Solving Cardinality Constrained Portfolio Optimisation Problem Using Genetic Algorithms and Ant Colony Optimisation

Yibo Li

A thesis submitted for the degree of Doctor of Philosophy

School of Information Systems, Computing and Mathematics,

Brunel University

2015

Abstract

In this thesis we consider solution approaches for the index tacking problem, in which we aim to reproduces the performance of a market index without purchasing all of the stocks that constitute the index. We solve the problem using three different solution approaches: Mixed Integer Programming (MIP), Genetic Algorithms (GAs), and Ant-colony Optimization (ACO) Algorithm by limiting the number of stocks that can be held. Each index is also assigned with different cardinalities to examine the change to the solution values.

All of the solution approaches are tested by considering eight market indices. The smallest data set only consists of 31 stocks whereas the largest data set includes over 2000 stocks. The computational results from the MIP are used as the benchmark to measure the performance of the other solution approaches. The Computational results are presented for different solution approaches and conclusions are given.

Finally, we implement post analysis and investigate the best tracking portfolios achieved from the three solution approaches. We summarise the findings of the investigation, and in turn, we further improve some of the algorithms.

As the formulations of these problems are mixed-integer linear programs, we use the solver 'Cplex' to solve the problems. All of the programming is coded in AMPL.

Acknowledgement

First, I would like to thank my supervisor Dr. Cormac Lucas for his excellent guidance, constant support, and great inspiration through my Ph.D. studies. It was great pleasure to work with him. I would also like to thank my parents and other family members, without their help and support none of these would have been possible. I am very lucky to have all of you stand by my side and I am very grateful for all the things you have done for me.

I would like to thank all the staff in the Mathematics Department for their excellent work. Finally, I would also like to thank my Ph.D. colleagues and friends: Cristiano, Xiang, Antonio, Zhenghong, Longhui, Martin, Fei and many others.

Table of Contents

List of Figures

List of Tables

Chapter 1

1 .Introduction

1.1 Introduction

Fund managers are often hired to control a large amount of money which is invested in various securities and other assets such as: real estate. The investor can be categorized into two groups: institutions and private investors. Fund managers usually provide investment management services that include financial statement analysis, asset/stock selection, and direction of management plans and ongoing monitoring of investments. The aim for fund managers is to make capital growth and income over the short-term or the long-term. The basic investment strategies adopted by fund managers can be broadly classified into two types: active management and passive management.

- **Active management**

For active management, the fund managers have more confidence in their own ability to estimate cash flows, growth rates, and discount rates. Based on these estimates, they value assets and determine whether an asset is fairly valued. In an actively managed portfolio, assets that are undervalued, or have a chance of offering above-normal returns, will have a higher weight than that in the index, whereas other assets will have a zero weight, or even negative weight if short selling is permitted. This style of investing is called active management, and the portfolios are referred to as active portfolios. Most open-end mutual funds and hedge funds practice active management, and most analysts believe that active investment can provide capital growth. Active management often has high fixed cost (payments to the management team) and its high frequency of trading often incurs high transaction costs. If the market performs well these fees will be offset by the returns.

- **Passive management**

This is based on the assumption of an efficient market. If the markets are efficient, the price in the market is an unbiased estimate of all future discounted cash flows. In other words, the price aggregates and reflects all information that is publicly available, and investors cannot expect to earn a return that is greater than the required rate of return for that asset. So there is no way to outperform the market. In that case, a simple and convenient approach to investing is to rely on the prices set by the market. Portfolios that are based on the assumption of unbiased market prices are referred to as passive portfolios. Passive portfolios most commonly replicate and track market indices, which are passively constructed on the basis of market prices and market capitalizations. Examples of market indices are the FTSE 100, the Nikkei 225, and the S&P 500. Passive portfolios based on market indices are called index funds and generally have low running costs because no significant effort is expended in valuing securities that are included in an index. By comparison, passive management has lower fixed costs and lower transaction costs, but once the market falls it will inevitably affect the index fund.

Historical evidence reveals that active fund managers averagely underperform their corresponding benchmarks. Researchers also found that some of the best active fund managers did perform reasonably well in some periods, but most of them failed to carry their success over a long-term period. The following two figures show the evidence found by researchers from ‘Vanguard’ (Charles Thomas, Peter Westaway and Todd Schlanger, n.d.):

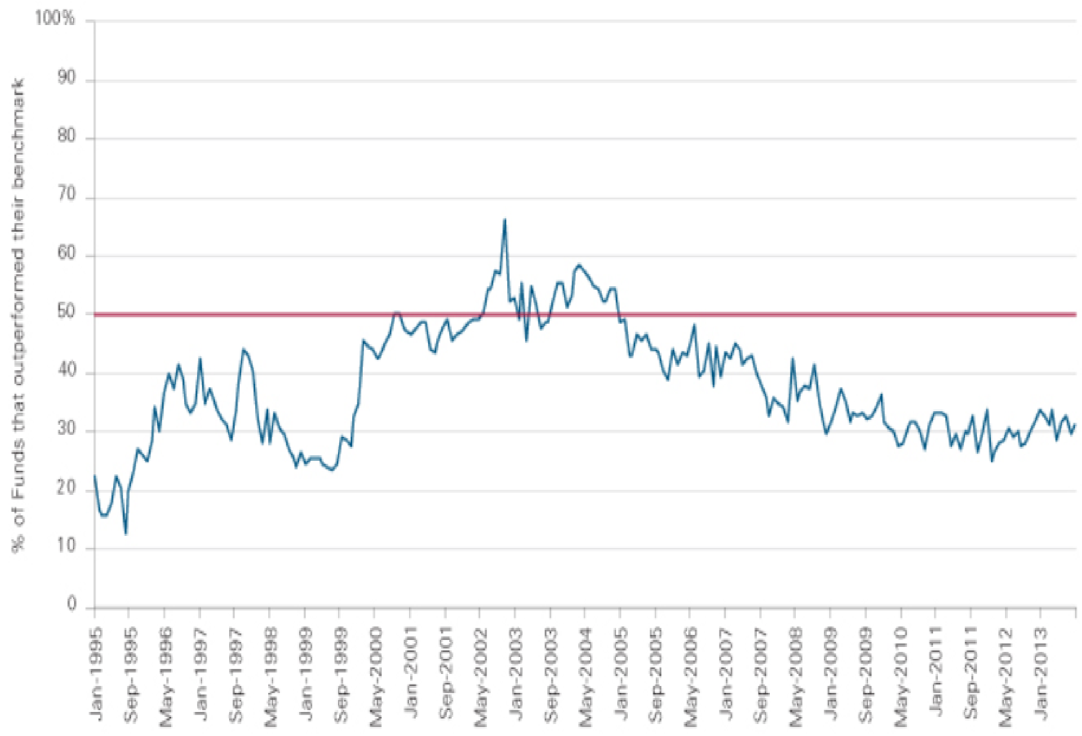


Figure 1.1: Percentage of active managers outperforming their benchmark over rolling five-year periods

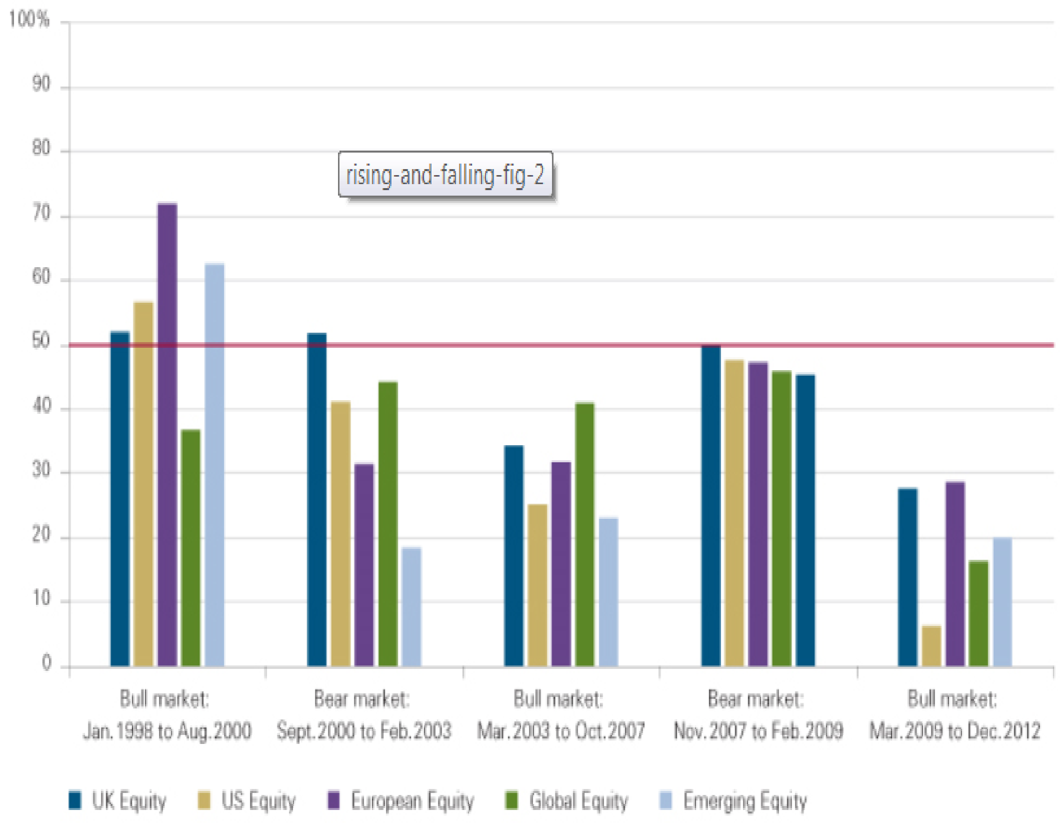


Figure 1.2: Percentage of active managers outperforming their benchmark during bull/bear cycles

The research shows that active managers failed to provide any consistency in either bull or bear markets, and the majority tend to underperform their benchmarks over time. Because of that, both in the USA and in Europe, passive management has been receiving a much higher profile recently. Index-Tracking is a method of passive portfolio management that attempts to track a single given index, such as the FTSE-100 or the S&P 500. The simplest way to reproduce an index is to purchase all of its constituents assigning the same weights as given in the index. This approach can achieve a perfect match. However, it has many disadvantages: certain stocks may only contribute a tiny proportion to the whole index thus reconstruction of the index may require rebalancing of all stocks which incurs high transaction costs. Furthermore stocks of small companies may be illiquid which also entails relatively high transaction costs (Beasley, J.E., Meade, N. and Chang. T.J., 2003). Due to these disadvantages, fund managers usually use a subset of the stocks in the index, subsequently introducing a tracking-error, which can be illustrated on a graph showing the gap between the tracking portfolio and the index. The problem, then, is concerned with the selection of a tracking portfolio that best matches the performance of the market whilst avoiding a too diversified portfolio.

Generally, there are four mathematical techniques used for solving portfolio optimization problem, which are Quadratic Programming, Nonlinear Programming, Mixed Integer Programming and Meta-Heuristic Methods. In this paper, we focus on the last two solution approaches. First, we introduce the Mixed Integer Programming approach. When we use MIP to solve the portfolio optimization problem, it can produce optimal solutions. However, the MIP approach is often time-consuming. Particularly, as the size of the data set becomes larger, the computation time increases fast, and there is a possibility that the computer cannot solve the problem due to the limitation in capacity, in which case we often set a time limitation to get the best results that we can achieve. In this research, we use the results from MIP to serve as the benchmarks to measure the performance of our heuristic methods. Broadly, we present two heuristic methods, Genetic Algorithms and Any-colony Optimization. The aim of using heuristic methods for the Index Tracking is to shorten the computing time while maintain a considerably high quality of the solutions. The results from the heuristic methods do not necessarily have to be optimal but usually we want them to be better than the corresponding results (solution quality or computing time) from MIP. In many cases, the optimal solution is unknown. Additionally, we implement post analysis and out-of-sample tests on the results of some selected indices. The work presented in this thesis does not consider transaction costs and the revision of an existing tracking portfolio. However, we can easily achieve these simply by adding the transaction costs back into the formulation. For convenience, we use 'tracking portfolio' to represents the stocks chosen to hold to track an index.

