value of 0 suggested that the bonds are very favorable, because the fourth gene is the percentage drop in the FTSE index before buying bonds. Two of the three runs suggested that investor should invest into Canary Wharf Finance PLC, because of the high frequency of coupons payments. The other run suggested that investor should select the GILT bond, because of it high interest rate. All three runs suggested investors should use 17%-18% of the cash into bonds and a low buy ratio this suggests that bonds in this time period a more dominating instrument than equities and it also shows that a high percentage of cash was not invested. In this environment, the estimate proportion of cash allocated to each instruments is Bonds: 20%, Equities: 10% and Cash: 70%. From this I also understand that equities have a very small contribution in the fitness, to prove that a fitness landscape is drawn with X-axis as the sell ratio, Y-axis as the buy ratio and Z-axis as the fitness.

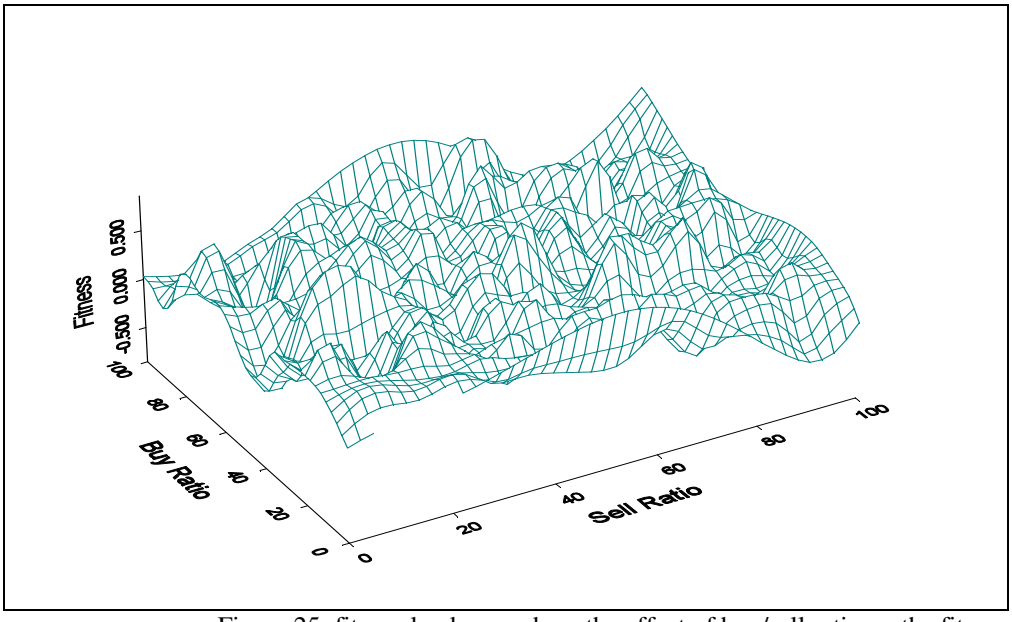


Figure 25: fitness landscape show the effect of buy/sell ratio on the fitness

This method of analysis the fitness landscape was suggested by T. Bäck's study in 1996 [2] and it is also used in T. Smith, P. Husbands, P. Layzell and M. O'Shea's study in 2002 [17]. The random peaks suggest that the buy/sell ratio has a very little effect on the fitness and the random peaks are produced by the genes responsible for bonds strategies. So a schema[16] is placed over the genes that are responsible for the equities investment

[16] Disregarding the some genes in the chromosome, because it doesn't or have a very small contribute to the fitness.

strategies and then another fitness landscape is draw with X-Axis as the percentage of cash in the portfolio to be use to trade bonds, Y-Axis as the percentage drop in index before buying bonds and the Z-Axis as the fitness of the chromosomes.

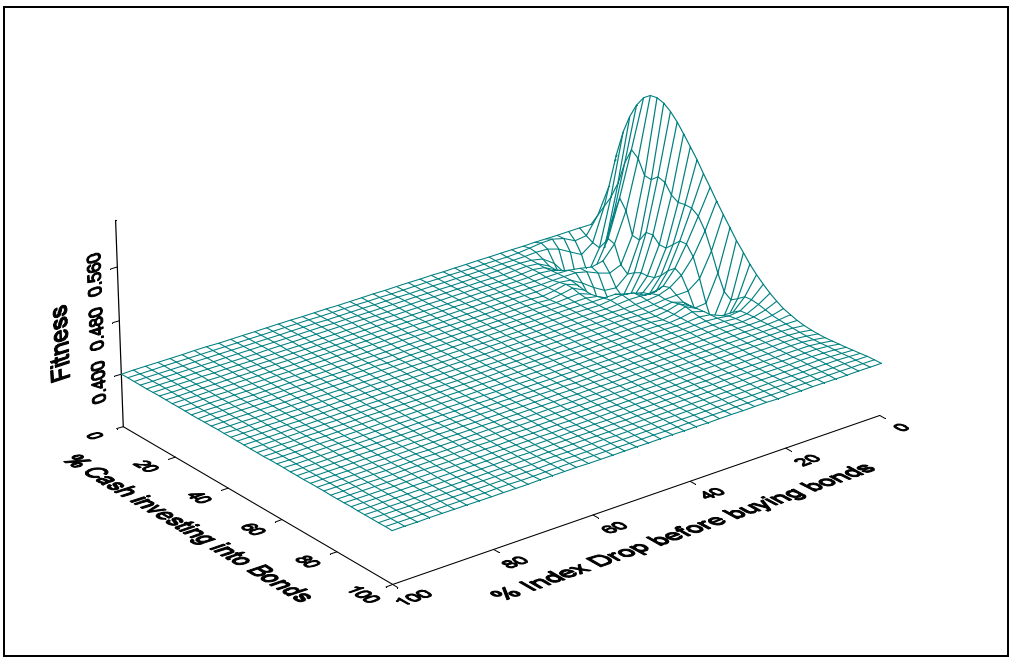


Figure 26: fitness landscape show the effect of drop percentage and percentage of cash use in trading bonds ratio on the fitness

This fitness landscape shows one high peak where percentage of cash to be use in trading bonds at 20 and the percentage drop in index before buying bonds is 0, The peak has reach to the fitness value of 0.6 which show bonds and cash is contributing ~85% (0.6/0.7) of the fitness value, and because of the low cash interest rate comparing to the bonds' interest rate in this training time period, bonds in this environment would have a higher return, hence it suggest that bonds have a higher contribution in the fitness than cash, so this tells us that the dominant instrument in this environment is bonds and chromosome $\mathbf{C_{training}}$ manages the portfolio using strategy that is bonds dominating. By drawing fitness landscapes, it provides an indication of what environment has the system been training on, which is the dominant instrument in this time period and an estimation of each instrument's contribution to the fitness. In this environment, an estimate percentage of each instrument's contribution would be bonds: 55%, cash: 30% and equities: 15%. The estimate percentage of bonds' and cash's contribution are 55% and 30% because bond's interest rate is about double the interest rate of cash, so the contribution of cash to the fitness will be about 1/3 of 85% (~30%) and the contribution of bonds to the fitness will be about 2/3 of 85% (~55%).

5.3.1.3. Results

Test 1
Applying the Chromosome ($C_{training}$) to time period 2001-2002 (Experimental Data 1).

$\mathbf{C_{training}}$	64	7	43	0	36	16
$\mathbf{F_{test\ 1}}$	= 0.74257					
$\mathbf{C_{best\ 1}}$	86	14	64	0	39	19
$\mathbf{F_{best\ 1}}$	= 0.74838					

Using $\mathbf{C_{training}}$ as an input for the system with the GA module disable (simulator only) on experimental data in the time period 2001-2002, I get the fitness of 0.74257 ($\mathbf{F_{test\ 1}}$) and when I use the system with GA module enabled on the experimental data, the best chromosome that the system found was 86 14 64 0 39 19 ($\mathbf{C_{best\ 1}}$) which has a fitness of 0.74838 ($\mathbf{F_{best\ 1}}$). Both $\mathbf{F_{test\ 1}}$ and $\mathbf{F_{best\ 1}}$ are very similar and the chromosomes ($\mathbf{C_{training}}$ and $\mathbf{C_{best\ 1}}$) which produced $\mathbf{F_{test\ 1}}$ and $\mathbf{F_{best\ 1}}$ are also very similar. In the fourth gene in the chromosomes ($\mathbf{C_{training}}$ and $\mathbf{C_{best\ 1}}$) has the value 0, this shows that bonds in this environment are more favorable and a high proportion of cash is allocated to invest using bonds and into bank's saving. The fitness value and the chromosomes suggest that this environment and the training environment are very similar hence the chromosome $\mathbf{C_{training}}$ adapt to the time period 2001-2002 very well.

Test 2
Applying the Chromosome ($C_{training}$) to time period 2003-2004 (Experimental Data 2).

$\mathbf{C_{training}}$	64	7	43	0	36	16
$\mathbf{F_{test\ 2}}$	= 0.47700					
$\mathbf{C_{best\ 2}}$	70	60	100	9	67	0
$\mathbf{F_{best\ 2}}$	= 0.67842					

Using $\mathbf{C_{training}}$ as an input for the system with the GA module disable (simulator only) on experimental data in the time period 2003-2004, I get a fitness of 0.47700 ($\mathbf{F_{test\ 2}}$) and when I use the system with GA module enabled on the experimental data, the best chromosome that the system found was 70 60 100 9 67 0 ($\mathbf{C_{best\ 2}}$), which has a fitness of 0.67842 ($\mathbf{F_{best\ 2}}$) and it is 29% better than the fitness I get using $\mathbf{C_{training}}$. By observing $\mathbf{C_{best\ 2}}$, the value 0 in the sixth gene suggested that bonds are very unfavorable, because the sixth gene represents the percentage of cash to be use in trading bonds. The high Buy/Sell ratio shows that equities in this time period are more favorable. The fitness landscape is drawn with X-Axis as the buy ratio, Y-Axis as the sell ratio and Z-Axis as the fitness.

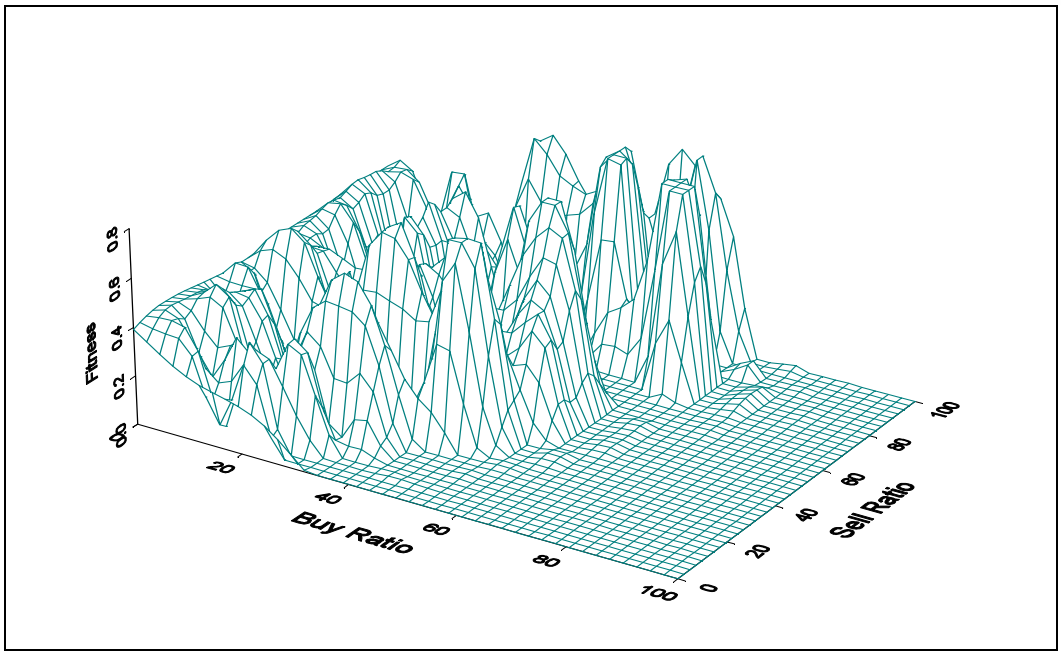


Figure 27: Fitness landscape show the effect of buy/sell ratio on the fitness for test 2

The landscape shows the peaks rise gradually as the buy ratio increase until buy ratio reaches around 60 and the peaks tend to be higher when the sell ratio is near 100. The highest peaks have the fitness value near 0.6. The large number of peaks suggest that there are more than one chromosome can achieve such fitness. By comparing this Buy/Sell ratio fitness landscape (figure 27) with the Buy/Sell ratio fitness landscape of the training environment (figure 25), I can compare and spot the differences between the two environments. The shapes of the two landscapes are very different, this shows that the two environments are very different and $\mathbf{C}_{\mathbf{training}}$ will give very different fitness value. In this test, the Buy/Sell ratio fitness landscape shows a trend on how the ratios affect the fitness, this indicates that the environment's dominant instrument is equities and this finding is reinforced by the configuration of the chromosome $\mathbf{C}_{\mathbf{best\ 2}}$ (value 0 in the sixth gene) and the high Buy/Sell Ratio. On the other hand the Buy/Sell ratio fitness landscape of the training environment (figure 25) does not show any trend and as explained before the dominant instrument in the training environment is bonds. This is the reason why $\mathbf{C}_{\mathbf{training}}$ has a lower performance in this environment.

Test 3
Applying the Chromosome ($\mathbf{C}_{\mathbf{training}}$) to time period 1997-1998 (Experimental Data 3).

$\mathbf{C}_{\mathbf{training}}$	64	7	43	0	36	16
$\mathbf{F}_{\mathbf{test\ 3}}$	= 0.40731					
$\mathbf{C}_{\mathbf{best\ 3}}$	8	6	5	72	33	0
$\mathbf{F}_{\mathbf{best\ 3}}$	= 0.70354					

Using $\mathbf{C}_{\mathbf{training}}$ as an input for the system with the GA module disable (simulator only) on experimental data in the time period 1997-1998, I get a fitness of 0.40731($\mathbf{F}_{\mathbf{test\ 3}}$) and when I use the system with GA module enabled on the experimental data, the best

chromosome that the system found was 8 6 5 72 33 0 ($\mathbf{C_{best\ 3}}$), which has a fitness of 0.70354 and it is 42% better than the fitness I get using $\mathbf{C_{training}}$. By observing $\mathbf{C_{best\ 3}}$, the value 0 in the sixth gene suggested that bonds are very unfavorable, which is very similar to the chromosome ($\mathbf{C_{best\ 2}}$) developed in test 2. To find the similarity, a fitness landscape of this time period is drawn and below shows the landscape with X-Axis as the buy ratio, Y-Axis as the sell ratio and the Z-Axis as the fitness.