1.2 Thesis Structure

This thesis consists of seven chapters. In Chapter 2, we present a literature survey related to portfolio optimization and index tracking problems, where we consider the Markowitz model, CAPM, Factor models, Genetic algorithms and other works done by previous researchers.

In Chapter 3, we detail the MIP solution approach. Section 3.1 will present an introduction based on the solution approaches used to solve the Index Tracking problem in our research. In section 3.2, we review a Mixed Integer Programming solution approach developed by BMC (Beasley, J.E., Meade, N. and Chang. T.J., 2003). We compare their work with ours and list the major differences between the two approaches. In section 3.3, we present the formulation of our MIP solution approach. In section 3.4, we use the MIP approach to solve eight market indices (Hang Seng, DAX, FTSE, S&P, Nikkei, S&P-500, Russell-2000, and Russell-3000) by explicitly limiting the number of stocks that can be selected in a tracking portfolio and the computing time. The computing results of the MIP are given at the end of the section.

In Chapter 4, the details of genetic algorithms are discussed. In section 4.1, we briefly introduce GAs from three aspects: initial population, crossover and reproduction, and mutation. In section 4.2, we present a basic algorithm called the Inverse Triangle GA (IT-GA) and give enhancements to the basic algorithm by removing the duplicates and adding the stopping criteria. The new algorithm is named the Enhanced Inverse Triangle GA (EIT-GA). We use the EIT-GA to solve the eight market indices and compare the computing results with those of MIP. In the end of the section, we illustrate the disadvantage of the EIT-GA and justify why it is necessary to develop another GA approach. In section 4.3, we first introduce the foundation theory of GAs: Schema Theorem. Based on the theory, we present the Roulette GA (R-GA). We run it with several different data sets and find reliable relations amongst the cardinality, the population size, and the index size. Consequently, we improve the R-GA and use the enhanced algorithm to solve the eight market indices. Concluding the section, we compare the computational results with those of EIT-GA and give conclusions on both of the solution approaches.

In Chapter 5, we present the Ant-colony Optimization solution approach. In section 5.1, we give an introduction to the ACO approach. Section 5.2 details the three steps of the ACO algorithm: Construct Ants Solutions, Update Pheromones and Daemon Actions. In section 5.3, we build an ACO algorithm to solve the index tracking problem following the three steps. In addition, we present our mutation method and stopping criteria. In section 5.4, we carry out investigations on two parameters, colony size and evaporation rate. The findings are also shown at the end of the section. In section 5.5, we enable the artificial ants to have more searching powers and run the algorithm on the eight market indices. In section 5.6, we compare the computational results with those of ER-GA and give conclusions on the approach.

In Chapter 6, we present the post analysis. In section 6.1, we investigate the stocks of the tracking portfolios of the first five indices. We discover three things which we refer to as, *the close stock effect, the combination effect, and the inheritance effect*. Particularly, we use the inheritance effect to improve the ER-GA and the ACO. In section 6.2, we present the out-of-sample performance of the first five indices from time [51, 60] and the conclusion is also given at the end of the section.

Chapter 2

2 .Literature Review

2.1 History of Portfolio Theory

2.1.1 Mean-Variance Model

Modern Portfolio theory, fathered by Harry Markowitz, is a theory that aims to maximize the portfolio expected return for a given amount of portfolio risk or minimize risk for a given level of expected return, by choosing the proportions of assets to hold in the portfolio. Markowitz introduced a mean-variance model, also known as the Markowitz Model, in his pioneering article in 1952 (Markowitz, 1952) and subsequent book published in 1959 (Markowitz, 1959). Based on the risk-return characteristics of portfolios, his model attempts to select the most efficient portfolios of the given securities. Before developing the model, Markowitz made several assumptions concerning the investment market and investors' behaviours. The key assumptions can be summarized as follows:

1. Markets are perfectly efficient.
2. Investors seek to maximize their expected return.
3. All investors have the same expected single period investment horizon.

4. All investors are risk-adverse, that is they will only accept greater risk if they are compensated with a higher expected return.

5. Investors base their investment decisions on the risk-return characteristic.

Other assumptions are listed below:

1. No taxes or transaction costs are involved

2. Investors can buy any security of any size

3. Investors can lend or borrow any amount of money at the risk free rate

4. The correlations between assets are always fixed and constant

5. The return on assets are normally distributed

The Markowitz model is defined as follows:

Consider a portfolio that consists of n securities 1 … n. Each security is weighted on a percentage basis by $\omega_1 \ldots \omega_n$ where the sum of the weights equals one. Let μ_i be the expected return of security i. Then the expected return of the portfolio is given by the following equation:

$$\mu_p = \sum_{i=1}^{n} \omega_i \mu_i$$

If σ_i is the standard deviation of security i, then the risk of the portfolio can be defined as below:

$$\sigma^2 = \sum_i \sum_j \omega_i \, \omega_j \sigma_{ij}$$

Where σ_{ij} is the covariance between the i^{th} and j^{th} securities, given as:

$$\sigma_{ij} = \sigma_i \sigma_j \rho_{ij}$$

Where ρ_{ij} is the correlation between the i^{th} and j^{th} securities, whose value is between -1 and 1. The Mean-Variance Model assumes that investors prefer a portfolio with higher return and lower risk. A portfolio that gives maximum return for a given risk, or a minimum risk for a given return is an efficient portfolio.

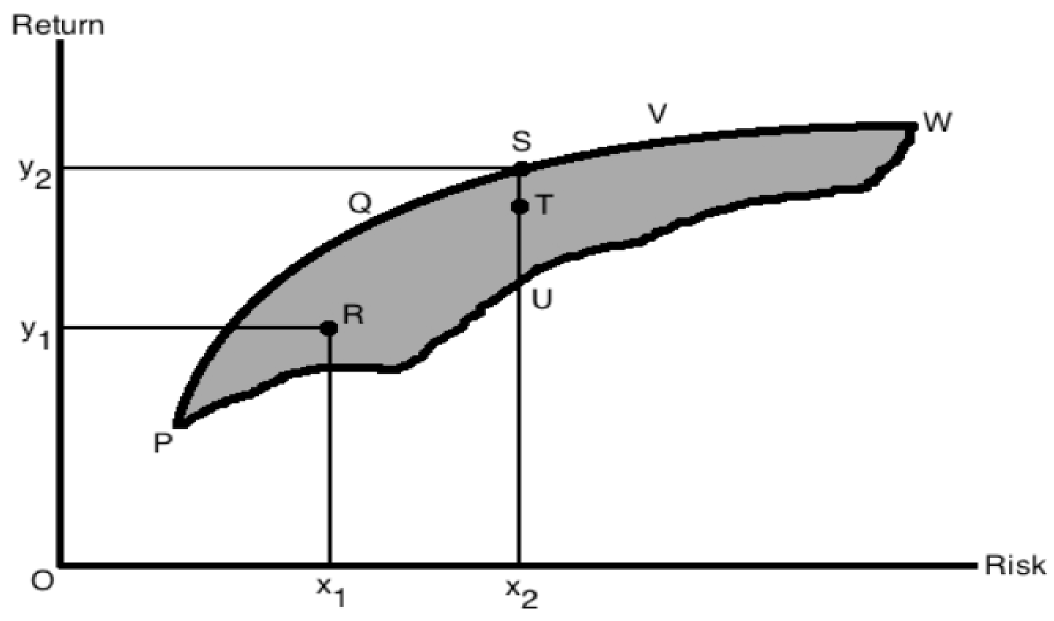


Figure 2.1: The efficient frontier

In figure 2.1 above, the shaded area PQWP includes a subset of all the possible portfolios that can be invested in. The curve PQW is called the Efficient Frontier. The portfolios that fall on the curve are called Efficient Portfolios as all portfolios on the curve have the maximum return for a given risk or the minimum risk for a given return. The Efficient Frontier is the same for all investors, as it is assumed all investors want maximum return with the lowest possible risk and they are risk averse. The tangent line to the Efficient Frontier is called the Capital Market Line, as shown below.

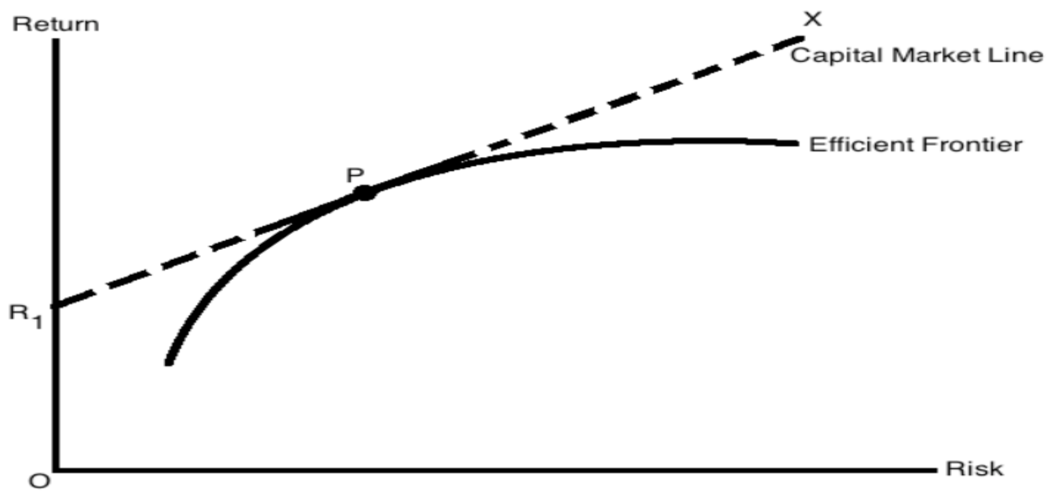


Figure 2.2: The capital market line

The line R_1PX is the Capital Market Line (CML) that represents the trade-off of risk and return in the capital market. It means that an investor will take higher risk if the return of the portfolio is higher. The tangent point P represents the Market Portfolio, which is known as the most efficient, and diversified portfolio. It consists of all shares in the capital market. This is the ultimate aim for all investors. The capital market consists of risky and risk-free securities and the CML is the optimal combination of them. The Capital Market Line states that the return of a portfolio is the risk-free rate plus a risk premium. The equation is shown below:

$$ER_p = r_f + (ER_m - r_f)\,\sigma_p / \sigma_m$$

ER_p	The expected return of the portfolio
ER_m	The expected return on the market portfolio
r_f	The risk-free rate
σ_m	The standard deviation of the market portfolio
σ_p	The standard deviation of the portfolio

When the risk-free rate is introduced, investors can choose any portfolio on the CML by lending or borrowing at the risk free rate. The portfolio that an investor will choose depends on their preference to risk. In the above figure, the portfolios on the section from R_1 to P represent the Lending Portfolio. In this section, an investor will lend at the risk-free rate. The portfolios on the section beyond P represent the Borrowing Portfolio, where an investor can borrow money at the risk-free rate and invest in the market portfolio P (Markowitz, 1991). The Mean-Variance model is a very important model for portfolio optimization, yet it also has many disadvantages. Primarily, it requires an investor to obtain many data, such as the variance of returns, the covariance of returns, the correlation between two securities and estimates of returns for all securities.

2.1.2 Capital Asset Pricing Model

Building on the work of Markowitz on modern portfolio theory, Sharpe (Sharpe, 1964), Treynor (Treynor, 1961), Lintner (Lintner, 1965) and Mossin (Mossin, 1966) introduced the Capital Asset Pricing Model (CAPM) independently. The model aims to predict the expected return of a risky asset. It describes the expected return of a single risk-asset as consisting of two parts: the risk-free rate and a risk-premium. The CAPM model assumes:

- Investors have the same expectations
- Investors can borrow or lend any amounts at the risk-free rate
- The market is in equilibrium at all times

The CAPM model for a given asset is given by:

$$ER_i = r_f + \beta_i(ER_m - r_f)$$

ER_i	The expected return of the risky asset
r_f	The risk-free rate
β_i	Represents the systematic risk (non-diversified risk) of an asset, which is represented by: $\beta_i = Cov(r_i, r_m)/Var(r_m)$
ER_m	The expected return of the market
$ER_m - r_f$	The market risk premium

The CAPM is restricted both theoretically and practically. Theoretically, CAPM is a single factor model, where except for systematic risk, no other investment characteristics are considered for estimating returns. In addition, it is a single period model that does not involve multi-period implications so that it can lead to suboptimal investment decisions. For practical limitations, they often arise in implementing the model. According to CAPM, the market portfolio includes all assets, which means that it may involve some assets that are not available for investment. Richard Roll noted that one reason the CAPM is not testable is that the true market portfolio is unobservable (Roll, 1977). Other issues are the estimation of beta risk and the assumption of homogeneity in investor expectations. For the first one, it requires long historical returns in order to give the estimation. In addition, by using a different period for estimation may result in different estimates of beta. For the latter one, without the assumption there will be many optimal risky portfolios and countless security market lines. Obviously, investors can process the same information but arrive at different optimal risky portfolios. Another important issue is that the empirical support for the CAPM is weak. It often gives poor predictions of returns. Because of the limitations of the CAPM, several models have been proposed to address some of the problems. Due to numerous numbers of the models, we only introduce two typical ones in the following.

2.1.3 Factor Model

2.1.3.1 APT

The Arbitrage Pricing Model (APT) is based on the same principle as CAPM but it expands the number of risk factors. It was proposed by Stephen Ross (Ross, 1976). Similar to CAPM, APT proposes a linear relationship between expected return and risk. The Model is shown below:

$$ER_p = r_f + \lambda_1 \beta_{p,1} + \cdots + \lambda_k \beta_{p,k}$$

ER_p	The expected return of the portfolio
r_f	The risk-free rate
λ_j	The risk premium for factor j
$\beta_{p,j}$	The sensitivity of the portfolio to factor j
k	The number of risk factors

Although the APT model is superior to CAPM, however, in practice the CAPM is preferred to APT. As the APT does not specify any of the risk factors and it is very difficult to identify them and estimate their corresponding betas [also see (Rudd, 1980), (Haugen, R.A. and Baker, N.L., 1990), (Wilmott, 1998), (Larsen Jr., G.A. and Resnick, B.G., 1998), (Alexander, 2001), and (Elton, E., 2007)].

2.1.3.2 Carhart Model

Based on the analysis of the relationship between past returns and a variety of different factors, Fama and French (Fama, E. F., and French, K. R., 1992)addressed three factors to explain asset returns. These three factors are relative book-to-market value, relative size and beta of asset. Carhart (Carhart, 1997) added one more factor, relative past stock returns, to the model. So the new model can be written as follows:

$$ER_{it} = \alpha_i + \beta_{i,MKT}MKT_t + \beta_{i,SMB}SMB_t + \beta_{i,HML}HML_t + \beta_{i,UMD}UMD_t$$

ER_i	The return on an asset in excess of the one-month T-bill return
MKT	The excess return on market portfolio
SMB	The difference in returns between small-capitalization stocks and large-capitalization stocks
HML	The difference in returns between high-book-to-market stocks and low-book-to-market stocks
UMD	The difference in returns of the prior year’s winners and losers

This factor model has been proved that it can give much better predictions on asset returns than CAPM does. It is extensively used in estimating returns for U.S stocks and has worked well for several years.

2.2 Index Tracking

2.2.1 Genetic Algorithm

Shapcott (Shapcott, 1992) solved the index tracking problem by using genetic algorithms and quadratic programming techniques. He used the genetic algorithm to generate the subsets of companies and quadratic programming to find the proportion that should be invested in each member company. In his project, he used a random assorting recombination (RAR) operator, which assigns weights according to the importance of the assortment. Results were presented for tracking portfolios made up of 20 assets from the FTSE100. He found that the use of RAR and RPL/Framework led to a flexible genetic algorithm, which performed significantly better than random search.

BMC (Beasley, J.E., Meade, N. and Chang. T.J., 2003) formulate a problem with constraints on transaction costs and the number of stocks held in a tracking portfolio, with the aim of tackling index tracking when transaction costs exist. Their approach involved a population heuristic (PH) and reduction tests to diminish the search space. They conducted experiments for five data sets (Hang Seng, DAX 100, FTSE 100, S&P 100 and Nikkei 225) drawn from major world markets. In their experiments, they cut the observation period into two parts: in-sample and out-of-sample. In the first part, they used the genetic algorithm to construct a tracking portfolio, and then they measured the tracking error associated with this tracking portfolio out-of-sample. In their GA model, the fitness of individuals was evaluated with respect to the objective function. Highly fit individuals were given opportunities to reproduce by the crossover operator, with other highly fit individuals. The mutation occurs after the crossover by altering the genes in the string. The algorithm repeats its cycle until a satisfactory solution is found. The average computing time for all of the five data sets was approximately 15.2 minutes. Moreover, they also considered the tracking errors associated with systematic revision and set 7 decision points for each of the data sets, where the tracking portfolio is revised. Computational results show that their work is very solid in tackling the index tracking problem.

Orito, Yamamoto and Yamazaki (Orito, Y., Tamamoto, H. and Yamazaki, G., 2003) adopt the coefficient of determination (CDP) as the measure of fitness between the total return and increasing rates of the stock price index in the market. They aim to find a simple method to select companies from the market. The selection method consists of two steps. The first step uses a heuristic approach to select N companies from the market, and the second step uses genetic algorithm to choose stocks to form the tracking portfolio. Before running the experiments, parameter investigations were implemented in order to set the crossover rate and the mutation rate (P_c, P_m). They tested several combinations and concluded that the best pair was $(0.9, 0.05)$. Furthermore, the algorithm was set to stop after reaching the maximum generations otherwise it goes back to crossover. Then they applied the algorithm on four data sets. The largest data set contained 1356 stocks and the smallest data set contained 457 stocks. Results show that the largest problems solved took 170 hours and 240 hours on a Pentium III (500 MHz), when the maximum generations were set to 2000 and 3000 respectively. The results show that the method produces a near optimal solution of selecting n companies from all of the companies in the market. Especially, the method works well when the increasing rate of stock price index over a period can be viewed as a linear time series data.

Jeurissen and Van Den Berg (Jeurissen, R. and Van Den Berg, J., 2005) use a hybrid genetic algorithm to approach the index tracking problem. In this case, the tracking error was taken to be the measure of fitness, where tracking error was defined as the variance of the difference between the returns of the tracking portfolio and the index. Weights for each stock in the tracking portfolio were assigned according to a genetic algorithm. Their GA approach used a Tournament Selection, where it runs a tournament among a few individuals and selects the winner, a two point crossover operator to ensure that the number of stocks in the tracking portfolios remain the same, and a reverse mutation operator. They also found the best parameters for the initial population, the number of generations, the crossover probability, and the mutation rate, which are 20, 100, and (1, 0) / (0, 1) respectively. Their research find that the performance of the tracking portfolio found is much better than that of randomly selected portfolios and that of low capitalized portfolios which can be derived from the AEX-index.

Lee, Kim and Min (Lee, J.Y., Kim, T.Y. and Min, S., 2005) construct a priority function for each stock and by applying these functions in a simple heuristic, the suitable stocks for the tracking portfolio are chosen. The priority function consists of the weighted sum of trading volume, market capitalization and beta (volatility). Once the stocks have been selected, the associated weights are then determined using a genetic algorithm. In the process of GA, the crossover rate runs from 0.5 to 0.8 and the mutation rate runs from 0.05 to 0.06, the best parameters are not presented. The GA is automatically stopped when there is no improvement over the 1% at the last 5000 trials. Empirical tests were carried out using the Korean KOPSI 200 index. The results strongly suggest that GA process has outstanding advantages over the conventional portfolio mechanism.

Chang, Yang, and Chang (Tun-Jen Chang, Sang-Chin Yang, Kuang-Jung Chang, 2009) introduced a heuristic approach to portfolio optimization problems in different risk measures, semi-variance, mean absolute deviation and variance with skewness by employing genetic algorithm (GA) and compares its performance to a mean–variance model in cardinality constrained efficient frontier. They set the initial population size to 100, and parents are selected by binary tournament. In the tournament, two pools of individuals are generated, each consisting of two individuals randomly drawn from the population. The individuals with the best fitness, one taken from each of the two tournament pools, are chosen to be parents. Then they use a uniform crossover, where the mutual assets are inherited by the child and non-mutual assets have an equal chance of being inherited by the child, to produce new offspring. Mutation is also implemented after the crossover by randomly changing assets of the individuals. The algorithm is stopped when the number of iterations reaches 500. For each iteration, the heuristic evaluates exactly $1000N$ (N is the number of the stocks in an index) solutions. Empirical tests were carried out using the daily historical data collected from the Hang Seng, FTSE and S&P 100 with price data of 33, 93 and 99 assets respectively. They also assigned the tracking portfolios with different cardinalities, initially set to 10 and incremented by 10 each time. Computational results are presented by drawing the efficient frontier of each case. The results show that the investors should include only one third of total assets into the portfolio which outperforms than those contained more assets.

Lin and Liu (Chang-Chun Lin, Yi-Ting Liu., 2008) presented three models based on the Markowitz' model to solve the portfolio selection problems by limiting the transaction cost. They used genetic algorithms to obtain the solutions. Their GA began by generating the initial population and evaluating their fitness with respect to different fitness functions. Then they found the best and the worst individuals. The best individual would be protected and the worst one would be replaced by fitter offspring which is generated by crossover and mutation. An empirical study was carried out using the Taiwanese mutual fund data from the year 1997 to 2000 and monthly rates of return were used instead of weekly ones. They used four data sets, which contained 129, 160, 197, and 204 numbers of funds respectively. In the experiments, the crossover and mutation rate were set to 1 and 0.05, respectively. The algorithms would be stopped after 5000 iterations. Computational results show that the proposed method is valid for the portfolio optimization problem.

Soam, Leon, and Iba (Soam, Palafox, and Iba, 2012) applied the genetic algorithm to the portfolio optimization problem, in which they introduced a new 'greedy coordinate ascent mutation operator' to fine tune the weights of the assets in the portfolios. In their GA work, they first randomly generate the initial population, and then they applied a Deterministic Tournament Selection (DTS) to select the individuals for breeding, as they found that the DTS worked well with small subsets of the population. In the breeding process, they used a K-point crossover technique which randomly chooses K-points in the strings of the parents, and then by alternatively copying elements from the two parents to produce the offspring. The mutation process proceeds instantly after the crossover, which contains two steps. In the first step, they randomly altered the gene values, and in the second step, they altered one particular gene while keeping all the other genes fixed. They used Dow Jones Industrial Average and NASDAQ100, which includes 30 and 100 assets respectively. For the parameters, the number of generations was set to 200 (stopping criteria) and both of the mutation rate and the crossover rate were 0.2. The results achieved by the simulations are very motivating, which show that the portfolio not only performs better than the index and the simple GA, but it also solves the problem of the availability of assets with insignificant weights.