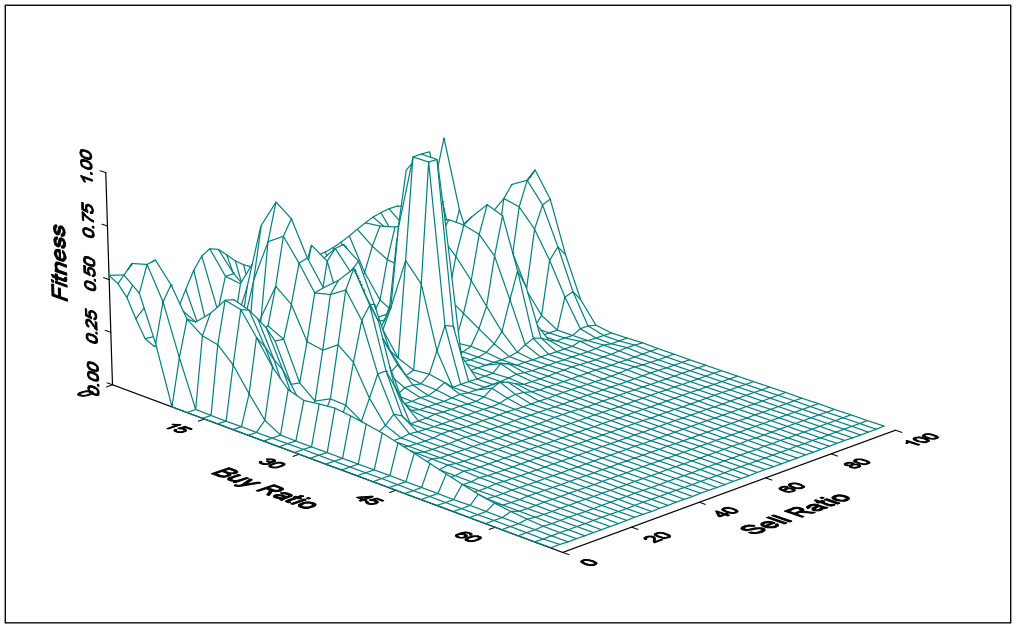


Figure 28: Fitness landscape show the effect of buy/sell ratio on the fitness for test 3

This landscape is very similar to the fitness landscape (figure 27) in test 2, this suggests $\mathbf{C_{best\ 3}}$ can be used in the time period 2003-2004 and still maintenance a reasonable performance, but the result of test shows that $\mathbf{C_{best\ 3}}$ cannot adapt to the time period 2003-2004, because when using this chromosome $\mathbf{C_{best\ 3}}$ in the system with the GA modules disabled and using 2003-2004 experiment data as the input, it only gives a fitness of 0.38308. Even though the landscapes are very similar, but the buy/sell ratio for both chromosomes ($\mathbf{C_{best\ 3}}$ and $\mathbf{C_{best\ 2}}$) are very different[17] and the main different is the bank's average interest rate[18]. It is $\mathbf{C_{best\ 3}}$ has very low buy ratio and sell ratio, it also have a value of 0 for percentage of cash to be use in trading bonds but it still manage have a fitness of 0.70354, so this suggests that the portfolio managed by $\mathbf{C_{best\ 3}}$ makes high portfolio return using bank's interest. This can be further proven by looking at the bank's average interest rate, the interest rate in this period of time was 6.8% , so when applying this to the experimental data in time period 2003-2004, the fitness is much lower because in 2003-2004, the bank's average interest rate is 3.6%, so there is 3.2% different in the portfolio return. This shows that the chromosome here ($\mathbf{C_{best\ 3}}$) is cash dominated. This is the reason why $\mathbf{C_{training}}$ yields a low fitness when applied to the system with GA modules disable and have 1997-1998 experimental data as the input.

[17] $\mathbf{C_{best\ 3}}$**:** Buy Ratio = 6 Sell Ratio = 5
$\mathbf{C_{best\ 2}}$**:** Buy Ratio = 60 Sell Ratio = 100

[18] Test 3 time period 2003-2004**:** Bank average interest rate = 6.8%
Test 2 time period 1997-1998**:** Bank average interest rate = 3.6%

5.3.1.4. Further experiment

The main purpose of this experiment is to find out how adaptive are the chromosomes that manages portfolio using strategy that is equities dominating in different equities dominant environments. The equities dominant chromosome I am going to use is the chromosome in test 2 ($\mathbf{C_{best\ 2}}$). In this experiment, I am going to choose a set of stocks that have a big raise in stock price over a year, this will ensure trading equities is the most profitable, hence setting an equities dominated environment. By applying $\mathbf{C_{best\ 2}}$ to this environment using the system with GA modules disabled (simulator only), I will obtain the fitness ($\mathbf{F_{test\ Extra}}$) and I can compare it with the best possible fitness[19] ($\mathbf{F_{best\ Extra}}$) of this environment to check the adaptability of equities dominating chromosomes.

Selecting stocks with big raise in equity price

The stocks I am going to select are Anglo American and Standard Charter, because they both have a raise in equity price of over 20% during 2003-2004, this will ensure that the test environment is more profitable when investing in equity because the highest fixed income return is 8.75%.

Selecting Bonds

I will use the bonds from the last tests, this will ensure the fairness of the experiment.

Overview of the experiment settings

Equities:	*Anglo American, Standard Charter*[20]
Bonds:	*GILT, Canary Wharf Finance PLC, EIB*
Time Period:	*2003-2004*
Currency:	*GBP*
Initial Portfolio Cash:	*100k pounds*
Bank's Average Interest Rate:	*3.6%*
Frequency of Interest Pay by bank:	*4*
Test Chromosome ($\mathbf{C_{best\ 2}}$):	*70 60 100 9 67 0*

Finding the Chromosome for this time period using the system with GA modules enabled

Chromosome ($\mathbf{C_{best\ extra}}$)	Fitness ($\mathbf{F_{best\ extra}}$)
91 81 94 30 31 0	0.8485984842952943
51 89 97 56 36 0	0.8409058260263506
59 82 86 77 11 0	0.8414211634761953

[19] Fitness which is very close to the global optimum of this environment

[20] The price graphs for these equities are shown in appendix C.

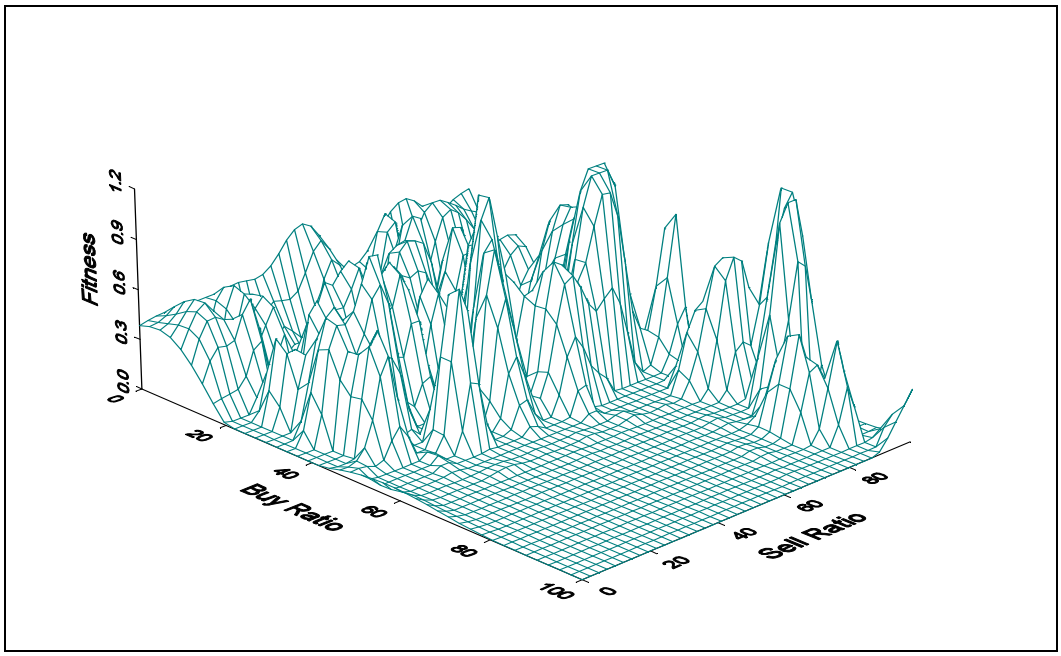


Figure 29: Fitness landscape showing the effect of buy/sell ratio on the fitness for further Experiment

The fitness landscape above shows how buy ratio and sell ratio affect the fitness where X-Axis as the buy ratio, Y-Axis as the sell ratio and the Z-Axis as the fitness. The fitness landscape is very similar to the fitness landscape in test 2 (figure 27), they both have higher peak as buy ratio increase and sell ratio tends to 100. Testing this environment using the Test Chromosome ($\mathbf{C}_{\mathbf{best\ 2}}$), I get a fitness of 0.79119 which is 6% below the best chromosome's fitness. The 6% drop in the fitness is considerably small and acceptable. This shows the Test Chromosome ($\mathbf{C}_{\mathbf{best\ 2}}$) can adapted to this environment. This result has reinforced the idea of chromosomes produced by the system in an environment (**env1**) can have a reasonably well performance in another environment where the dominant instrument is the same as the initial environment (**ev1**)'s dominant instrument.

5.3.1.5. Summary of findings in the experiments in Hypothesis One

From the experiment, I have found that environment of the stock market can be observed from the chromosomes produced by the system and how each gene affects the fitness value. Using these information, fitness landscape can be drawn to show a graphical representation of the environment. By comparing the fitness landscapes, I can qualitatively estimate the change in environment.

In the experiments, the results show that the system creates chromosomes according to the training environment. The combination of genes shows a bias in the amount of cash allocated to invest into different instruments. The instrument which contributes the highest proportion of the fitness value is the dominant instrument. To identify the dominant instrument, a schema can be place over the irrelevant genes in chromosome to remove the noise produced by other genes and the dominant instrument will be represented by a peak in a fitness landscape. The highest point of the peak will show the contribution of the instrument to the fitness value. In an environment where

only a very small amount of cash is allocated to bonds and equities, this will suggests that cash is the dominant instrument in the environment and cash deposited in bank's saving will produce higher return than investing into bonds and equities. In a portfolio where cash is split very evenly between instruments and the amount of cash for dominant instrument is only slightly higher than other instruments.

The adaptability of the chromosomes is shown in the experiments. In test 1, when applying the chromosome ($\mathbf{C_{training}}$) produced by the system which trained on the training data in the time period 2002-2003 to the experimental data in 2001-2002 using the system with the GA modules disabled, it has a fitness of 0.74257 ($\mathbf{F_{test\ 1}}$) which is only 0.7 percent lower than the best possible fitness ($\mathbf{F_{best\ 1}}$) found by the system with GA modules enabled, this shows the chromosome ($\mathbf{C_{training}}$) has adapted to the environment very well. But when applying this chromosome ($\mathbf{C_{training}}$) to the experimental data in 2003-2004, it has a fitness of 0.47700 ($\mathbf{F_{test\ 2}}$) and comparing this with the best possible fitness value ($\mathbf{F_{best\ 2}}$) for this time period, it shows that there is a 29% drop in performance. The reason for the chromosome ($\mathbf{C_{training}}$) produced using the training data can adapt to experimental data in 2001-2002 but not in 2003-2004 is that the training environment is very similar to the environment in 2001-2002 but very different to the environment in 2003-2004. The prices of equities in the training environment (2002-2003) was generally falling, so the fixed income instruments are more preferred, hence the chromosome ($\mathbf{C_{training}}$) is more bias towards bonds, so bonds is the dominant instrument in $\mathbf{C_{training}}$. In 2001-2002, the equities prices are very level, the prices' fluctuation is very small and the rise was not big enough to make profit when trading them. So the bonds were still the dominant instrument and $\mathbf{C_{training}}$ adapted to this environment very well. $\mathbf{C_{training}}$ did not adapted to the experimental data in 2003-2004 because the equities' price have a very big rise in that year and trading equities in that year can make a better profit than trading bonds, so the equities are more dominant in the portfolio, hence bonds dominating chromosomes have poor performance in this time period. $\mathbf{C_{training}}$ was then applied to the data in 1997-1998 using the system with GA modules disabled and comparing $\mathbf{F_{test\ 3}}$ and $\mathbf{F_{best\ 3}}$, there was a 42% difference in the two fitness, from the results, it shows that the environment of the time period 1997-1998 is cash dominated, a high portfolio return is obtained by the high interest from the bank. Applying the bonds dominant strategies in this time period will result in lower portfolio return hence a lower fitness.

The three tests in different time period shows chromosomes created by the system in a time period can adapt with a small sacrifice in the fitness value to other time period if the dominant instrument has not, this trend is shown in test 1, when applying the $\mathbf{C_{training}}$ to experimental data in 2001-2002, a percentage drop of 0.7% in the fitness value. This idea was reinforced by the extra experiment, an equities dominant environment is chosen for the test and a chromosome ($\mathbf{C_{best\ 2}}$) is produced by training the system using data in 2003-2004, the chromosome adapted to the equities dominant environment and there is only a 6% drop in the fitness value.

The experiments can conclude that the system's adaptation to different time period depends very much in the difference between the time periods' environments. The system can adapt better when the test time period is close to the training time period, because the environment will be similar when the time periods are close together, this is shown in the test 1 and the extra experiment. But the chromosome cannot adapt to the big

changes. For example in test 2, even though the time period is very close to the training time period but the environment of the stock market has changed significantly from prices being very level to sudden big raises. This shows the system is unable to adapt to big changes such as the dot com bubble burst in year 2000 and the high rise of interest rate in year 1993, because the magnitude of change in the environment is very high in these scenarios.

5.3.2. Hypothesis Two

5.3.2.1. Experiment Plan

I am going to select a time period which all test will done in this time period and then I am going to select a sector in the stock market. I will choose 3 equities from the sector and 3 bonds to train the system to create a chromosome ($\mathbf{C_{training\ 2}}$) that would give fitness that is very close to the global optimum. $\mathbf{C_{training\ 2}}$ will then be use to test other equities in the same sector in the same market. After that I will select stocks from the same sector but in a different market and testing it using $\mathbf{C_{training\ 2}}$ to check the adaptability of the chromosome in different markets.