Lin and Gen (Lin, C.M. and Gen, M., 2007) proposed the multistage decision-based genetic algorithm for the multi-objective portfolio optimization problem. In their approach, a random keys-based encoding method was used to generate the initial population, a simple crossover operator was used for producing the offspring, and the insertion mutation method was used to alter the value of the genes. They employed a roulette wheel for the selection. For the parameters, they set the population size to 100, the maximum generation to 1000 (stopping criteria), the crossover probability to 0.7, and the mutation rate to 0.5. The empirical tests were carried out by using 40 sample companies' data from the Taiwan stock market, which show the effectiveness of the proposed algorithm is validated for solving portfolio optimization problem.

Pandari, Azar, and Shavazi (Pandari, A.R., Azar, A., and Shavazi, A.R., 2012) employed the genetic algorithm to form the best portfolio in 50 supreme Tehran Stock Exchange companies. The initial population size was 150 and the maximum generation was set to 250 generations. In the reproduction process, all of the 50 individuals would produce 3 new offspring and only the fittest amongst the three was transferred into the next generation. Furthermore, they set the mutation rate to 0.01, and used the reverse crossover operator to produce offspring, but the crossover rate was not given. The computing results of the GA were compared with those of a Markowitz model. Their study shows that GA model works better than Markowitz's model for selecting stock portfolio. By using this model, one can take into account different dimensions of the reality of the issue, and as a result, achieve more actual responses

Lai, Yu, Wang, and Zhou (Lai, K. K., Yu, L., Wang, S. Y. and Zhou, C. X., 2006) proposed a double-stage genetic optimization algorithm for portfolio selection. In the initial stage, they employed a genetic algorithm to identify good assets in the market, which is measured by the *Return on Capital Employed* (ROCE), *Price-Earnings* (P/E), *Earnings per Share* (EPS), and Liquidity ratios. In the second stage, they used a GA to decide the asset allocation which is based on Markowitz's theory. In their GA approach, the initial generation was created by encoding four input variables, each of them representing the corresponding ratio performance. Additionally, a simple crossover and roulette wheel was used for the reproduction process. However, they did not specify the crossover rate. The mutation was executed by simply altering the values of genes (0 or 1). The mutation rate was set to 0.005. The data used in their study was obtained from the Shanghai Stock Exchange. Simulation was completed with 100 randomly selected stocks. Experimental results reveal that the proposed double-stage genetic optimization algorithm for portfolio selection provides a very feasible and useful tool to assist the investors in planning their investment strategy and constructing their portfolio

2.2.2 Other Approaches

Chen and Kwon (Chen, C. and Kwon, H. R., 2012) developed a robust portfolio selection model for index tracking. They used the binary integer program to maximize the similarity between the selected assets and the target index. In their model, they considered the transaction costs and used a computationally tractable robust framework to protect against the worst-case realizations of potential estimation errors and other deviations. They used the S&P 100 for the out-of-sample test to demonstrate the advantage of the robust model. Computational results of the robust solutions with the cardinalities of 5, 10, 15, 20, 25, and 30 were shown in the end. They concluded that the robust model performed better in terms of beta coefficient, market ratio, and tracking error.

Canakgoz and Beasley (Canakgoz, N. and Beasley, J., 2009) considered index tracking by purchasing a subset of stocks of the index. They proposed mixed-integer programming formulation for the index tracking problem, which includes the transaction costs, cardinality constraints, and the limitations of the total transaction cost. Based on the regression based view of index tracking, they adopted a two-stage approach, where the first objective is to achieve the intercept of zero and the secondary objective is to achieve the slope of one, because they believed that the ideal tracking portfolio would have an alpha (intercept) of zero and a beta (slope) of one. They used a standard solver (Cplex) to solve eight data sets drawn from major markets and the largest data set contains 2151 stocks. Problems of this size are much larger than have been considered previously by other authors in the literature. Computational times are in all cases reasonable.

Wang et al. (Wang, 2012) considered a mixed-integer programming for Index Tracking, in which a CVaR risk constraint was added to the model to control the downside risk of the tracking portfolios. They believed that when the index falls, the tracking portfolio of the index would incur a large downside risk that could incur large losses for the investor, so it is necessary to select a proper risk measure to control the downside risk, and CVaR is a good choice. They used the Hang Seng and FTSE 100 for the numerical tests to show that adding the CVaR constraint has no impact on the tracking portfolio when the market is bullish, while it can limit the downside risk of the tracking portfolio when the market is bearish.

Yao, Zhang and Zhou (Yao, D.D, Zhang, S, and Zhou, X.Y., 2006) considered index tracking by dynamically managing a portfolio consisting of a small number of traded stocks in the market. The problem was formulated as a stochastic linear quadratic control problem and the optimal feedback control was generated by semi-definite programming. A numerical test was carried out by tracking the Hang Seng Index using four large-cap stocks: HSBC Holdings (0005), Hutchison Whampoa (0013), Sun Hung Kai (0016), and China Telecom (0941). Empirical studies show that the SLQ-via-SDP approach is sound and efficient. Also, the examples demonstrate that the tracking performance is independent from the performance of the market and the selection of the stocks.

Okay and Akman (Okay, N. and Akman, U., 2003) proposed a solution approach that used a constraint aggregation (CA) technique to achieve a single constraint to solve the index tracking problem. In their research, they considered the index tracking by using the formulation given by Beasley and colleagues (Beasley, J.E., Meade, N. and Chang. T.J., 2003) and used the CA to transform it to a new formulation with only one inequality constraint. Numerical experiments were implemented using the Hang Seng Index and computational results were compared with those of Beasley. Okay and Akman concluded that the CA technique produced the same solution as that of Beasley in terms of CPU time, total number of branch-and-bound nodes searched, and total number of NLP calls. This paper applies, for the first time, the CA technique to the IT problem, which gives the same solution with appreciably favorable computational results in terms of CPU time, total number of branch-and-bound nodes searched, and total number of NLP calls.

Focardi and Fabozzi (Focardi, S.M. and Fabozzi, F.J., 2004) argued that the clustering methodology is a good candidate for the Index Tracking, which can discover the correlation and integration structure of an index by suitably defining the distances between the time series of assets prices. They believe a suitable distance function that can distinguish time series significantly close to each other or where they depart from each other is the key to the clustering process. Once the distance is clearly defined, the clustering performs hierarchically. It starts with clustering the stocks with the smallest distance, and then clusters the pair of stocks with the smallest distance, and then clusters groups of stocks with the smallest distance etc. The process continues until a single cluster is formed. A numerical test was implemented using the S&P 500 index. The advantage of this approach is that it reduces the difficulties and computational burden of density forecasts and full optimization.

Cowell, El-Hassan, and Kwon (David Colwell, Nadima El-Hassan, and Oh Kang Kwon, 2007) considered the Index Tracking problem as a dynamic hedging problem under the incomplete markets framework. They applied and extended a local risk-minimization approach for contingent claims, whose results were used to build close connections between local risk minimization and *Tracking Error Volatility* (TEV) minimization and reveal their equivalent conditions. By exploiting the connections, they obtained criteria for the selection of optimal tracking portfolios and used a value-at-risk (VaR) type measure to evaluate any given tracking portfolio. The computing results show that it is possible to hedge an entire process by local risk minimization under incomplete markets framework.

Corielli and Marcellino (Francesco Corielli and Massimiliano Marcellino, 2006) proposed a factor based index tracking approach. In their research, they presented a dynamic factor model, where the price of each stock is driven by a set of common and idiosyncratic factors. The solution approach was split into two steps. In the first step, a tracking portfolio was constructed by the same persistent factors as the index. In the second step, the tracking portfolio was refined to minimize the loss function. A numerical test was implemented using the EuroStoxx50 index. In addition, the Monte Carlo simulations were also proved to support the research. Simulation and computational results were shown by comparing them with the results of the Ordinary Least Square (OLS) based index replications. The results are quite encouraging, and emphasize the importance of a statistical approach to index tracking.

Work, other than that discussed above, dealing with index tracking can be found in (Fang, Y. and Wang, S.Y., 2005), (Coleman, T.F., Henninger, J., and Li, Y., 2006), (Yu, L., Zhang, S., and Zhou, X.Y., 2006), (Maringer, 2008), (Ruiz-Torrubiano, R., & Suarez, A., 2009), (Guastaroba, G., & Speranza, M. G., 2012).

2.3 Summary

In this Chapter, we present the historical overview of the portfolio optimization theory and discussed some of the studies related to Index Tracking. We reviewed several different solution approaches for the Index Tracking problem such as: Quadratic Programming, Nonlinear Programming, Mixed Integer Programming, and Genetic Algorithms. It should be noted that the majority of previous works only considered one problem or a very limited number of problems for the empirical experiments. Additionally, the problems were extracted from various markets at different times, and each of them was only solved by one particular solution approach, consequently making it impossible for comparisons. Particularly, we reviewed the previous work of GAs and noticed the following issues:

- The problems concerned are independent from each other, and each problem was usually solved using only one particular approach. Therefore, there is no benchmark to measure the performance of the solution approaches.

- Most of the work only considered a single problem or multiple problems of similar size. In addition, the size of the problems is generally small and the cardinality was fixed.

- The parameters setting investigations only involved crossover probability and mutation rate. There is hardly any research done on the stopping criteria and the initial population size. Moreover, they usually set different problems with the same number of maximum generations (stopping criteria) and population size.

In our research, we would like to address the above issues. We decide to set a benchmark first, and then use the benchmark to measure the performance of the heuristic solution approaches. For the empirical experiments, we consider eight different market indices. The smallest data set includes 31 stocks and the largest data set includes 2151 stocks. We believe that the performance of a solution approach can only be guaranteed after testing it with a variety of problems. Also of interest is to implement investigations on the initial population size and to set stopping criteria based on convergence rather than the number of maximum generations. We believe the initial population size and the stopping criteria play decisive roles in finding reasonably good solutions in reasonable time. In addition, we notice that few researchers have used the ACO algorithm to solve the Index Tracking problem and we believe it is a good candidate.

Chapter 3

3 .Mixed Integer Programming

3.1 Introduction

A Mixed Integer Programming (MIP) is an optimization problem in which some of the variables are restricted to be integers, while others are allowed to be non-integers. The term often refers to Mixed Integer Linear Programming (MILP), where the objective function and the constraints are linear. A mixed integer leaner program in canonical form can be expressed as (Papadimitriou and Steiglitz, 1998):

$$Maxmize \qquad c^T x$$

$$s.t. \qquad Ax \leq b$$

$$x_1, x_2 \in x$$

$$x_1, x_2 \geq 0, \;\; x_2 \;\; integer$$

Where c and b are vectors, and A is a matrix. The above formulation can also be transferred to standard form by introducing a variable s to eliminate the inequalities. In this Chapter, we use the MIP and a standard branch and bound optimization to formulate and solve the Index Tracking problem.

3.2 Research Comparisons

Mixed integer programming can serve as a foundation approach for the Index Tracking problem. BMC (Beasley, J.E., Meade, N. and Chang. T.J., 2003) used the Mixed Integer Programming as a basic approach for their evolutionary heuristic to solve the Index Tracking problem. In their research, they use a historical look-back approach which assumes the past as a guide to the future. They observe the price of N stocks over a certain period of time [0, T] and decide to select the best set of K stocks (K < N) to track the performance of the index over the given time period. By finding the optimal portfolio during the time period [0, T] they deduce what should be held into the future [T, T + L]. Our MIP approach is similar to BMC's. In the following section, we proceed to outline the major differences between the two studies.

- **Transaction cost**

In BMC's paper, they consider the revision of an existing portfolio and the transaction costs which are incurred through buying and selling of stocks. They further set the limit on the total transaction costs that can be incurred. In our approach we do not consider transaction costs as we assume there is no cash inflow or outflow associated with buying and selling. However it does not downgrade the practicability and technicality of our solution approaches as the transaction cost is not an important factor in our research work, and also we could easily reconsider it by simply adding the transaction costs into our constraints.

- **Objective**

There are a range of objectives that can be defined for the Index Tracking problem. BMC are interested in two factors, which are tracking error and excess return. They state an objective that plays a trade-off between them. He defines the tacking error as the following:

$$E = \left[\left[\sum_{t \in S} \left| r_t - R_t \right|^{\alpha} \right]^{1/\alpha} \right] \Big/ T$$

Where

r_t	The single period continuous time return given by the new tracking portfolio at time t
R_t	The single period continuous time return given by the index
T	The ending time point of a historical observation time
S	$S = [t \mid t = 1, 2\ldots \mathrm{T}]$
α	The penalisation parameter

They uses α to penalise the difference between r_t and R_t , and S is the suitable observation period. By comparison, our tacking error is defined as the sum of the under-performance (U_t) and the over-performance (O_t) during the whole observation period.

- **In-sample and Out-of-sample**

In BMC's paper, they state that a value for α determined in-sample does not automatically guarantee a good performance by the same value of α for out-of-sample. Thus he assigns weightings for each observation period with the most recent time receiving the highest weighting. In contrast, our research is more centred towards the solution approach and so the stability of the model is of less importance to us. The MIP only serves as a foundation for our research and based on this we will build more enhanced solution approaches to shorten the computing time while maintaining solution quality. Note that, we do not completely neglect the stability of the model as we implement the out-of-sample performance simulations in the post analysis.

- **Observation Period**

Initially, we use the entire observation period of 290 time periods. However, we quickly found out that if we set the observation period to 50 time units it can still produce good out-of-sample results. Moreover, when we shorten the observation period we can save a lot of computing time. Finally, as our research is mainly focused on the solution approach it is acceptable to shorten the observation periods as they are not a determining factor in the experiment but should still be reasonably controlled. The computational results given in this thesis all use 50 time periods unless otherwise stated.

- **Cardinality**

In BMC's paper, they only limit the number of stocks (cardinality) that can be selected from each index, but they did not vary the cardinality for each index. In our research, we run the programme with different scenarios of cardinalities for each index and try to find out its impact on the tracking error. Table 3-1 shows the data sets of BMC's paper and Table 3-2 shows those of ours.

Table 3-1: BMC's MIP data sets

Index	Number of stocks	Cardinality
Hang Seng	31	10
DAX	85	10
FTSE 100	89	10
S&P 100	98	10
Nikkei 225	225	10
S&P 500	457	40
Russell 2000	1318	90
Russell 3000	2151	70

Table 3-2: MIP data sets of our research

Index	Number of stocks	Cardinality					
Hang Seng	31	10	15				
DAX 100	85	10	15	20			
FTSE 100	89	10	15	20			
S&P 100	98	10	15	20			
Nikkei 225	225	10	15	20	25		
S&P 500	457	10	15	20	25	30	
Russell 2000	1318	30	40	50	60	70	
Russell 3000	2151	30	40	50	60	70	80

3.3 Formulation

For convenience of comparison, we adopt the notation of BMC's (Beasley, J.E., Meade, N. and Chang. T.J., 2003). Our interest is to find the best K ($K < N$) stocks to construct an optimal portfolio and to determine their corresponding weights. The following is the outline of the formulation.

Index

$i = 1 \ldots n$	denotes the number of stocks in an index
$\mathrm{t} = 1 \ldots \mathrm{T}$	is the historical time period

Data

K	The cardinality represents the maximum number of stocks allowed in a portfolio.
Σ_i	The maximum proportion of the tracking portfolio that can be held in stock i ($i = 1 \ldots . n$).
E_i	The minimum proportion of the tracking portfolio that can be held in stock i ($i = 1 \ldots . n$)
I_t	The index value at time t
V_{it}	The value of one unit of stock s ($i = 1 \ldots n$) at time t
R_t	The single period continuous time return of the index, i.e. $R_t = log_e[\frac{I_t}{I_{t-1}}]$
N_{it}	The single period continuous time return of the stock i ($i = 1 \ldots n$), at time t. $N_{it} = log_e[\frac{V_{it}}{V_{it-1}}]$

Decision Variables

ω_i	The proportion of the tracking portfolio that can be held in stock i ($\omega_i \geq 0$)
Z_i	1 if any stock i ($i = 1 \dots n$) is held in the tracking portfolio, 0 otherwise
U_t	The under-performance of the tracking portfolio at time t. ($U_t \geq 0$)
O_t	The over-performance of the tracking portfolio at time t. ($O_t \geq 0$)

Model

Minimize Tracking-Error $$\sum_t (U_t + O_t)$$

Constraints

$$\sum_i Z_i = K \tag{3.1}$$

$$\sum_i \omega_i = 1 \tag{3.2}$$

$$E_i * Z_i \leq \omega_i \quad \forall \text{ i} \tag{3.3}$$

$$\Sigma_i * Z_i \geq \omega_i \quad \forall \text{ i} \tag{3.4}$$

$$R_t = \sum_i N_{it} * \omega_i + O_t - U_t \qquad \forall \text{ t} \tag{3.5}$$

From the above formulation, the cardinality K indicates that our tracking portfolio contains exactly K different stocks. The minimum threshold proportion of a single stock is E_i and Σ_i is the maximum. The objective is to minimize the tracking-error. In this case, it is the under-performance plus the over-performance of the tracking portfolio during the whole observation period that constitutes the tracking-error. The first constraint ensures that there are exactly K stocks in the new tracking portfolio.

The second constraint ensures that we fully invest in the portfolio. The third and fourth constraints state that if a stock i is not in the new tracking portfolio then both Z_i and ω_i are zero, while if stock i is in the new tracking portfolio then ω_i is limited by the proportion thresholds.

3.4 Computational Results

To test the above formulation we use eight market indices, which are the Hang Seng (Hong Kong), the DAX 100 (Germany), the FTSE 100 (UK), the S&P 100 (USA), the Nikkei 225 (Japan), the S&P 500 (USA), the Russell 2000 (USA) and the Russell 3000 (USA). Stock price data was originally obtained from DATASTREAM, stocks with missing values were dropped and for each index we had 290 time units (weekly) stocks returns. We get the data from John Beasley's OR library which is publicly accessible. For each index we have 50 observation periods. The computational results presented in this paper are coded in AMPL-64 bit and solved by *Cplex* on a desktop that has the following system:

- Processor: Intel ® Core ™ i5-2500 CPU @3.30GHz 3.30 GHz
- Installed memory (RAM): 16.0 GB
- System type: 64-bit Operating System.

Due to the size of the problems, optimal solutions cannot be produced within a reasonable computational time for all indices except the Hang Seng. For example, it takes almost 10 days to produce the optimal solution for the DAX-100, with such long computing time making the index-tracking meaningless. Considering the above issues, we decided to limit the computing time to 2 hours. Table 3-3 shows the computing results and we set these results as the bench mark to measure the results produced by our other solution approaches. K, TE, and CT represent cardinality, tracking error, and computing time respectively. Except the Hang Seng index, MIP is not able to find optimal solutions of the other indices in a reasonable time. Thus, we develop two heuristic methods to solve the Index Tracking problem, which are

Genetic Algorithms and Ant Colony Optimization. The core aim of developing these two heuristics is to find reasonably good solutions in a reasonable time.

Table 3-3: Computational results of MIP

Index	K	TE	CT
Hang Seng	10	0.101010742	2.44
	15	0.056886088	0.583
DAX 100	10	0.067296843	7209.49
	15	0.037275415	7221.324
	20	0.019241806	7203.913
FTSE 100	10	0.120696147	7312.577
	15	0.055322442	7298.064
	20	0.033403056	7339.73
S&P 100	10	0.106237693	7203.85
	15	0.047551997	7225.657
	20	0.030086608	7207.366
Nikkei 225	10	0.087352372	7403.56
	15	0.048647172	7389.435
	20	0.026986383	7503.684
	25	0.014928127	7447.022
S&P 500	10	0.088660628	7260.256
	15	0.056483628	7268.022
	20	0.027066374	7342.924
	25	0.015999239	7341.46
	30	0.006709845	7337.536
Russell 2000	30	0.016186968	7250.086
	40	0.005202331	7257.58
	50	0.000185917	7274.063
	60	0	240.11
	70	0	145.868
Russell 3000	30	0.016114459	7204.806
	40	0.004705511	7200.741
	50	0.000508717	7205.823
	60	0	1555.994
	70	0	143.835
	80	0	531.07

Chapter 4

4 .Genetic Algorithms

4.1 Introduction

John Holland (Holland, 1975) first invented Genetic algorithms (GAs) in the 1960s. It was further developed by him, his students, and colleagues at the University of Michigan in the 1960s and 1970s. GA is a search heuristic that is derived from the process of natural evolution and is routinely used to generate useful solutions to optimization and searching problems. Many computational problems need to search for a desired solution from a large number of possibilities (candidate solutions) that is usually called searching in a 'search space' in computer science. In biology, the set of possible genetic sequences is the various set of possibilities and highly fit organisms can be seen as the desired solutions. Evolution is a method of searching solutions from various numbers of possibilities, and it is indeed a very efficient tool to address such searching problems, as it allows the searching to be carried out simultaneously in an efficient way. The evolution usually starts from the initial generation which is a population of randomly generated individuals. In each generation, the fitness of the individuals is evaluated. Multiple individuals are selected from the current generation and modified (recombined or mutated) to form a new generation. The new generation is then used in the next iteration of the algorithm. Normally, the algorithm terminates when either a maximum number of generations is reached or the fittest individual is found. Typically, genetic algorithms contain at least the following fundamental elements: population of chromosomes, selection according to fitness, crossover to produce new offspring, and random mutation of new offspring (Mitchell, 1998).

4.2 Application for Index Tracking

Till date, Genetic Algorithm has been applied to a wide range of optimization problems. It has become a very popular optimization method because it is significantly advantageous when used to find the best solution by global search in contrast to most common optimization algorithms. To solve Index Tracking problems, researchers often use binary to encode stocks information in tracking portfolios. For example, to select 10 stocks from the Hang Seng index consisting of 31 stocks, each portfolio string (chromosome) can be represented by a binary string including 31 bits, and 10 of which are equal to 1. However, this encoding method is only feasible when dealing with small size indices, when it deals with large size indices, the information processed by the GAs would be too overwhelming. In our research, we deem each stock as a gene, each tracking portfolio as an individual, and the fitness value of each tracking portfolio is the reciprocal of its tracking error. This section presents the details of our GA work.

- **Initial Population**

In our research, the creation of the initial population is carried out in a random manner, whereby we randomly select K different stocks from the index and combine them to form a tracking portfolio. By repeating the above selection N times, we are able to create the initial generation. The number of times that we use the selection operator is vital in searching for the optimal solution, because it determines the size of the initial population. In our research, one of our main goals is to find for a desirable initial population size for each index so that we can balance the computing time and the solution quality.

- **Crossover and Reproduction**

The crossover operator is often used to produce new offspring. Typically, it randomly chooses a locus and exchanges the subsequence of chromosomes before or after that locus to create two offspring. Figure 4.1 shows an example.