Selecting a sector of the market

The sector I am going to test is the banks, because they are the most influential sector in the market.

Selecting equities

The equities I am going to use for training are the FTSE 100 index constituents. The following equities are going to be use to train the system[21].

1. Barclays 2.HBOS 3.Royal Bank of Scotland

Selecting Bonds

To keep the experiment fair, I will select 3 bonds and they will be used in every test

Issuer	Industrial Class	Rating	Interest Rate	Coupon Payment Frequency	Issue Data	Mature Date
LCRF	Finance Service	AA3	4.5%	Semi-Annual	18 -02-99	07 -12-25
Transco PLC	Utilities	A-	4.7%	Semi-Annual	14-12-99	14-12-22
EGG PLC	Banking	AA3	4.665%	Annual	19-11-02	22-08-07

[21] The price graphs for these equities are shown in appendix C.

Other settings for the experiment

Time periods	Currency	Initial Cash Amount in pounds	Bank's average Interest Rate	Weeks per interest pay by the bank
2002-2003	GBP	100k	4%	4

Assumption made in the experiment
It is assumed that bonds' interest rate does not change throughout the year and bonds are not tradable in the system.

5.3.2.2. Training
The table below show the chromosomes produced by the system in three separate run:

	Chromosomes ($C_{training\ 2}$)	**Fitness ($F_{training\ 2}$)**
Run 1	67 59 86 93 52 26	0.4194905193535031
Run 2	93 58 76 83 60 0	0.4161616398296217
Run 3	89 59 75 83 98 74	0.4169877404811152

All three runs of the system produced very similar chromosomes, all chromosomes have a very high value for the fourth gene. This gene represents the percentage drop in the FTSE 100 index before trading bonds and a very high value for this gene suggests that the bonds are not preferred in this environment. The chromosomes suggest that momentum investment should be used in this environment and it has a medium buy ratio and a high sell ratio. The shows that trading in equities is more favorable in this environment, hence equity is the dominant instrument in this environment.

5.3.2.3. Results
Test 4
This test will select 3 other equities from the same sector for testing the adaptability of the ($\mathbf{C}_{\mathbf{training\ 2}}$). The equities[22] I am going to select are:

1. HSBC(UK)
2. Alliance & Leicester PLC
3. Northern Rock

The bonds selected for this test is the same as the set of bonds in the training environment and other setting such as the banks' average interest rate and initial cash are also the same as the training environment. These conditions are all controlled, so it ensures the variable I am testing is different equities in the same sector. The environment is then tested for the chromosome which will give the fitness that is very close to global optimum using the system with GA modules enabled. The chromosome ($\mathbf{C}_{\mathbf{best\ 4}}$) that the system found was

[22] The price graphs for these equities are shown in appendix C.

82 95 100 58 0 0 and its fitness is 0.43487 **($F_{best\ 4}$)**. When I apply ($C_{training\ 2}$) to the system with GA disabled, I get a fitness value of 0.42272 **(F_{test4})**, which is only 2.7% lower than $F_{best\ 4}$. This shows the chromosome ($C_{training\ 2}$) has adapted to this environment very well and this environment is very similar to the training environment. This idea is further verified by selecting more equities from the sector for testing the chromosomes.

Test 5

The three equities I am going to select for this test are: EGG PLC, Bradford & Bingley and Standard Charter. All other conditions are same as the conditions in the training environment. Using $C_{training\ 2}$ in the system with the GA modules disabled, I yield a fitness of 0.39780 **($F_{test\ 5}$)** and then I used the system to find the best possible chromosome ($C_{best\ 5}$), the chromosome I found is 8 0 29 5 70 0 and it has a fitness of 0.402631 **($F_{test\ 5}$)**. The chromosome created using experimental data here is very different from the chromosome create in the training environment, the chromosome here ($C_{best\ 5}$) is clearly a cash dominant strategy because of the 0s in the second and the sixth gene, this shows the environment here can produce a similar portfolio return using a cash dominating chromosome or equities dominating chromosome. It also shows ($C_{training\ 2}$) can adapt to this environment with a decrease in the fitness value of 1.1%, hence demonstrating that the system can adapt to other equities in the same sector.

Test 6

In this test I am going to apply $C_{training\ 2}$ to a different market and the sector of the market that I am going to apply $C_{training\ 2}$ will be the same sector (banks). This will show the adaptability of the chromosomes in different market. I will select the Hong Kong exchange market as the test market for this test

Selecting Equities

The equities I am going to test are the Hang Seng Index constituents. The equities are as follow[23]:

1. HSBC (HK)
2. Bank of East Asia
3. Hang Seng Bank

Selecting Bonds

To keep the experiment fair, I selected 3 bonds that are very similar to the bonds selected in the training environment.

Issuer	Industrial Class	Rating	Interest Rate	Coupon Payment Frequency	Issue Data	Mature Date
CWLBKA	Banking	AA	4.95%	Annual	28 -11-01	28 -11-05
BCEE	Banking	A-	4.63%	Annual	23-11-01	23-11-06
CPSEC	Financial Service	A	3.21%	Quarterly	29-09-00	20-09-06

[23] The price graphs for these equities are shown in appendix C.

Other settings for the experiment

Time periods	Currency	Initial Cash Amount in Hong Kong Dollar	Bank's average Interest Rate	Weeks per interest pay by the bank
2002-2003	HKD	1380k	0.5%	4

Assumption made in this test
It is assumed that bonds' interest rate does not change throughout the year and bonds are not tradable in the system. The exchange rate for GBP to HKD is 13.8 throughout the year and the saving account interest rate of HSBC (HK) is used as the bank's average interest rate.

Applying $\mathbf{C}_{\mathbf{training\ 2}}$ to this environment using the system with GA modules disabled, surprisingly I get a fitness of 0.38882 **($\mathbf{F}_{\mathbf{test\ 6}}$)**. Using the system with GA modules enabled I found the chromosome ($\mathbf{C}_{\mathbf{best\ 6}}$) that give a fitness that is very close to the global optimum for this environment using the system. $\mathbf{C}_{\mathbf{best\ 5}}$ has the genes 97 32 100 43 52 0 and it has the fitness 0.39357 **($\mathbf{F}_{\mathbf{best\ 6}}$)**. It shows there is a 1.2% decrease in the fitness. The small decrease indicates that $\mathbf{C}_{\mathbf{training\ 2}}$ can adapt to this environment, hence show chromosomes created by the system using data from the bank sector of the market can be use in other markets, but this only shows the adaptability of the system in the bank sector because equities price of banks are closely related to the oil price and other economic indicators, so banks in the same period of time have very similar trend in their equities price, hence similar market environment and the chromosome created using data from the bank sector of one market can adapt to the bank sector of other markets. Other sectors of the market may not follow this trend, because companies in the same sector may not have a common economic indicator that their equities prices are closely related to it.

5.3.2.4. Summary of findings in the experiments in Hypothesis Two

The result of test 4 shows that chromosomes produced by the system using price data from the bank sector can be use in different equities in the same sector of the market with a small sacrifice in the fitness of 1.1%, it also suggests that the environments of different equities in the same sector are generally very similar, hence the chromosomes can adapt to it very well. This finding is reinforced by testing $\mathbf{C}_{\mathbf{training\ 2}}$ with more banks' equities in the London stock exchange which is shown in the result of test 5.

In test 6, $\mathbf{C}_{\mathbf{training\ 2}}$ is tested using equities from the bank sector of the Hong Kong stock exchange. The result shows $\mathbf{C}_{\mathbf{training\ 2}}$ can adapt to the environment with a sacrifice of 1.2% in fitness. From observations in the stock price data, it is shown that price movement of banks' equities in Hong Kong is very similar to the price movement of banks' equities in London stock exchange, hence shows the environments are very similar. The reasons behind is that banks' equity price are closely related to international economic indicators such as the international oil price. This allows chromosomes created using data from the bank sector of a market to be used in the bank sector of other markets.

Other sectors of the market may not follow the same trend, because their equities' price may not relate directly to international indicators. Further experiments have to be carried out in order to prove the adaptability of the system in different sector of different markets.

Chapter 6

6. Conclusion

This chapter contains a brief review of the project, firstly, a summary of what was done and will be given and then it will follow by a critical evaluation of the design of the system. After that I will give an analysis of the experimental results and state the contributions made in the project. Finally I will suggest the further developments in the project.

6.1. Summary

The main goal of this project is to design and implement a GA system for selecting and customizing investment strategies, the system is then use to evaluate the adaptability of chromosomes in different market scenarios.

The system implemented using Java programming language, the GA used in the system is customized to reduce the chance premature convergence of population (described in chapter 3). An investment simulator is connected to the GA for calculating fitness of chromosomes in the population. Tests were done to find the optimal GA parameters for the system to reduce the overall time complexity and to improve the accuracy of the solution found by the system. The system is then use to run experiments using real world data obtained from Reuters 3000Xtra Online stock market monitor system.

6.2. Evaluation

The system developed has achieved all the main technical objectives, but if I am going to re-design and implement the system, I am going to use the C programming language instead because this language will have a much better time complexity, so the searching for the best strategy will complete in much shorter time and it will allow the usage of daily price data instead of weekly price data. In the system, the representation of chromosome can be improved because the chromosome do not allow the system to trade multiple bonds and it will force the system to make decision for every equities in the portfolio, clearly this will reduce the robustness of the system. To make the system more robust, genetic programming (GP) could be use instead because the chromosomes it produces can include a section of code, hence it can have a higher adaptability and robustness. I think the system should model the stock market more precisely, stock market features such as dividends, trading of bonds and impose trading commissions should be added, because these features can affect the fitness of chromosomes and give a more precise Shape ratio that can be compare to real world portfolio's Sharpe ratio for benchmarking. I also think using just one technical indicator does not give a clear enough indication on the performance of portfolio, because from literature on finance and

investment they have shown a range of indicators reviewing different aspect of the portfolio to give the advantages and disadvantages in different scenarios.

All the experimental objectives have been achieved, each hypothesis is tested using the system and results are obtained for analysis. If the experiments are carried out again, I would select a wider range of price data and experiment on more sectors of the market to reinforce the hypothesis on adaptability of chromosomes in different sectors.

Overall, all the goals are met and I am happy with the outcome of the project. I believe this project would be a useful base for anyone to further extends their work in this area of GA.

6.3. Conclusion and Findings

My experimental results show that chromosomes created by the system trained using a set of price data and market information[24] has a dominant instrument. The dominant instrument is the most profitable investment instrument in the environment and higher proportion of cash will be allocated to invest using this investment instrument.

The results also show that the chromosome created by the system trained in a time period can adapt to other time periods if the dominant instrument of the environment is same as the dominant instrument of the training environment. It also suggests that chromosomes can adapt better if the time period is close to the training time period, because the trend of price movement is very similar. In addition the results show that a change in dominant instrument is an indication of a big change in the environment and the chromosome is not capable of adapting to big changes in environment.

The results of testing the adaptability of the chromosome on different equities in the same sector of market show that the adaptability is high. The chromosome can adapt with a small sacrifices of 1%-3%, because the equities in the same sector have a very similar price movement, hence the environment is very similar. Another test was carried out to analyze the adaptability of chromosome on the same sector but in a different market, the result shows that the chromosomes can adapt to it but the experimental result I obtain cannot convince this finding is absolutely correct. It is because the only sector I tested was the bank sector and this sector in different markets is related by the international economic indicators such as oil price. Further experiments on different sectors have to be carried out to make sure this finding is correct.

[24] The set of price data and other market information is the environment of the system.

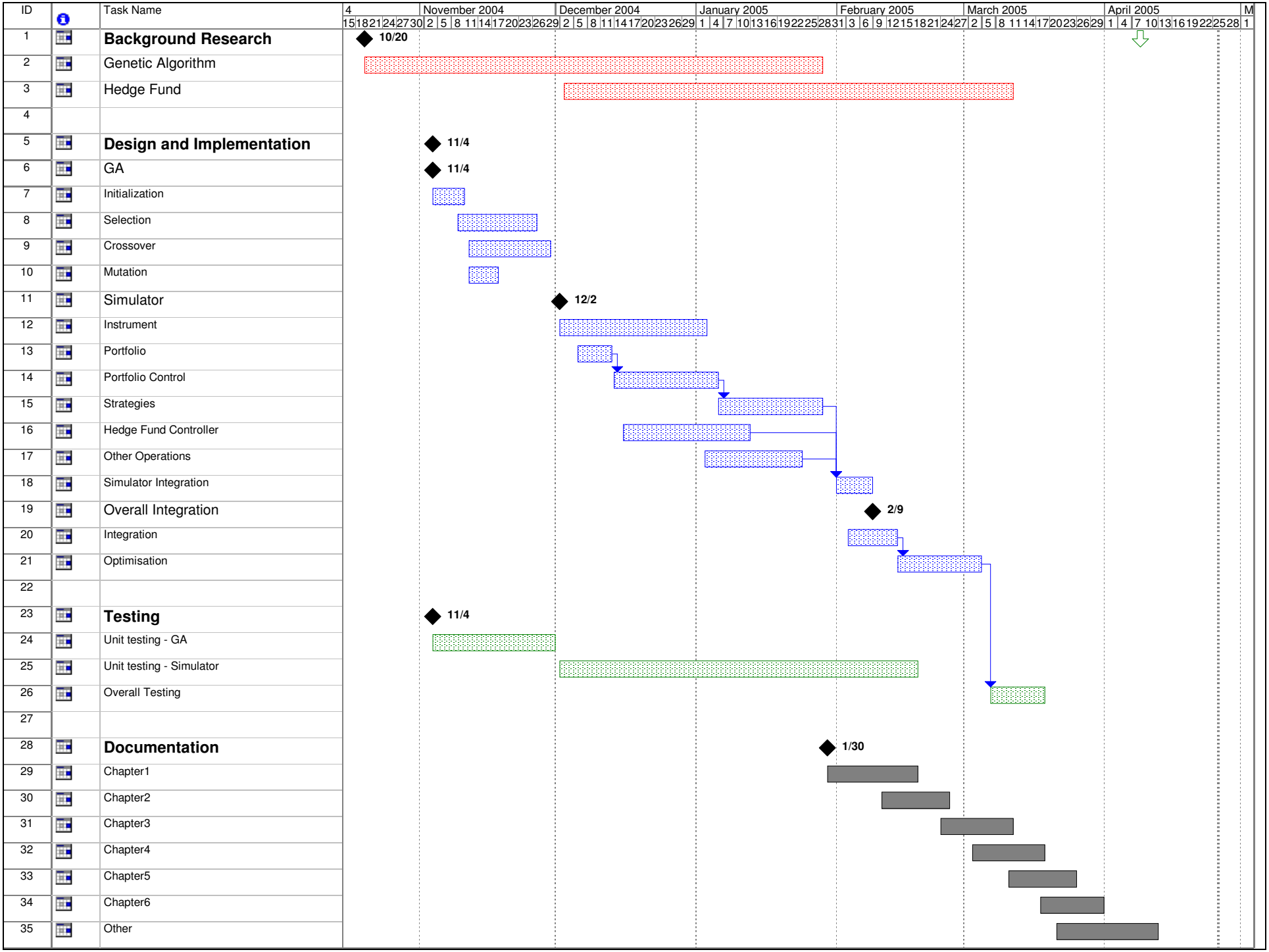
6.4. Further Work

There are a number of possible developments that this project can be enhanced and extended. Listed below are some areas that it would be very interesting to extend this project into:

- Implementing the system using Genetic Programming to improve the robustness and adaptability.
- Implement the derivatives into the simulator to model the stock market more precisely.
- Further experiments to prove the adaptability of chromosomes in different markets.
- Implementing more investment strategies to improve the performance of chromosomes.