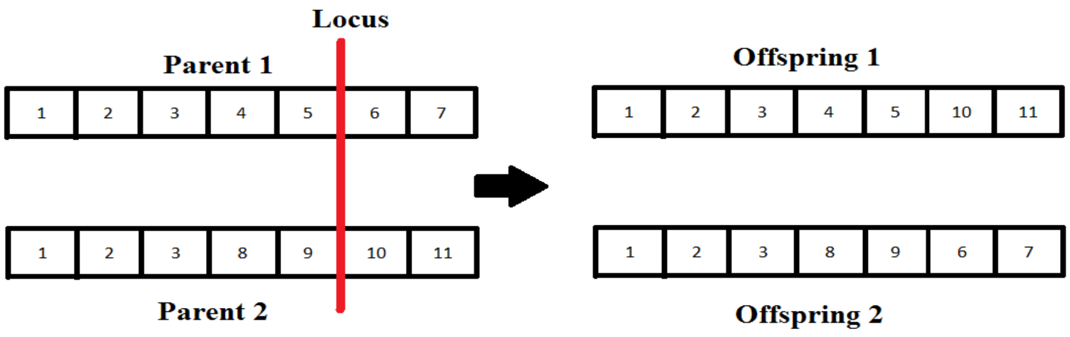


Figure 4.1: Crossover

However, in a simple crossover, the crossover position could be located at any position on the parent portfolio strings. Hence, it is very likely to produce deformed offspring. Figure 4.2 shows an example.

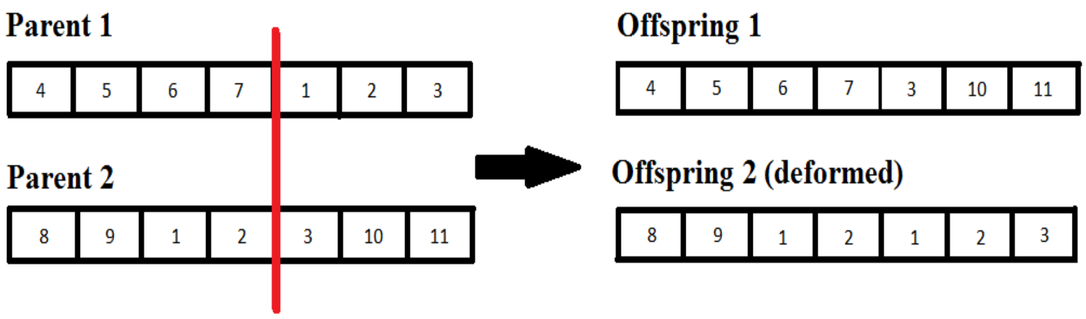


Figure 4.2: Deformed offspring

In the above figure, the crossover position is located at the fourth position on both of the parent portfolios. After exchanging their genetic pieces, they produce one deformed offspring that contains two duplicating stocks. To meet the cardinality constraint (3.1), we have to replace the duplicates with other stocks to ensure the offspring contains exactly K stocks.

Regarding the above issue, we use a semi-optimization approach to produce new offspring. In our approach, parent portfolios only produce one offspring and this offspring automatically inherits the mutual bits of chromosomes from the parents. The selection of the rest of the chromosomes relies on the optimization. Figure 4.3

shows an example of the semi-optimization approach. Parent 1 and Parent 2 have mutual stocks of 1, 2, and 3, so their offspring directly inherit these mutual stocks. For the other four stocks, they will be chosen by the solver from the non-mutual stocks (4, 5, 6, 7, 8, 9, 10, 11), and together with the mutual stocks they will form an offspring. The semi-optimization approach not only avoids the deformation issue but also ensures that the offspring is at least as fit as the fitter one of the parents. In contrast, the crossover operator cannot guarantee superiority of the offspring.

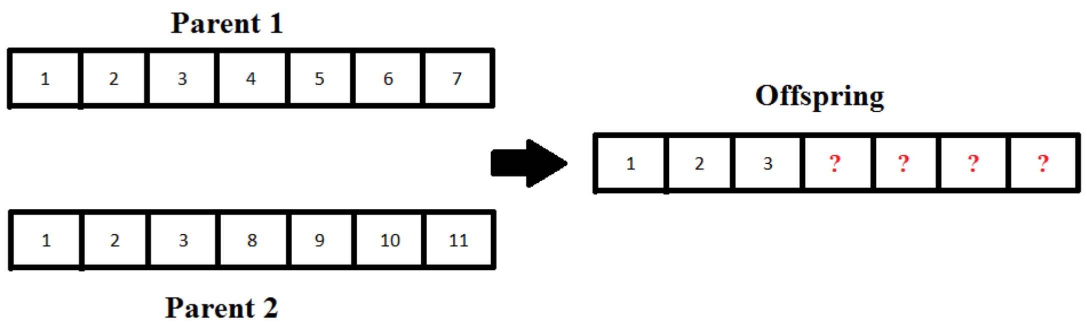


Figure 4.3: Semi-optimization approach

After producing the offspring, we can use them to either replace the whole preceding generation or replace only the unfit individuals in the previous generation. The former approach is called the generational approach and the latter one is called the steady-state-approach.

- **Mutation.**

Before introducing mutation, it is necessary to mention an important concept known as ‘fitness landscape’, which was originally defined by the biologist Sewell Wright (Wright, 1984). It represents the space of all possible genotypes (Genotypes refers to the particular set of genes in a genome) along with their fitness. Wright assumes that each genotype is a part of the string of chromosomes and can be assigned with a value of fitness. If the length of the genotype is l, then the fitness landscape can be seen as a $'l+1'$dimensional plot and each genotype is a point in l dimension and its fitness is plotted along the $‘l+1’$ axis. For simplicity, suppose each tracking portfolio only contains two stocks, then we have a plot with $l=2$ dimension landscape. The fitness values of possible portfolio combinations form ‘hills’ and ‘valleys’ on the landscape. As it shown in figure 4.4, each point on the $x-y$ surface represents a possible portfolio combination, the Z axis represents the fitness axis, and the fitness landscape formed is in grey. (Notably, as duplicate stocks are not allowed in a tracking portfolio, combinations such as (6, 6), (7, 7) and (8, 8) are not assigned with any fitness values. Also, combinations such as (6, 7) and (7, 6) have the same fitness value as the sequence of stocks has zero influence on tracking portfolios.)

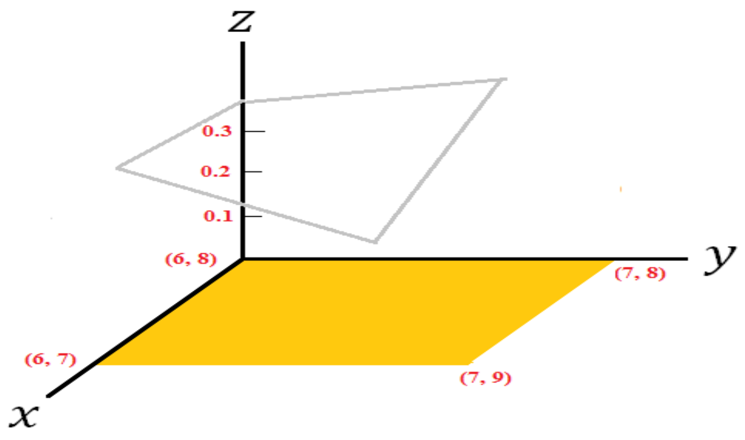


Figure 4.4: Fitness landscape

The evolution drives the populations to move along the landscape in a certain way. During the evolutionary process, the movement of populations is very likely to be directed toward local 'hills' (local optimum). This is commonly caused by the loss of useful chromosomes. In order to maintain genetic diversity in the populations, it is necessary to introduce mutation. By mutation, it is similar to a random walk on the fitness landscape to enable the evolution of a certain probability to jump out of the local optimum. Conventionally, mutation is operated by flipping some of the bits in a chromosome. Figure 4.5 shows an example.

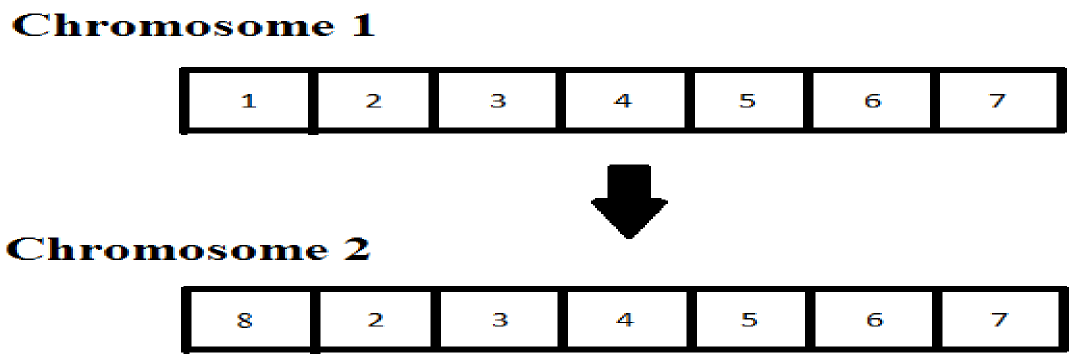


Figure 4.5: Traditional mutation

The string 1234567 in chromosome 1 has mutated from its initial position to yield 8234567, chromosome 2. This kind of mutation can occur at any position in a chromosome with a certain probability, usually very small. The flipping mutation certainly can bring back the eliminated chromosome bits and preserve diversity in a generation. However, it has more negative influence than positive influence on the fitness value of an individual, because it often breaks the bonds of stocks in portfolios that make the mutants less fit than the originals. In order to keep the mutant as fit as its original, we develop a new way of mutation. In our approach, with probability P_m we impose mutation on individuals by providing extra bits of chromosome (usually the number of extra bits is set to 5). In the index tracking problem, the extra bits are selected from a specific set which is defined as the follows:

P_j	Represents portfolio *j*
A	Represents all the assets in an index
B	Represents the special set from which we select extra different genes for each portfolio

$$B = \overline{P_j} \cap A$$

Once the extra bits are provided to each individual, we let the solver sort out the new optimal combination that can be defined as $\widehat{\mathrm{P}}_{\mathrm{j}}$.

In the following sections, we present two GA approaches to solve the Index Tracking problem, which are named the Inverse Triangle GA and the Roulette GA. The computing results for the two approaches are also presented. At the end of the section, we compare the two approaches and summarize a conclusion on the findings.

4.3 The Inverse Triangle Genetic Algorithm

4.3.1 Introduction

The name Inverse Triangle genetic algorithm (IT-GA) is derived from its evolutionary pattern, which looks like an inverse triangle. In the IT-GA, we randomly generate a number of individuals and evaluate the fitness of each individual, sort the initial generation from the fittest to the least fit, and then select the top 30% (this number is user-specified) to form the first generation, see figure 4.6.