Project management

Iterative approach was used in this project, initially a main goal was set and research was done to refine the main goal into milestones. Then deliverables were set based on the milestones, implementation and design were carried out side by side to achieve the deliverables. After each deliverables, requirements were reviewed to make sure the project was on course. Experiments were planned after first complete implementation of the system and then the design was reviewed and extra functions were added to make sure the experiments can be executed in the most efficient way. Tests were done after each deliverables to make sure the functions in the system is operating correctly. Equities price data were gathered from Reuters 3000Xtra online stock market monitor system. Experiments were carried out using these real world price data and results were analyzed.


ID
Task Name
November 2004
December 2004
January 2005
February 2005
March 2005
April 2005
1 Background Research
2 Genetic Algorithm
3 Hedge Fund
4
5 Design and Implementation
6 GA
7 Initialization
8 Selection
9 Crossover
10 Mutation
11 Simulator
12 Instrument
13 Portfolio
14 Portfolio Control
15 Strategies
16 Hedge Fund Controller
17 Other Operations
18 Simulator Integration
19 Overall Integration
20 Integration
21 Optimisation
22
23 Testing
24 Unit testing - GA
25 Unit testing - Simulator
26 Overall Testing
27
28 Documentation
29 Chapter1
30 Chapter2
31 Chapter3
32 Chapter4
33 Chapter5
34 Chapter6
35 Other
10/20
11/4
11/4
12/2
2/9
11/4
1/30

Appendix A: Parameters Testing Configuration 1

The table below shows the configuration of the GA for each test run.

Generation	Pop size	Elite size	mating pool	Random size	MutateRate	Time
250	150	6	75	30	0.3	89
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.497560302	0.486118082	0.49725482	0.509628953	0.496682004	0.497448832	

Generation	Pop size	Elite size	mating pool	0.4974488	MutateRate	Time
250	100	4	50	20	0.2	68
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.511806609	0.493787904	0.491401763	0.487533867	0.480471318	0.493000292	

Generation	Pop size	Elite size	mating pool	Random size	MutateRate	Time
250	80	3	40	16	0.16	50
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.484106695	0.496012621	0.475988716	0.49725482	0.503450657	0.491362702	

Generation	Pop size	Elite size	mating pool	Random size	MutateRate	Time
250	70	2	35	14	0.14	43
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.477544482	0.490567388	0.479279958	0.472468003	0.490014878	0.481974942	

Generation	Pop size	Elite size	mating pool	Random size	MutateRate	Time
250	60	2	30	12	0.12	37
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.471369079	0.468727397	0.49394009	0.49860461	0.496027677	0.485733771	

Generation	Pop size	Elite size	mating pool	Random size	MutateRate	Time
250	120	5	60	24	0.24	72
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.495706648	0.49819693	0.491668122	0.546776869	0.504697845	0.507409283	

Generation	Pop size	Elite size	mating pool	Random size	MutateRate	Time
250	130	5	65	26	0.26	78
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.495428023	0.500042448	0.501455171	0.520135022	0.502044115	0.503820956	

Generation	Pop size	Elite size	mating pool	Random size	MutateRate	Time
250	160	6	80	32	0.32	96
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.503450657	0.529908711	0.495228399	0.542897101	0.501366307	0.51	

Generation	Pop size	Elite size	mating pool	Random size	MutateRate	Time
250	170	7	85	34	0.34	101
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.511806609	0.539002339	0.504046217	0.545925043	0.503759975	0.520908037	

Appendix B: Parameters Testing Configuration 2

The table below shows the configuration of the GA for each test run.

10%

Generation	Pop size	Elite size	mating pool	Random size	MutateRate	Time
250	100	4	50	10	0.2	57
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5104586	0.4929646	0.48052863	0.49064647	0.46635381	0.48819042	
Generation	**Pop size**	**Elite size**	**mating pool**	**Random size**	**MutateRate**	**Time**
250	120	5	60	12	0.24	66
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5064295	0.4937359	0.48976662	0.50376068	0.49270089	0.49727871	
Generation	**Pop size**	**Elite size**	**mating pool**	**Random size**	**MutateRate**	**Time**
250	170	7	85	17	0.34	98
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5116479	0.5101905	0.49876495	0.49491679	0.49433069	0.50197017	

40%

Generation	Pop size	Elite size	mating pool	Random size	MutateRate	Time
250	100	4	50	40	0.2	70
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5047254	0.507912	0.4810356	0.48591017	0.51592723	0.49910207	
Generation	**Pop size**	**Elite size**	**mating pool**	**Random size**	**MutateRate**	**Time**
250	120	5	60	48	0.24	85
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.4923618	0.5048268	0.49063215	0.54677687	0.50432394	0.5077843	
Generation	**Pop size**	**Elite size**	**mating pool**	**Random size**	**MutateRate**	**Time**
250	170	7	85	68	0.34	118
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5264305	0.517912	0.50457117	0.52438856	0.49545383	0.51375122	

60%

Generation	Pop size	Elite size	mating pool	Random size	MutateRate	Time
250	100	4	50	60	0.2	80
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5052284	0.5104586	0.5270085	0.51850929	0.48850425	0.5099418	
Generation	**Pop size**	**Elite size**	**mating pool**	**Random size**	**MutateRate**	**Time**
250	120	5	60	72	0.24	96
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5216033	0.5077797	0.52990871	0.49295633	0.50295633	0.51104087	
Generation	**Pop size**	**Elite size**	**mating pool**	**Random size**	**MutateRate**	**Time**
250	170	7	85	102	0.34	137
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5101905	0.5374423	0.53550535	0.52643054	0.54643054	0.53119983	

70%

Generation	Pop size	Elite size	mating pool	Random size	MutateRate	Time
250	100	4	50	70	0.2	83
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5152284	0.5104586	0.5270085	0.51850929	0.49845625	0.5139322	
Generation	**Pop size**	**Elite size**	**mating pool**	**Random size**	**MutateRate**	**Time**
250	120	5	60	84	0.24	100
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5216033	0.5197797	0.52990871	0.50995633	0.52595633	0.52144087	
Generation	**Pop size**	**Elite size**	**mating pool**	**Random size**	**MutateRate**	**Time**
250	170	7	85	119	0.34	142
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5121905	0.5374423	0.53755054	0.52843054	0.54843054	0.53280887	

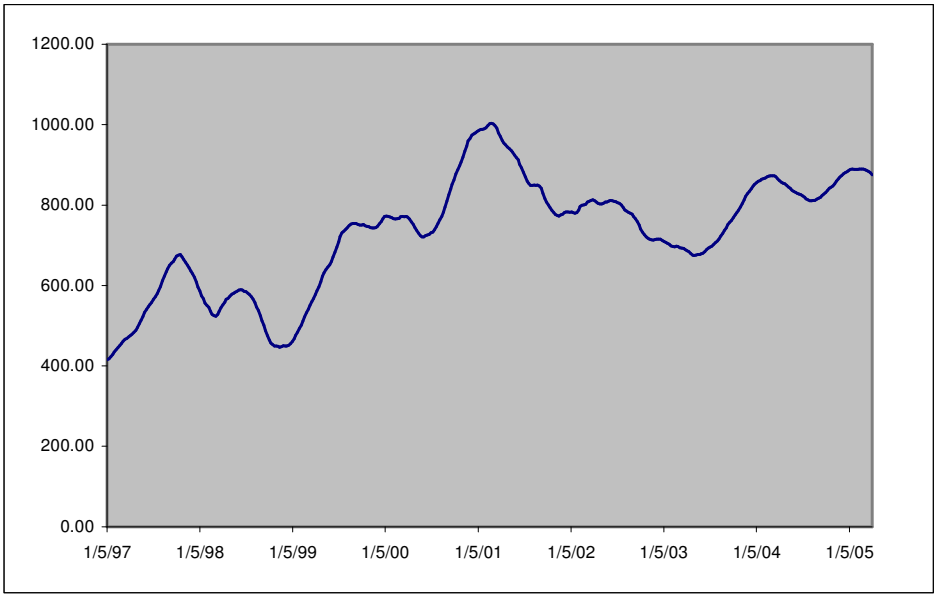
80%

Generation	Pop size	Elite size	mating pool	Random size	MutateRate	Time
250	100	4	50	80	0.2	90
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5172246	0.5174586	0.52706706	0.51850924	0.50945626	0.51794315	
Generation	**Pop size**	**Elite size**	**mating pool**	**Random size**	**MutateRate**	**Time**
250	120	5	60	96	0.24	110
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5279003	0.5207796	0.52990871	0.51134524	0.52678333	0.52334343	
Generation	**Pop size**	**Elite size**	**mating pool**	**Random size**	**MutateRate**	**Time**
250	170	7	85	136	0.34	153
Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**	
0.5169665	0.5375798	0.53755054	0.5386864	0.54956385	0.53606941	

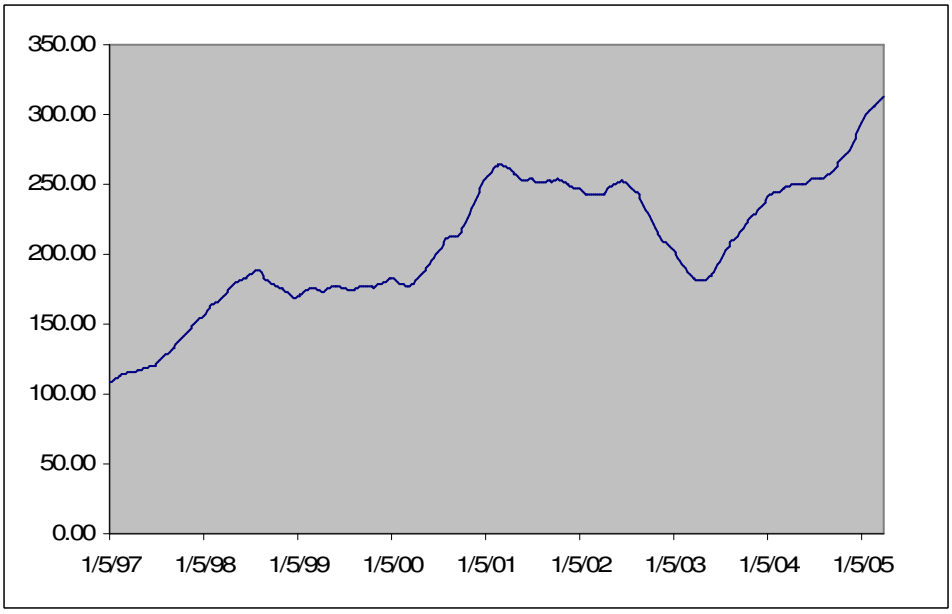
Appendix C: Equities Price Data

Equities price data of Hypothesis 1
Test 1, 2, 3

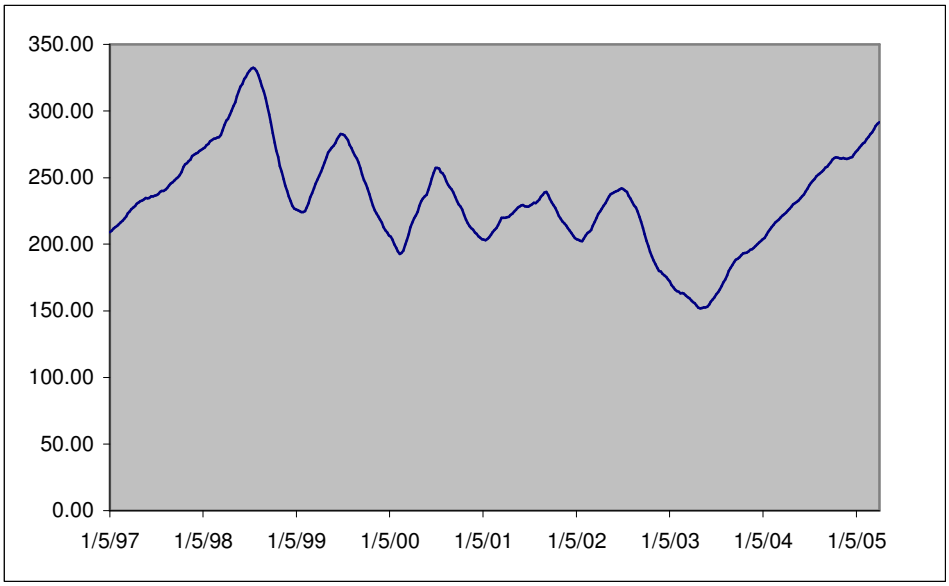
HSBC 1997 - 2005



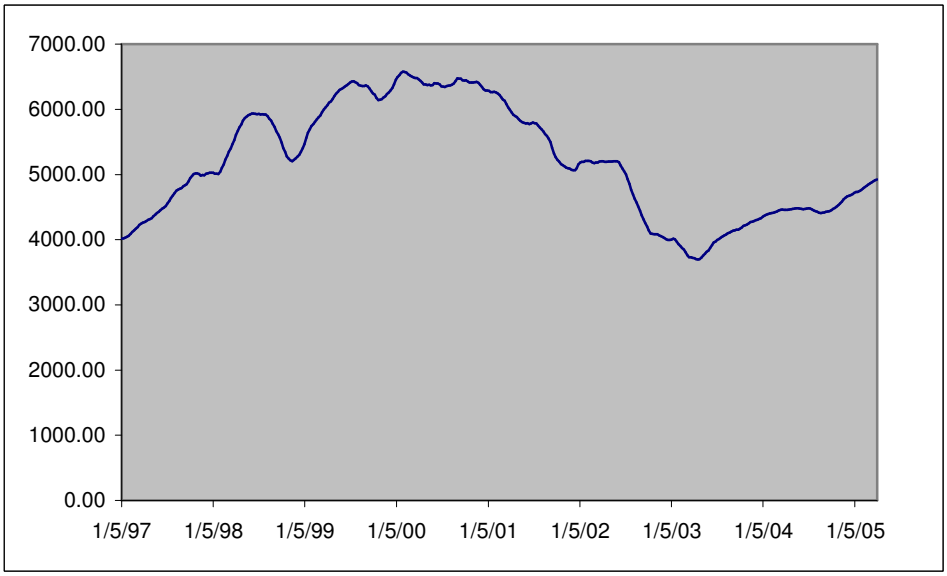
Tesco 1997 - 2005



Hilton Group 1997 - 2005

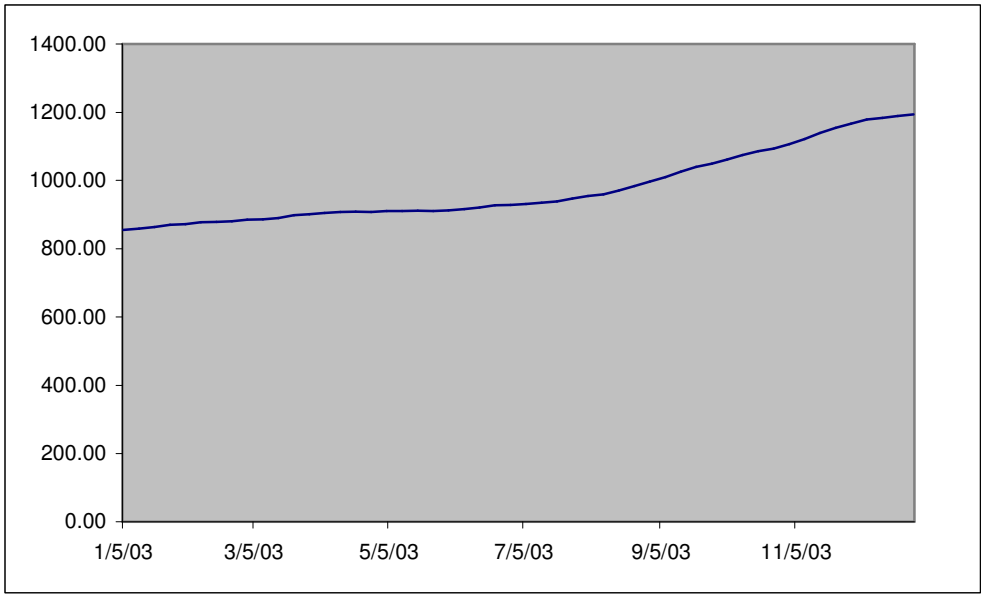


FTSE Index 1997 - 2005

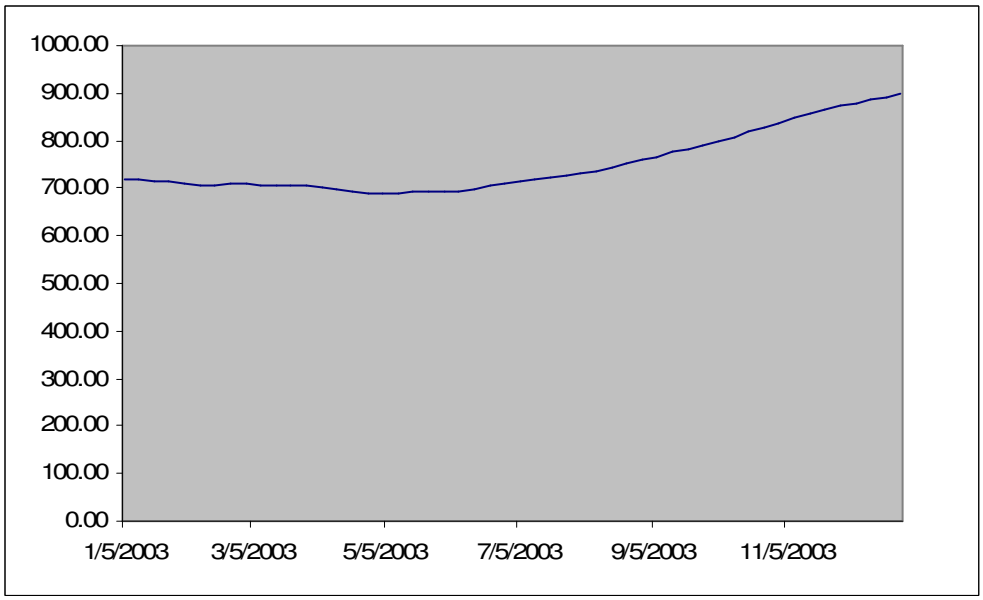


Extra Test

Anglo American 2003-2004

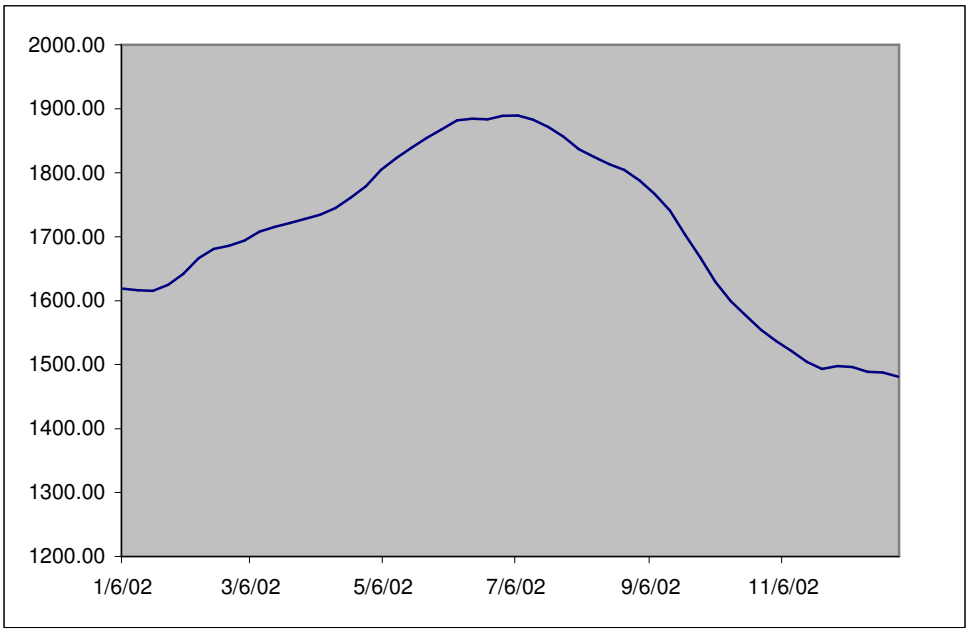


Standard Charter 2003-2004

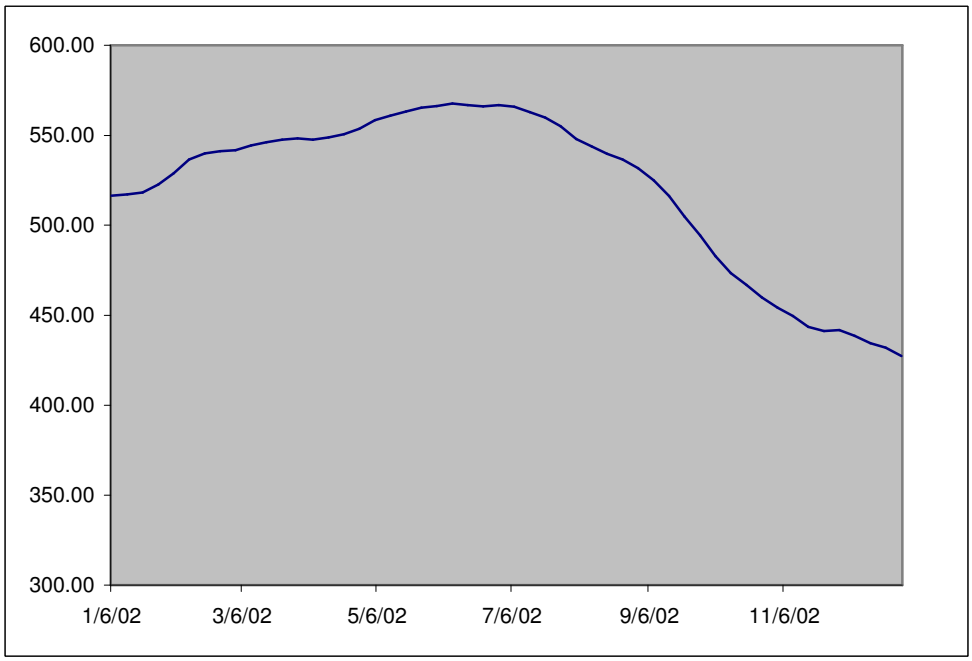


Equities price data of Hypothesis 2
Test 4, 5, 6

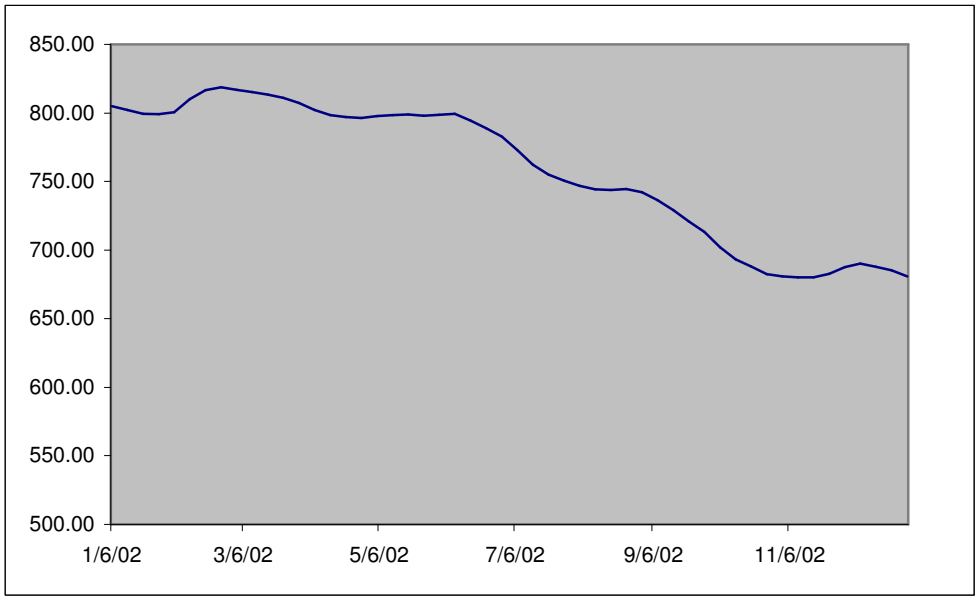
Royal Bank of Scotland 2002-2003



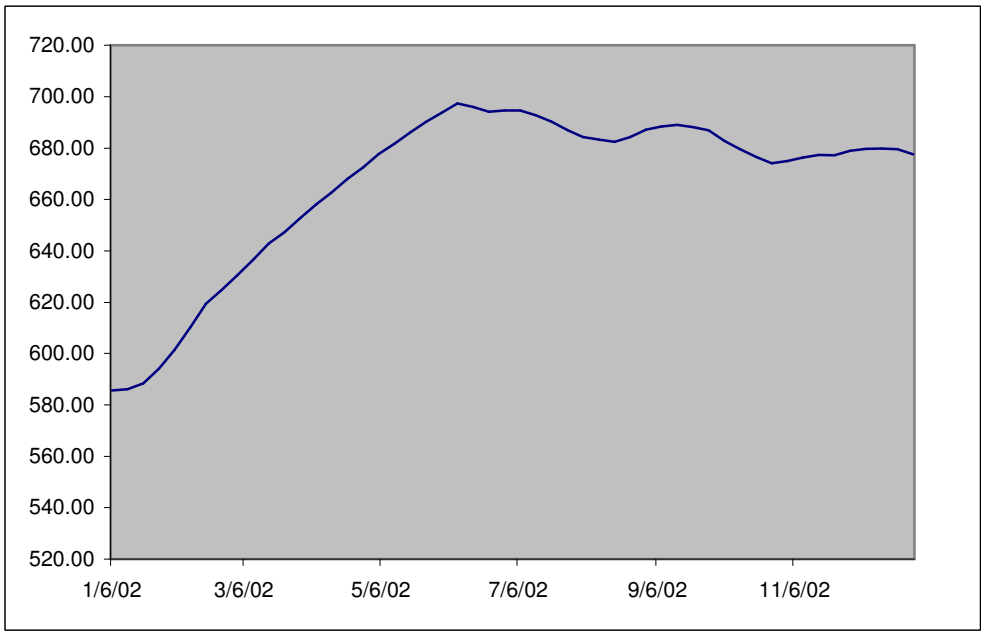
Barclays 2002-2003



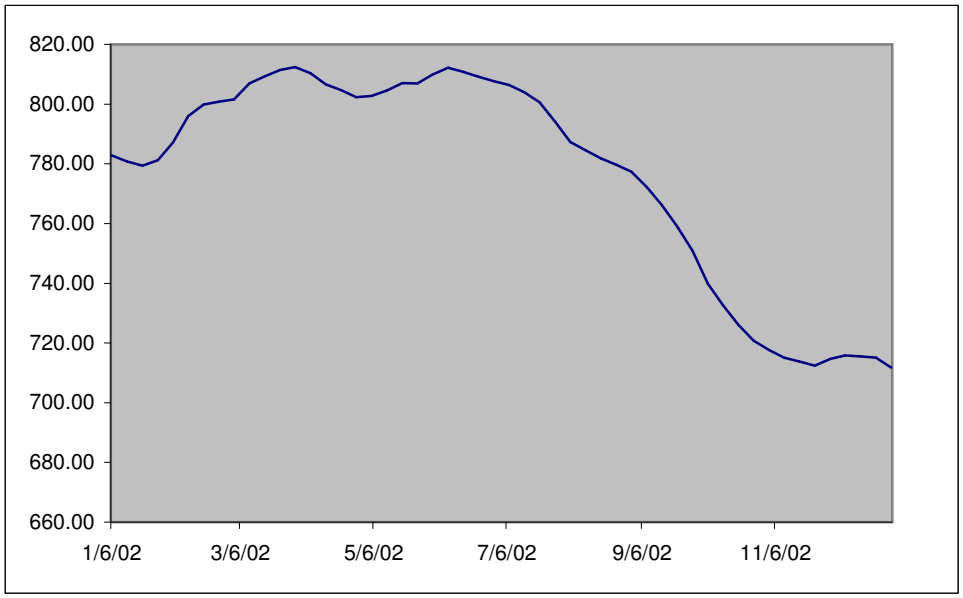
HBOS 2002-2003



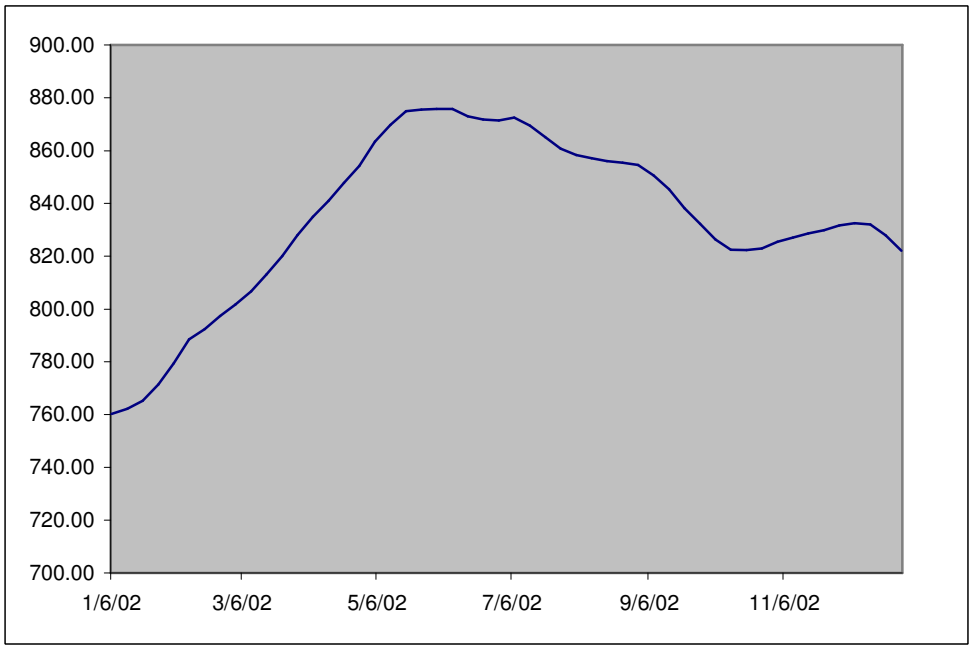
Northern Rock 2002-2003



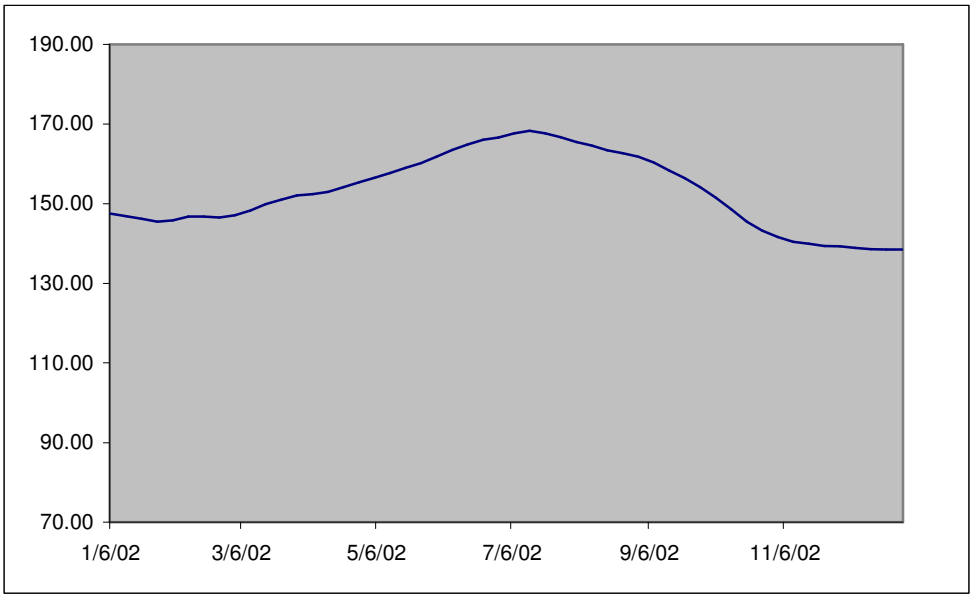
HSBC (UK) 2002-2003



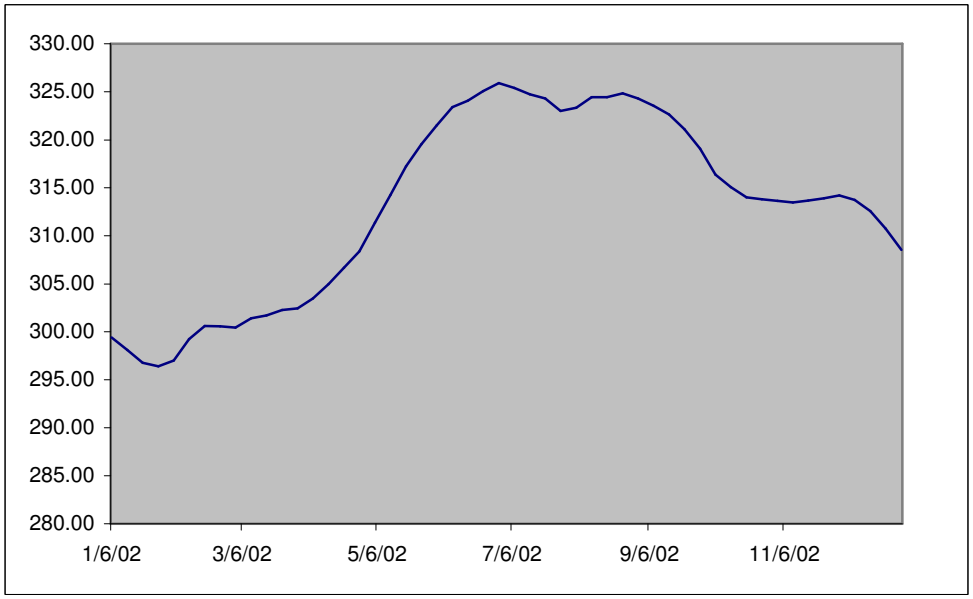
Alliance & Leicester PLC 2002-2003



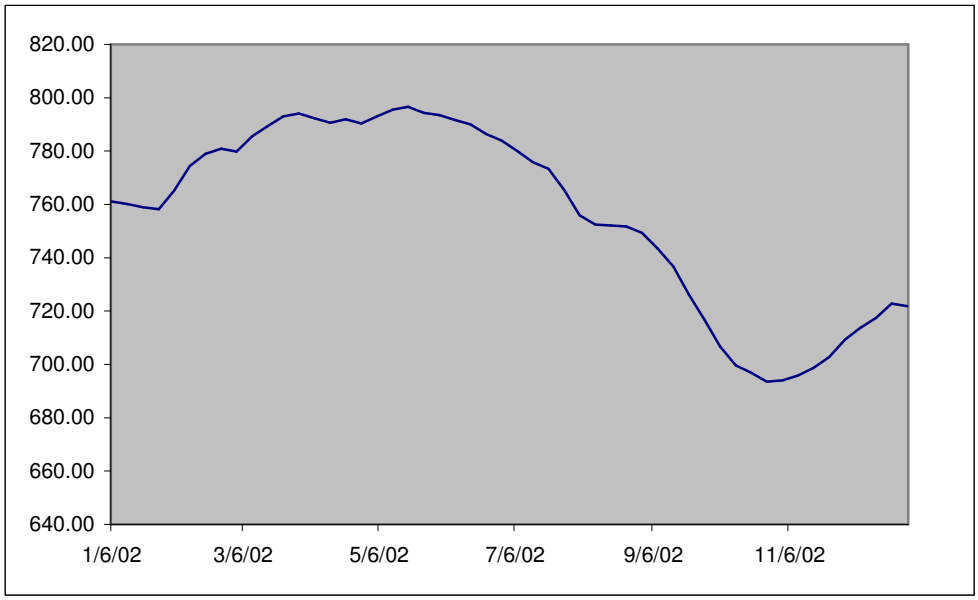
EGG PLC 2002-2003



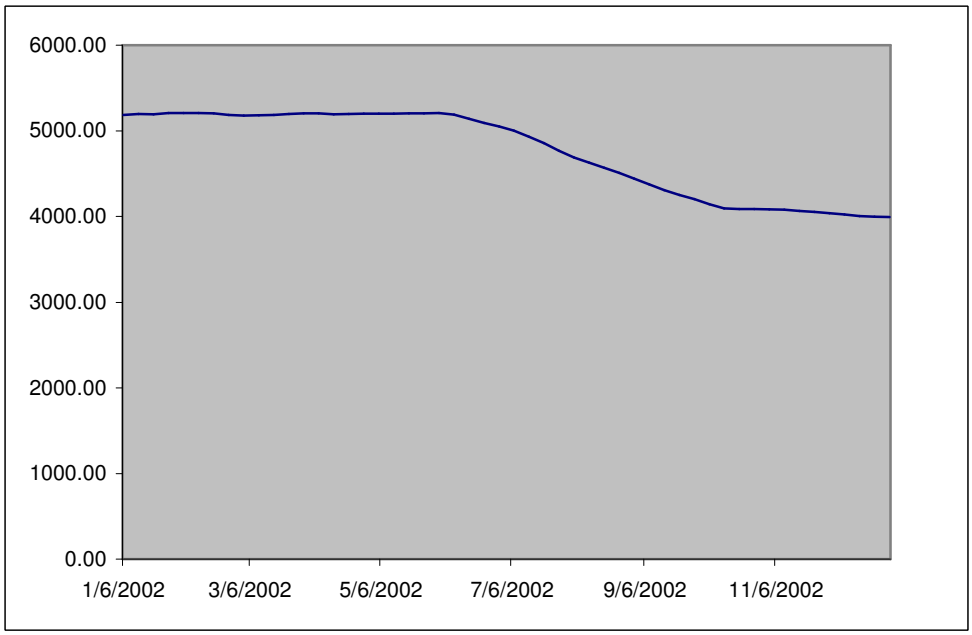
Bradford & Bingley 2002-2003



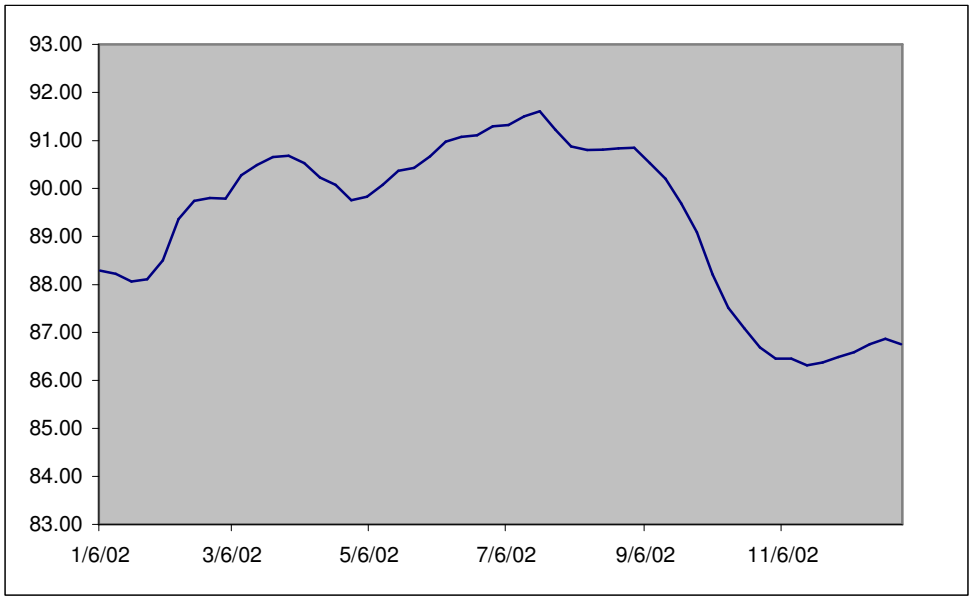
Standard Charter 2002-2003



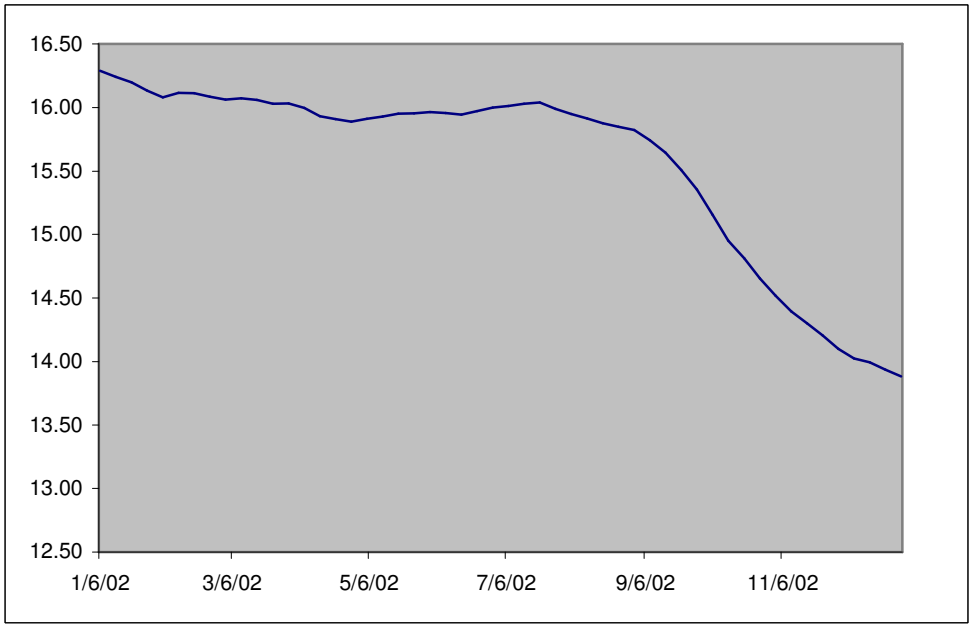
FTSE index 2002-2003



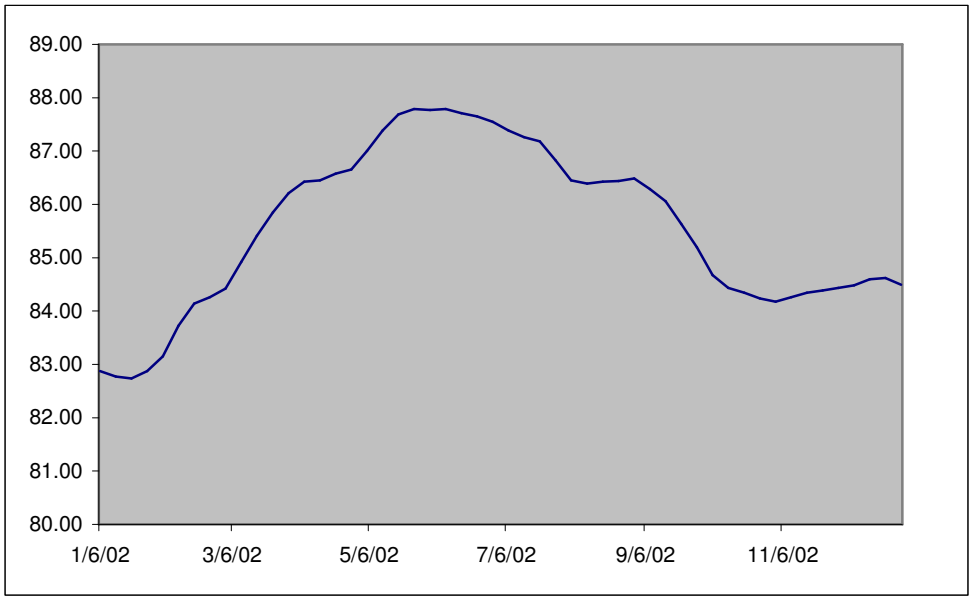
HSBC (HK) 2002-2003



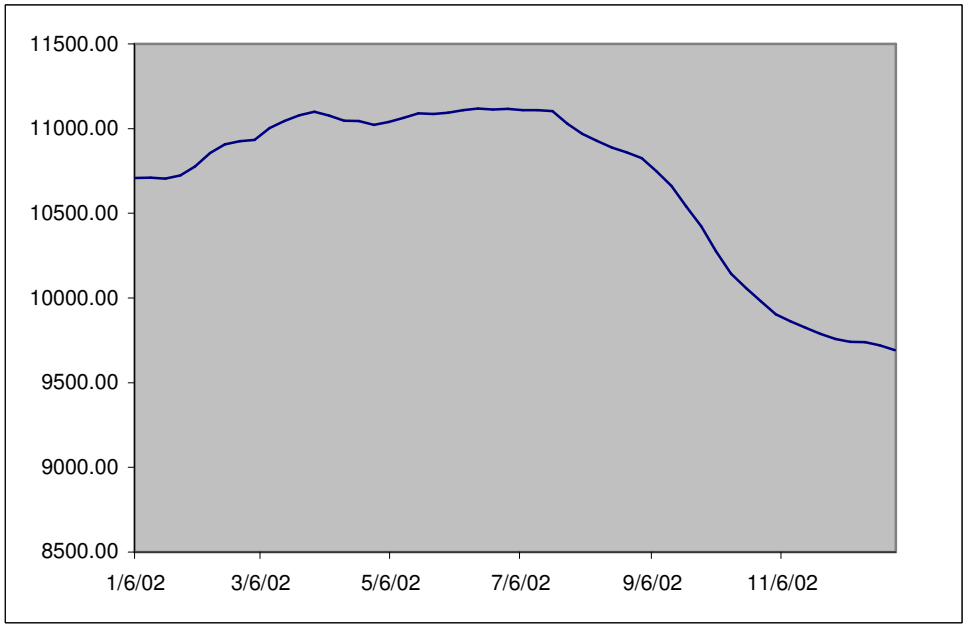
Bank of East Asia 2002-2003



Hang Seng Bank 2002-2003

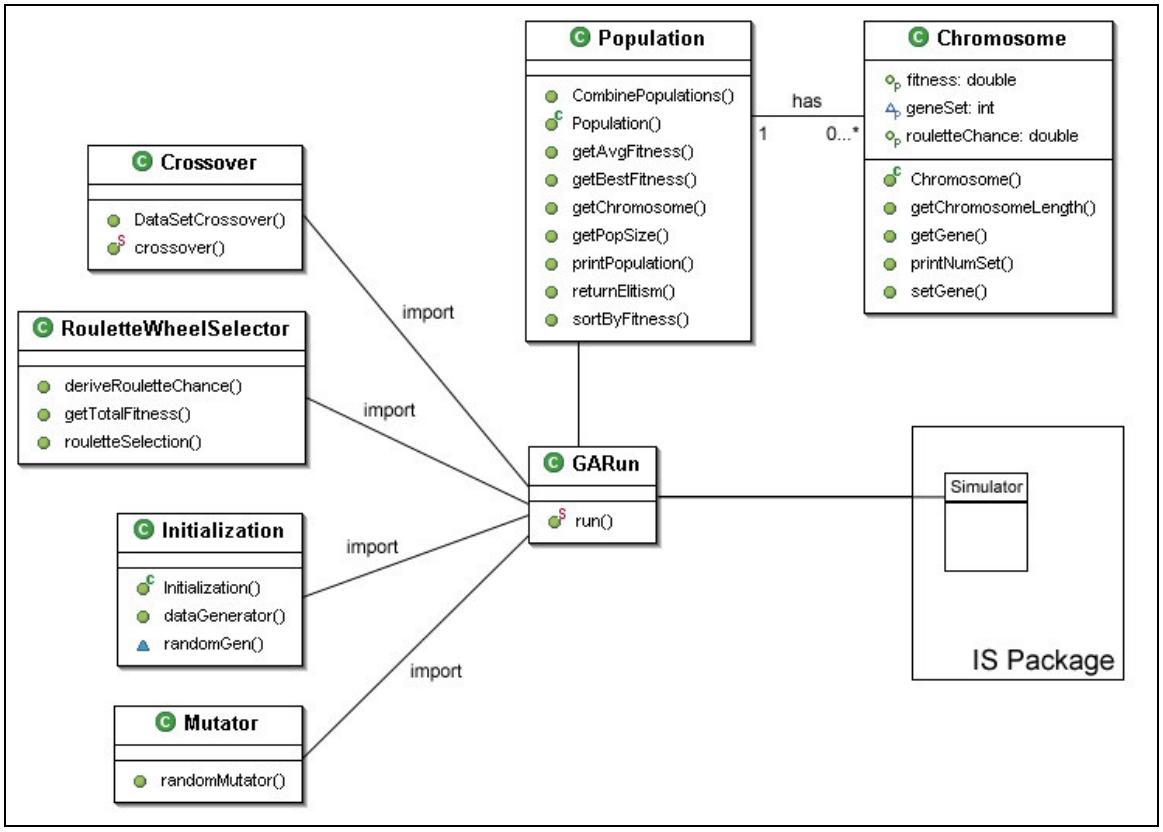