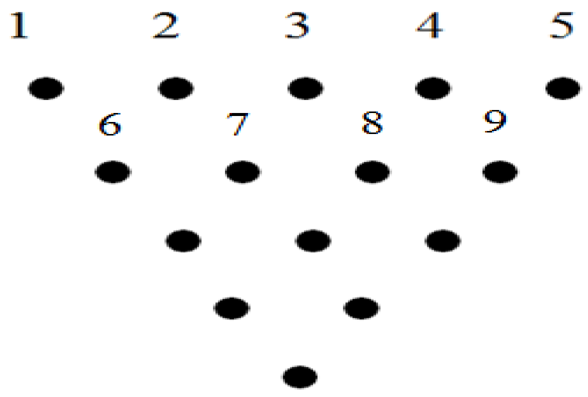


Figure 4.6: Inverse triangle evolution

In the above figure, each black dot represents a portfolio. We assume that the first generation is composed by portfolios 1, 2, 3, 4, and 5. The reproduction process in the first generation consists of 4 mating pairs: ‘12’, ‘23’, ‘34’, and ‘45’. Therefore, the parent portfolios together will produce 4 offspring, 6, 7, 8, and 9. Subsequently, the four offspring change roles to serve as the parents and repeat the previous mating process. Following this fashion, the evolution goes on for several generations until there is only one individual left which is potentially the fittest individual achievable. The IT-GA is different from any previous GAs from two aspects:

1. It continuously narrows down the searching space until there is only one individual left.
2. It allows us to approximately calculate the computational time even before the evolution starts.

4.3.2 Formulation

The following is the pseudo code of the IT-GA. It is transformed from the formulation of the MIP with several changes, see the details below.

Index

$For\ J = 1 \ldots j - 1$

P_j	Represents Portfolio j
A	$P_j \cup P_{j+1}$
I	$P_j \cap P_{j+1}$
D	$A \cap I$

Model

Minimize Tracking Error $\sum_t (U_t + O_t)$

Subject

$$\sum_{d \in D} Z_d + |I| = K \tag{4.1}$$

$$\sum_{a \in A} \omega_a = 1 \tag{4.2}$$

$$E_i \le \omega_i \le \Sigma_i \qquad \forall\, i \in I \tag{4.3}$$

$$E_d * Z_d \le \omega_d \qquad \forall\, d \in D \tag{4.4}$$

$$\Sigma_d * Z_d \ge \omega_d \qquad \forall\, d \in D \tag{4.5}$$

$$R_t = \sum_{a \in A} N_{at} * \omega_a + O_t - U_t \qquad \forall\, t \tag{4.6}$$

P_j *and* P_{j+1} represent two neighboring portfolios that serve as parent portfolios. 'A' represents the set of P_j *union* P_{j+1}. 'I' represents the mutual stocks of the parent portfolios and D represents the non-mutual stocks. Equation (4.1) means that a new offspring is composed by the mutual stocks inherited from the parents, and a selected number of the non-mutual stocks.

4.3.3 Simulation

We illustrate the mechanism of the Inverse Triangle GA by simulation. In the simulation, we generate 20 portfolios (each portfolio contains 4 different stocks) and select the top 10 fit ones to form the first generation. Subsequently, the semi-optimization operator is used to produce the offspring. The following shows the basic steps of the IT-GA:

- *Generate an initial population*
- *Evaluate the fitness of individuals in the initial generation*
- *Sort them from the fittest to the least fit*
- *Select the top 30% (user-specified) to form the first generation*
- ***Repeat***
  - *Mate the neighbouring individuals to produce offspring*
  - *Sort the offspring from the fittest to the least fit*
  - *With probability* $P_m \leq 0.02$ *implement mutation*
- ***Until***
  - *We find the fittest offspring*

The data set used to implement the simulation is from the Hang Seng index. For simplicity, we decrease the number of stocks of the index to 10 and set the cardinality to 4. We only present the simulation to the fourth generation, as the rest of the generations can be easily deduced. Note here that, the relation between the tracking error and the fitness value is given by:

$$f_i = 1/f(x)$$

The fitness value is the reciprocal of the tracking error, which means that the smaller the tracking error the fitter the individual. Table 4-1 shows the simulation results.

Table 4-1: IT-GA simulation

Initial population (Portfolio No.)	Stocks in Portfolio				Tracking error $f(x)$	Fitness value fi
1	4	9	7	8	0.564493	1.771501152
2	2	7	9	3	0.566167	1.766263311
3	3	1	9	4	0.567521	1.762049334
4	2	4	1	7	0.570928	1.751534344
5	5	6	8	7	0.571562	1.74959147
6	5	9	7	2	0.574703	1.740029198
7	1	2	5	4	0.579622	1.725262326
8	10	6	9	7	0.588624	1.698877382
9	7	1	4	3	0.595866	1.678229669
10	5	2	6	10	0.603175	1.657893646
11	5	4	1	3	0.607775	1.645345728
12	1	9	8	5	0.612161	1.633557185
13	2	5	8	6	0.615245	1.625368756
14	5	10	2	4	0.616589	1.621825884
15	8	1	2	9	0.622338	1.606843869
16	5	1	6	2	0.622885	1.605432785
17	5	1	7	4	0.625538	1.598623905
18	2	10	9	1	0.629814	1.587770358
19	8	1	4	2	0.635992	1.572346822
20	2	4	10	1	0.638456	1.566278647

First Generation (Portfolio No.)	Stocks in Portfolio				Tracking error $f(x)$	Fitness value fi	offspring (stocks in Portfolio)				Tracking error $f(x)$	Fitness value fi
1	4	9	7	8	0.564493	1.771501	9	7	4	2	0.53126	1.882318
2	2	7	9	3	0.566167	1.766263	9	3	2	4	0.515001	1.941744
3	3	1	9	4	0.567521	1.762049	1	4	9	2	0.51498	1.941823
4	2	4	1	7	0.570928	1.751534	7	2	4	6	0.478501	2.08986
5	5	6	8	7	0.571562	1.749591	5	7	6	9	0.569804	1.754989
6	5	9	7	2	0.574703	1.740029	5	2	9	4	0.435712	2.295094
7	1	2	5	4	0.579622	1.725262	5	4	6	9	0.416266	2.40231
8	10	6	9	7	0.588624	1.698877	7	6	4	3	0.430683	2.321893
9	7	1	4	3	0.595866	1.67823	1	3	2	6	0.410773	2.434435
10	5	2	6	10	0.603175	1.657894					Ave	2.118274
					Ave	1.730123					Max	2.434435
					Max	1.771501						

Sorted

Second Generation (Portfolio No.)	Stocks in Portfolio				Tracking error $f(x)$	Fitness value fi	offspring (stocks in Portfolio)				Tracking error f(x)	Fitness value fi
11	1	3	2	6	0.410773	2.434435	1	3	2	6	0.410773	2.434435
12	5	4	6	9	0.416266	2.40231	5	4	6	9	0.416266	2.40231
13	7	6	4	3	0.430683	2.321893	5	4	6	9	0.416266	2.40231
14	5	2	9	4	0.435712	2.295094	2	4	9	6	0.431132	2.319475
15	7	2	4	6	0.478501	2.08986	2	4	9	6	0.431132	2.319475
16	1	4	9	2	0.51498	1.941823	1	4	9	2	0.51498	1.941823
17	9	3	2	4	0.515001	1.941744	9	3	2	4	0.515001	1.941744
18	9	7	4	2	0.53126	1.882318	9	7	4	6	0.465436	2.148523
19	5	7	6	9	0.569804	1.754989					Ave	2.238762
					Ave	2.118274					Max	2.434435
					Max	2.434435						

Sorted

Third Generation (Portfolio No.)	Stocks in Portfolio				Tracking error $f(x)$	Fitness value fi
11	1	3	2	6	0.410773	2.434435
12	5	4	6	9	0.416266	2.40231
12	5	4	6	9	0.416266	2.40231
20	2	4	9	6	0.431132	2.319475
20	2	4	9	6	0.431132	2.319475
21	9	7	4	6	0.465436	2.148523
16	1	4	9	2	0.51498	1.941823
17	9	3	2	4	0.515001	1.941744
					Ave	2.238762
					Max	2.434435

offspring (stocks in Portfolio)				Tracking error $f(x)$	Fitness value fi
1	3	2	6	0.410773	2.434435
5	4	6	9	0.416266	2.40231
5	4	6	9	0.416266	2.40231
2	4	9	6	0.431132	2.319475
2	4	9	6	0.431132	2.319475
2	4	9	6	0.431132	2.319475
1	4	9	2	0.51498	1.941823
				Ave	2.305615
				Max	2.434435

Fourth Generation (Portfolio No.)	Stocks in Portfolio				Tracking error $f(x)$	Fitness value fi
11	1	3	2	6	0.410773	2.434435
12	5	4	6	9	0.416266	2.40231
13	5	4	6	9	0.416266	2.40231
20	2	4	9	6	0.431132	2.319475
20	2	4	9	6	0.431132	2.319475
20	2	4	9	6	0.431132	2.319475
16	1	4	9	2	0.51498	1.941823
					Ave	2.305615
					Max	2.434435

From the above simulation, we notice that the average fitness value gradually increases from 1.73 to 2.31, which means the fitness level of the whole population is improved after each generation. The maximum fitness value is 1.77 in the first generation and it increases to 2.43 in the second generation and remains the same till the last generation. By further observation, we find that some duplicate portfolios appear in the evolution, such as portfolio 12 and portfolio 20 in the third generation. These duplicate portfolios are redundant individuals in an evolution and should be removed, examples are shown below.

Generation										
1	1	2	3	4	5	6	7	8	9	10
2	11	12	13	14	15	16	17	18	19	
3	11	12	12	20	20	21	16	17		
4	11	12	12	20	20	20	16			
5	11	12	12	20	20	20				
6	11	12	12	20	20					
7	11	12	12	20						
8	11	12	12							
9	11	12								
10	11									

Figure 4.7: Evolution with redundant duplicates

Figure 4.7 shows the whole evolution process for the simulation. Except for the first generation, the whole evolution produces 45 offspring in total. Each offspring is associated with a 'solve' command. As stated earlier, we use the *Total Solve Elapsed Time* as our computing time, which is the sum of the elapsed seconds used by all the 'solve' commands. So the total computing time can be affected by the duplicate members. Figure 4.8 shows the transformed evolution after removing the duplicates in each generation.

Generation										
1	1	2	3	4	5	6	7	8	9	10
2	11	12	13	14	15	16	17	18	19	
3	11	12	20	21	16	17				
4	11	12	20	16						
5	11	12	20							
6	11	12								
7	11									

Figure 4.8: Transformed evolution

After removing the duplicates, the evolution only has 7 generations and produces 25 offspring. Besides the removal of duplicates, we could possibly further shorten the computing time. As we stated before, the maximum fitness value remains unchanged from the second generation till the end. Thus, we could introduce stopping criteria to terminate the algorithm before it reaches the 10th generation. Our stopping criteria can test the maximum fitness value in each generation. If the value is unchanged for a specified number of generations, the evolution would be stopped. The following shows the expression of the stopping criteria.

g	Represents generation
$Best_g$	Represents the best portfolio in generation g
P_1^g	Represents the first portfolio in generation g
C	A counter

```
let $\mathrm{Best}_g = P_1^g$
        $if \quad Best_g = Best_{g-1} \ then \quad C = \mathrm{C} + 1$
        else $\quad \mathrm{C} = 0$
```

Because each generation has been sorted, thus P_1^g is the fittest offspring in the gth generation. If $Best_g = Best_{g-1}$then it means that two consecutive generations have the same maximum fitness value. If this situation goes on for 3 generations, we would declare that we would force the programming to stop and report the best solution encountered. We name the Inverse Triangle GA with removing duplicates and stopping criteria as the ‘Enhanced Inverse Triangle GA’ (EIT-GA).

4.3.4 Computational Results

We apply the IT-GA and EIT-GA to the real market indices to further prove the findings of the simulation. We generate different size of initial populations for each index and test them by the first five indices (Hang Seng, DAX, FTSE, S&P 100, and Nikkei 225). Table 4-2 shows the size of the initial populations generated.