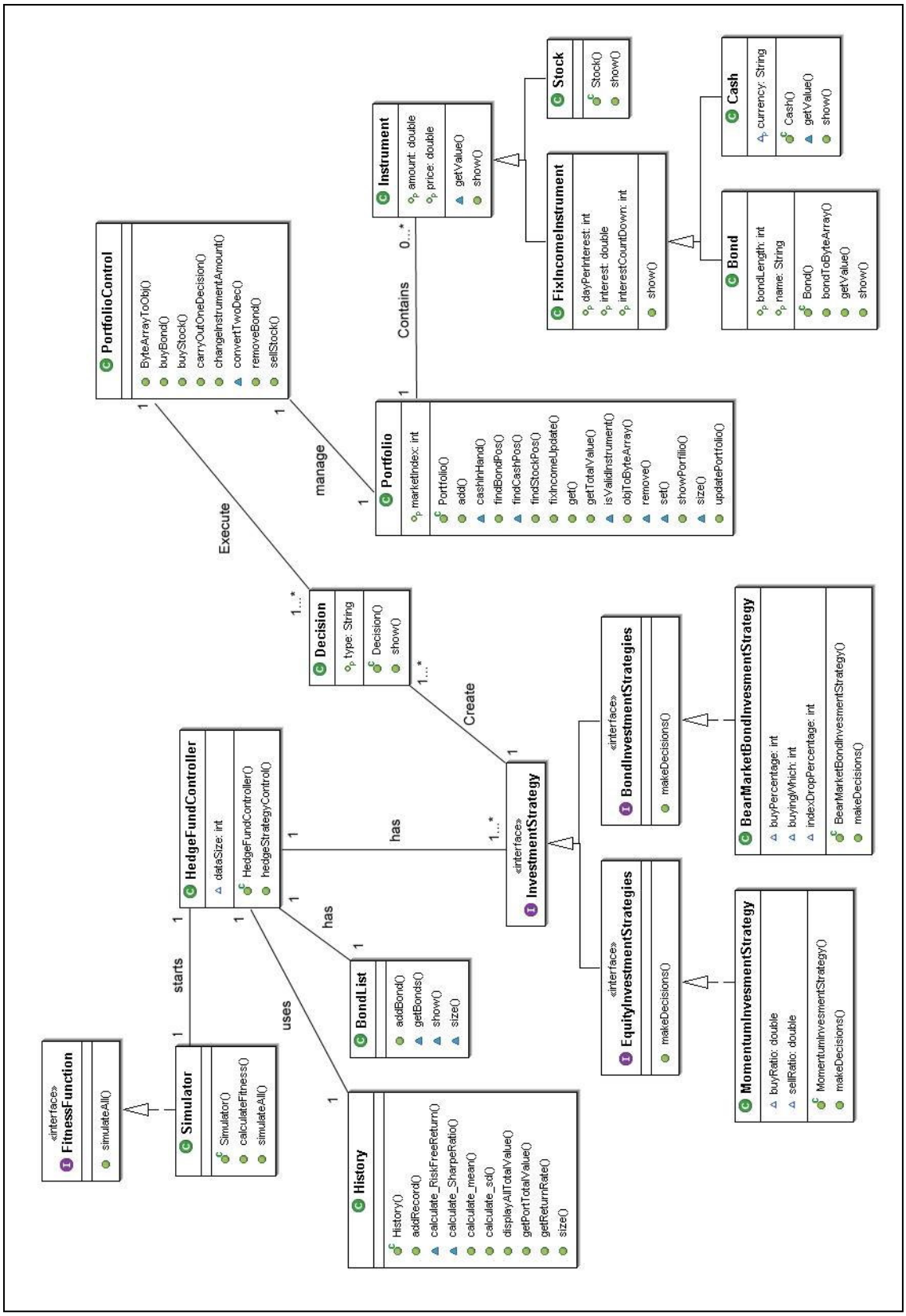
Hang Seng Index (HSI) 2002-2003



Appendix D: UML Class Diagram

The overall UML class diagram is too big so I separated it into two class diagrams and these two diagrams only shows the main functions in GA and IS, other functions such as importing data and exporting result are not shown.




«interface»
FitnessFunction
simulateAll()
Simulator
Simulator()
calculateFitness()
simulateAll()
starts
uses
HedgeFundController
dataSize: int
HedgeFundController()
hedgeStrategyControl()
has
has
BondList
addBond()
getBonds()
show()
size()
History
History()
addRecord()
calculate_RiskFreeReturn()
calculate_SharpeRatio()
calculate_mean()
calculate_sd()
displayAllTotalValue()
getPortTotalValue()
getReturnRate()
size()
Execute
Create
Decision
type: String
Decision()
show()
«interface»
InvestmentStrategy
«interface»
BondInvestmentStrategies
makeDecisions()
«interface»
EquityInvestmentStrategies
makeDecisions()
BearMarketBondInvesmentStrategy
buyPercentage: int
buyingWhich: int
indexDropPercentage: int
BearMarketBondInvesmentStrategy()
makeDecisions()
MomentumInvesmentStrategy
buyRatio: double
sellRatio: double
MomentumInvesmentStrategy()
makeDecisions()
PortfolioControl
ByteArrayToObj()
buyBond()
buyStock()
carryOutOneDecision()
changeInstrumentAmount()
convertTwoDec()
removeBond()
sellStock()
manage
Contains
Portfolio
marketIndex: int
Portfolio()
add()
cashInHand()
findBondPos()
findCashPos()
findStockPos()
fixIncomeUpdate()
get()
getTotalValue()
isValidInstrument()
objToByteArray()
remove()
set()
showPorfilio()
size()
updatePortfolio()
Instrument
amount: double
price: double
getValue()
show()
Stock
Stock()
show()
FixIncomeInstrument
dayPerInterest: int
interest: double
interestCountDown: int
show()
Bond
bondLength: int
name: String
Bond()
bondToByteArray()
getValue()
show()
Cash
currency: String
Cash()
getValue()
show()

Appendix E: System Manual

This manual outline the software requirements of the system, this section will also provide information on the CD enclosed in the report and the file system of the program. It will also provide the source of softwares that are used in the development of the program.

Software Requirements

- Java 2 Standard Edition JDK 5.0
- JUnit 3.8.1
- Eclipse SDK 3.02 (Optional but recommended)

Computer System Requirements

Pentium III or above, AMD Athlon or above
Windows® 95/98/NT/XP, Mandrake Linux, Fedora Core 3
128 MB RAM Minimum, 256 MB RAM recommended
Hard Disk: 250 MB for Eclipse only
CD Rom Drive (For Installation from CD)

Obtaining Softwares

Required softwares

Java 2 Standard Edition JDK 5.0
- www.java.sun.com

JUnit 3.8.1
- www.junit.org

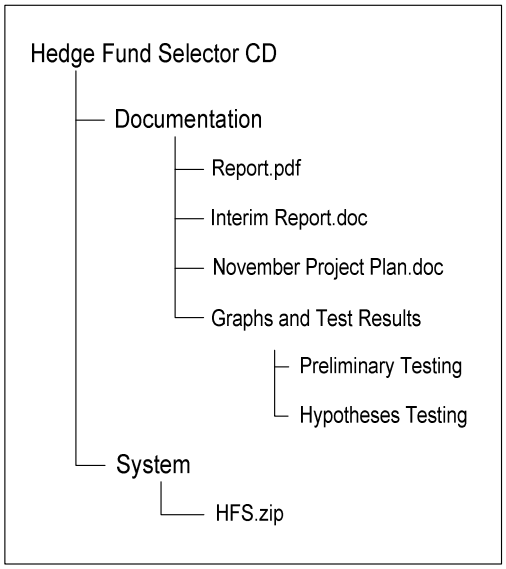
Eclipse SDK 3.02
- www.eclipse.com

Other software used in the system

Omondo Eclipse SDK 2.0.0
- http://www.omondo.com

S-Plus 6.2 Student Edition
- http://elms03.e-academy.com/splus/
Note: a copy of the software can be obtained by registering in this website using the UCL CS email.

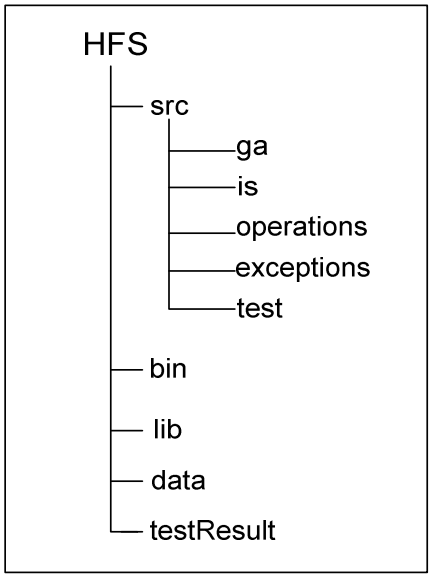
File Structure on the CD



- Report.pdf – The file of this report.
- Interim Report.doc - The file of the interim report.
- November Project Plan.doc – The file of the November Project Plan.
- Graphs and Test Result – The folder containing all data obtained from the experiments.
- HFS.zip – the source code of the system.

File Structure of the system

After extracting HFS.zip, the file structure is shown in the diagram below:



- src – This folder contains the source code of the system.
- ga – This folder contains all the .java files for the GA module.
- is – This folder contains all the .java files for the IS module.
- operations – This folder contains all the .java files for all extra functions of the system.
- test – This contains all the code for unit testing.
- bin – This folder contains all the .class objects.
- lib - This folder contains all the library class that are required by the system.
- data – This folder contains all the data input for the simulation.
- testResult – the outputs of the system.

Appendix F: User Manual

This manual will explain to the reader the method of installing the program, usage and running unit testing.

Installation

1. Before installing the system, all the softwares required must be install correctly, information on installing those softwares can be found in the softwares' documentation.
2. Extracting HFS.zip from the CD into the destination folder.
3. Start Eclipse.
4. In Eclipse, select "File" from the menu bar.
5. Select "New > Project".
6. Select "Java Project" in the pop up window and then click "Next".
7. Enter a name for the "Project Name" field.
8. In the "location" section, select the "Create project at external location" radio box.
9. Browse and select the destination folder in the "directory" field.
10. Click "Next".
11. Select the "libraries" tab in the window.
12. Click "Add external jar".
13. Browse and select "junit.jar".
14. Click "Finish" to complete installation.

Compilation

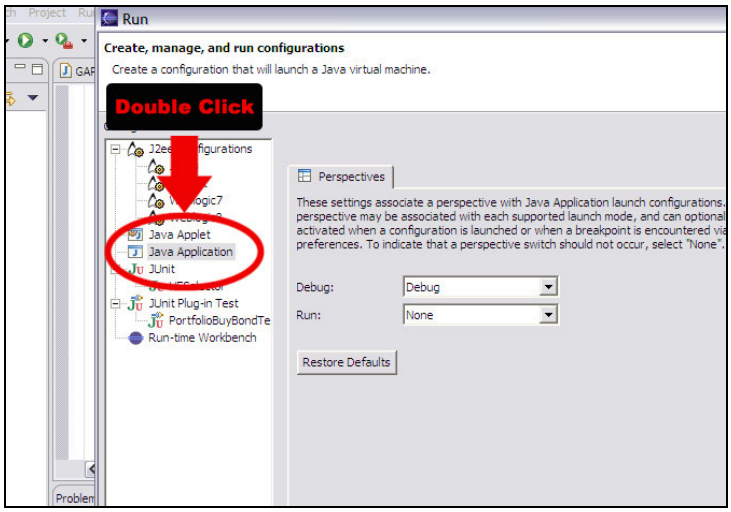
Compilation is not needed in Eclipse, because Eclipse builds the source code automatically.

Running the System

The system is control using a text input user interface.

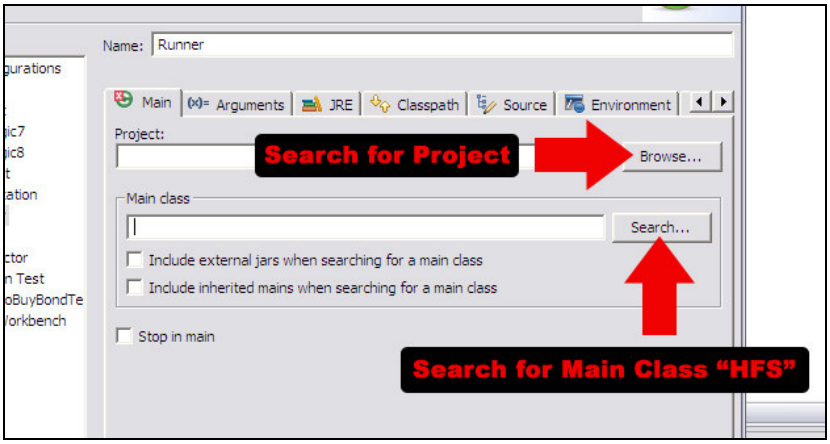
Running the system using the text input user interface

1. In Eclipse, select "Run" from the menu bar.
2. Select "Run…"
3. On the left section of the pop up window, double click on the "Java Application" icon. (Screenshot A)



Screenshot A

4. In the "Project" field, click "Browse" to find the project and in the main class field, click "Search" to find "HFS" main class. (Screenshot B)



Screenshot B

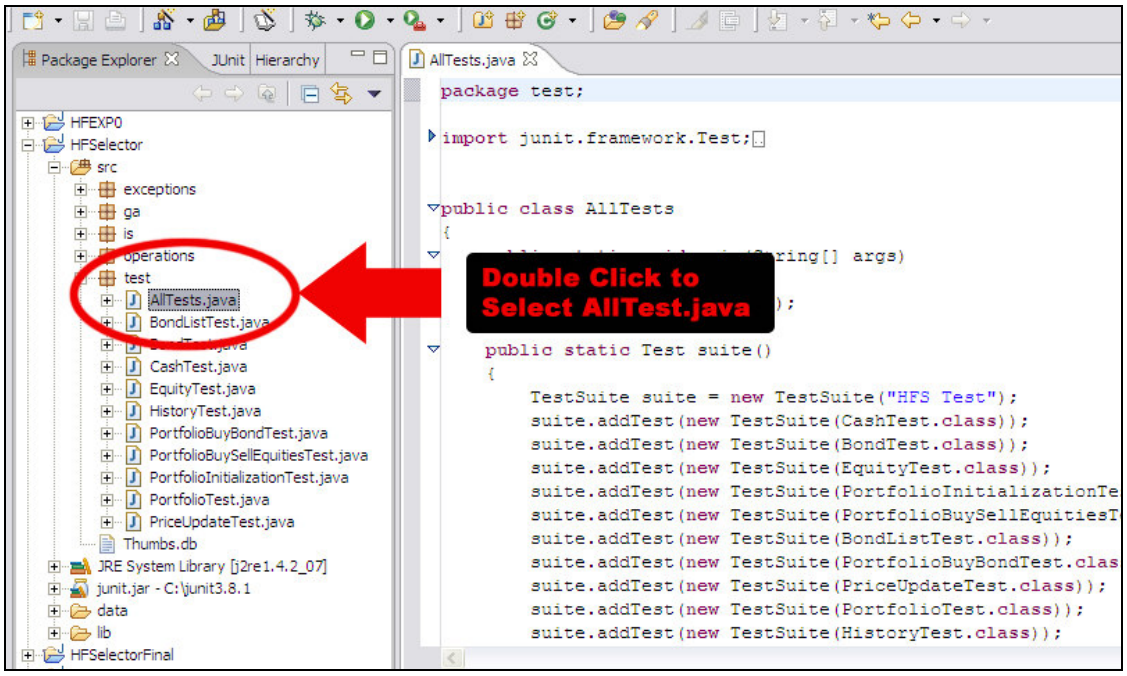
5. Click "Run" in the bottom of the pop up window to run.
6. The system will start in the console of Eclipse. (Screenshot C)

Screenshot C

7. To start a normal run of the system, user enters “normalRun”. The system will then ask user for information about the stock market and settings for GA. After the system has acquired all information, simulation will start and the system will display the chromosome and its fitness when the system has completed simulation. (Note: all commands are case sensitive)
8. To start a run with the GA module disable, user enters “benchmarkRun”. The system will then ask user for information about the stock market and in addition the system will ask for 6 values for the 6 genes in the chromosome. After all information is gathered, the system will compute the fitness.

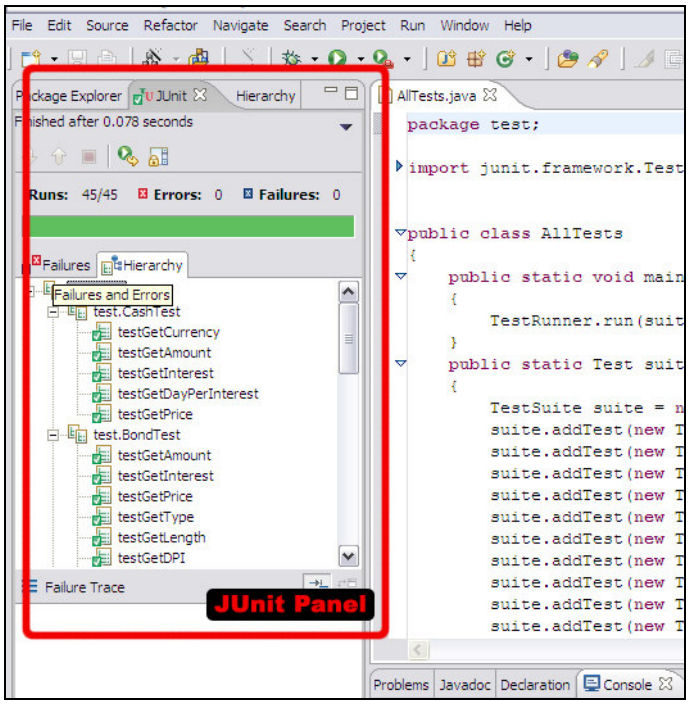
Running Unit Testing of the System

1. In Eclipse, select the AllTest.java in the package explorer by opening the “test” package of the project. (Screenshot D)



Screenshot D

2. Select “Run” from the menu bar.
3. Select “Run As” from the menu bar.
4. Click “4 JUnit Test”
5. This test will start and the package explorer will change to the JUnit test result explorer. The result will be shown there.



6. The Green bar shows that all tests have passed with no error.

Format of the Data Input

Data are in text file and in a single column of number or doubles down the file.

Troubleshooting

For any problem, please contact me by emailing to nicky.ntsang@gmail.com

Appendix G: Code Listing

Due to the large amount of classes for the system, not all the code can be display here. I will only list the important classes such as the GA operators and the main controlling class in the investment simulator. All other code can be found in the CD enclosed.

Bibliography

[1] Baker, J. E., (1985). Adaptive selection methods for genetic algorithms. In J. J. Grenfenstette, editors, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. Erlbaum.

[2] Bäck, T., (1996). *Evolutionary Algorithms in Theory and Practice,* Oxford University Press.

[3] Bäck, T. and Hoffmeister, F. (1991). Extended Selection Mechanisms in Genetic Algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithm*. 92-99, Morgan Kaufmann.

[4] Bodie, Z., Kane, A. and Marcus, A.J. (2002). *Investment,* McGraw-Hill.

[5] Bramlette, M. F. (1991). Initialization, Mutation and Selection Methods in Genetic Algorithms for Function Optimization. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithm*. 100-107, Morgan Kaufmann.

[6] Darwin, C.R., (1859). *On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life,* John Murray, London.

[7] De Jong, K., (1975). *The Analysis and behavior of a class of Genetic Adaptive System*, PhD thesis, University of Michigan.

[8] Falkenauer, E. (1994). A New Representation and Operators for Genetic Algorithms Applied to Grouping Problems. In K. De Jong, editors, *Evolutionary Computation Volume 2, Number 2, Summer 1994.* MIT Press.

[9] Goldberg, D.E (1989). *Genetic Algorithms in search, Optimization, and Machine Learning*. Addison – Wesley.

[10] Grenfenstette, J. J and Fitzpatrick, J. M (1985). Genetic search with approximate Function Evaluations. In J. J. Grenfenstette, editors, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. Erlbaum.

[11] Holland, J.H (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

[12] Janikow, C. Z and Michalewicz, Z. (1991). An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms . In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithm*. 31-36, Morgan Kaufmann.

[13] Mitchell, M (1998). *An Introduction to Genetic Algorithms.* MIT Press.

[14] Reeves, C, R (1993). Using Genetic Algorithms with Small Population. In Stephanie Forrest, editors, *Proceedings of the Fifth International Conference on Genetic Algorithms*. Morgan Kaufmann.

[15] Schaffer, J. D. and Eshelman, L. J. (1991). Preventing Premature Convergence in Genetic Algorithms by Preventing Incest. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithm*. Morgan Kaufmann.

[16] Schaffer, J. D., Eshelman, L. J., Caruana, R.A. and Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In J. D. Schaffer, editors, *Proceedings of the third International Conference on Genetic Algorithm*. Morgan Kaufmann.

[17] Smith, T., Husbands, P., Layzell, P. and O'Shea, M. (2002). Fitness Landscapes and Evolvability. In K. De Jong, editors, *Evolutionary Computation Volume 10, Number 1, Summer 2002.* 1-34, MIT Press.

[18] Jones, T., (1995) *Crossover, macromutation and population-based search*. In L. Eshelman, editor, Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95), pages 310-317, Pittsburgh, PA, USA, 15-19 July 1995. Morgan Kaufmann.

[19] Huelsbergen, L., (1998) Finding General Solutions to the Parity Problem by Evolving Machine-Language Representations. *Proceedings of the Conference on Genetic Programming*, pp. 158-166, July 1998.

[20] Angeline, P.J, (1997), *Subtree Crossover: Building Block Engine or Macromutation?.* In Koza, John R., Deb, Kalyanmoy, Dorigo, Marco, Fogel, David B., Garzon, Max, Iba, Hitoshi, and Riolo, Rick L. (editors). 1997. *Genetic Programming 1997: Proceedings of the Second Annual Conference, July 13&shyp;16, 1997, Stanford University*. San Francisco, CA: Morgan Kaufmann.