Table 4-2: Initial population size for each index

Index	Cardinality		
	$K = 10$	$K = 15$	$K = 20$
Hang Seng	50	50	
DAX	200, 400, 600, 800, 1000	200, 400, 600, 800	200, 400, 600, 800, 1000
FTSE	200, 400, 600, 800	200, 400, 600, 800	200, 400, 600, 800
S&P	200, 400, 600, 800	200, 400, 600, 800	200, 400, 600, 800
Nikkei	400, 800	400	400

Table 4-3 presents the comparison of the computing results of the MIP, the IT-GA and the EIT-GA. Note here, we only show the best solutions achieved from each index (PZ represents the population size). For convenience of readability, we round the tracking error and the computing time to the fifth decimal place and the first decimal place respectively.

Table 4-3: Computing results of MIP, IT-GA, and EIT-GA for the first five indices

Index	K	Solution Approach							
		MIP		IT-GA			EIT-GA		
		TE	CT	TE	CT	PZ	TE	CT	PZ
Hang Seng	10	0.10101	2.4	0.10101	20.6	50	0.10101	16.5	50
	15	0.05689	0.6	0.05689	18.4	50	0.05689	13.9	50
DAX	10	0.06730	7209.5	0.06730	5551.8	1000	0.06730	617.8	1000
	15	0.03728	7221.3	0.03486	521.5	200	0.03486	271.9	200
	20	0.01924	7203.9	0.01924	2130.7	200	0.01924	1036.7	200
FTSE	10	0.12070	7312.6	0.09982	2621.6	600	0.09982	615.1	600
	15	0.05532	7298.1	0.05436	988.4	200	0.05436	401.7	200
	20	0.03340	7339.7	0.02858	7531.3	400	0.02858	2359.7	400
S&P 100	10	0.10624	7203.9	0.09629	1205.6	400	0.09629	437.2	400
	15	0.04755	7225.7	0.04755	15853.4	800	0.04755	2102.3	800
	20	0.03009	7207.4	0.02789	8348.3	600	0.02789	3200.4	600
Nikkei 225	10	0.08735	7403.6	0.08581	9036.8	800	0.08581	1748.2	800
	15	0.04865	7389.4	0.04477	8798	400	0.04477	3721.6	400
	20	0.02699	7503.7	0.02372	36515.6	400	0.02372	17595.8	400

In the case of Hang Seng, DAX, and FTSE, the computing results of the IT-GA are better than the corresponding results of MIP. However, the total computing time is increased by about 11620 time units, which is caused by large indices such as S&P 100 and Nikkei-225, especially when the cardinality is big as well. The situation is improved when we use the EIT-GA. The solution qualities maintain the same whilst each computing time is shortened dramatically. For example, the computing time of S&P-100 with cardinality of 15 (S&P 100-15) is shortened by 13751 time units and the computing time of Nikkei 225-10 is shortened by 7288 time units. The total computing time of the five indices is decreased by 65003.6 time units. Nevertheless, the computing time for the Nikkei 225-20 still goes beyond the limitation. We observe that it is caused by the reproduction. As the cardinality becomes bigger, it takes more computing time for the solver to find the optimal combination. Therefore, we restrict the ‘single solve time’ to 1 time for Nikkei and S&P-500 and to 0.5 units for Russell-2000 and Russell-3000. Table 4-4 shows the computing results for the Nikkei (bigger cardinalities), S&P-500, Russell-2000, and Russell-3000. Notably, we want to achieve good solutions whilst control the computing time, so we have to carefully set the initial population size. (All of the below initial population size are set based on experience.) Table 4-4 shows the computational results.

Table 4-4: Computational results of EIT-GA and MIP for the larger indices

Index	K	MIP		EIT-GA		
		TE	CT	TE	CT	PZ
Nikkei	20	0.02699	7504	0.02512	3312	400
	25	0.01493	7447	0.00913	3725	400
S&P 500	10	0.08866	7260	0.06296	607	400
	15	0.05648	7268	0.03989	1854	400
	20	0.02707	7343	0.02166	2233	400
	25	0.01600	7342	0.01157	1933	400
	30	0.00671	7338	0.00640	3442	400
Russell 2000	30	0.01619	7250	0.01214	1969	600
	40	0.00520	7258	0.00490	476	600
	50	0.00019	7274	0.00116	3650	600
	60	0	240	0	277	600
	70	0	146	0	286	600
Russell 3000	30	0.01611	7205	0.01449	477	600
	40	0.00471	7201	0.00455	240	600
	50	0.00051	7206	0.00039	1703	600
	60	0	1556	0	277	600
	70	0	144	0	389	600
	80	0	531	0	482	600

There are five instances where the MIP and the EIT-GA produce the same solutions. Apart from that, the computing time for EIT-GA is always faster than MIP. The total computing time decreased by about 70178 time units. Therefore, we conclude that the EIT-GA is better than the MIP and it is able to find reasonably good solutions in reasonable time. However, we could not construct solid relations amongst the population size, the size of indices, and the cardinality size. Therefore, we cannot immediately know the suitable size of initial population for a given problem to help the algorithm produce fairly good solution.

With the aim of constructing reliable relations among the three factors while maintaining good computing results, we develop another GA solution approach based on the Schema Theorem.

4.4 Roulette Genetic Algorithms

4.4.1 Schema Theorem

Although the mechanics of Genetic Algorithms are surprisingly simple to describe, which usually involve nothing complex other than copying chromosomes and exchanging bits of chromosomes, the reason why it works is very subtle and complicated. There are generally two questions about GAs: how do GAs work, and what kind of problems are they good for? According to the traditional theory of GAs, good solutions generally tend to be made up of short, low-order, highly-fit 'building blocks'—particular bits of strings. Instead of obtaining good solutions by trying each possible combination, the work of GAs is to discover, emphasise, and recombine the good building blocks to construct even more highly fit and higher-order building blocks in a parallel fashion. The ability to produce fitter solutions by recombining good building blocks is considered to be the main source of the GA's search power. Holland (Holland, 1975) first introduces the notion of schema to formalise the notion of building blocks. A schema is a template describing a subset of strings with similarities. It is composed of ones, zeroes, and asterisks which are the wild cards meanings *'don't care'*. The Schema Theorem is very important and it is considered to be the fundamental theorem of genetic algorithm.

In this section, we first introduce the theoretical foundations of GAs: the schema theorem. Following, we detail how we apply this theorem to the Index Tracking problem. Before the introduction of the schema theorem, we need to present some notation. For convenience, some of the notations are adopted from Goldberg's (Goldberg, 1989).

$o(H)$	Represents the order of a schema H, which is simply the number of fixed positions in a schema. ($o(111 ** 0) = 4, o(0000 * 1) = 5$)
$\delta(H)$	Represents the defining length of a schema H, which is the distance between the first and last specific string position. ($\delta(111 ** 0) = 5$)
$m(H,t)$	Represents that there are m examples of a particular schema H contained within the population $A(t)$ at a given time t
$f(H)$	The average fitness of the strings representing schema H
f_i	The fitness value of string A_i
$\bar{f}$	The average fitness of the entire population
P_i	The probability that string A_i gets selected during reproduction
P_d	The probability of schema H being destroyed under simple crossover
P_s	The probability of schema H surviving under simple crossover
P_c	The probability of crossover
P_m	The probability of mutation
n	The population size
M	The number of mutual stocks

D The number of non-mutual stocks

The operation of GAs is very straightforward. It starts with randomly generating a population of strings, and then it explicitly manipulates the strings: copies strings with the bias toward fitness, mates and swaps bits of strings, and occasionally mutates a part of a string. Although it operates on a population of strings in such straightforward manner, it actually implicitly processes the fitness of much more schemas during each generation. To analyse the operation, we assume that a string is copied regarding its fitness value during the reproduction process, or more precisely it is selected by the probability $P_i = f_i / \sum f_j$. Accordingly, the expected number of the representatives of schema H at time $t + 1$ can be given by the following equation:

$$m(H, t + 1) = m(H, t) * n * f(H) \Big/ \sum f_j \tag{4.7}$$

Alternatively, the above equation can be written as:

$$m(H, t + 1) = m(H, t) * f(H) / \bar{f} \tag{4.8}$$

The above equations show that a schema with a fitness value above the population average would be increased in the next generation, those below the population average would be decreased in the next generation and will gradually die out during the evolution. If we assume that the fitness value of a schema is above average with an amount $c * \bar{f}$, where c is constant, then equation (4.8) can be transformed as follows:

$$m(H, t + 1) = (1 + c) * m(H, t) \tag{4.9}$$

If time starts at $t = 0$, then we can achieve the following expression:

$$m(H, t) = m(H, 0) * (1 + c)^t \tag{4.10}$$

The above expression shows that for the fitness values of a schema above population average, they would be sampled under reproduction at an exponential rate. In contrast, for those below population average, the number would be decreased exponentially in future generations. Based on the above findings, we put the crossover under consideration. To investigate how the crossover influences on a schema, we need to consider the following example:

$$A = 0011|10$$

$$H_1 = 0 *** | * 0$$

$$H_2 =** 11| **$$

String ‘A’ is an example of the schema $H_1\, and\, H_2$ with six fixed positions. Assuming that the crossover position takes place on the fifth locus, where we use the symbol ‘|’ as the crossing position, it is clear that schema H_1 is destroyed as one ‘0’ in the first position and the other ‘0’ in the last position are separated. On the other hand schema H_2 survives as the crossover position cannot separate the ‘11’. Hence, if the crossover position is randomly located on schema H with length l, then the survival probability is given by the following equation:

$$P_s = 1 - \frac{\delta(H)}{l-1} \tag{4.11}$$

If the crossover is operated occasionally with probability P_c, the survival probability can be expressed as:

$$P_s \geq 1 - P_c * \frac{\delta(H)}{l-1} \tag{4.12}$$

Assume that the reproduction and crossover are two independent events. Thus, the expected number of the representatives of schema H at time $t+1$ can be given by the following equation:

$$m(H, t+1) \geq m(H, t) * \frac{f(H)}{\bar{f}} * \left[1 - P_c * \frac{\delta(H)}{l-1}\right] \tag{4.13}$$

The last factor we need to consider is the mutation operator. A schema H that can survive from mutation is only when each of its fixed positions survives. Therefore, the probability of surviving from mutation is $(1 - P_m)^{o(H)}$. As the probability of mutation is usually very small, the survival probability can be written as $1 - o(H) *$

P_m. Therefore, the number of samples of schema H in the next generation can be given by:

$$m(H,t+1) \geq m(H,t) * \frac{f(H)}{\bar{f}} * \left[1 - P_c * \frac{\delta(H)}{l-1} - o(H) * P_m\right] \tag{4.14}$$

In our research, we define each tracking portfolio as a string and the cardinality (K) is the length of a string. However, we are not able to directly apply the Schema Theorem to the index tracking problem, because the sequence of stocks in a tracking portfolio has no influence on its tracking error, so a particular combination of stocks can be arranged in several patterns (e.g., ‘**456123**’, ‘**145263**’, and ‘**124563**’ are the same). On the other hand, the sequence of zeros, ones, and asterisks determine the fitness of a schema. Thus, it is impossible to investigate the growth or decay of combinations, but we can investigate on a particular stock to calculate its survival or death probability in future generations. What we believe is that if a stock is an element of a good ‘building block’ then its quantity would be increased in the future generations, otherwise its quantity would be decreased. In our research, we denote that a ‘building block’ is the mutual stocks of tracking portfolios. A good ‘building block’ can give positive influence on the fitness of tracking portfolios. Generally speaking, the larger the size of a good ‘building block’ the bigger the influence it can impose on tracking portfolios. As ‘building blocks’ is recognized as the mutual stocks of the parent portfolios, they are very subjective and can change from time to time. For example, assume that portfolio A and portfolio B share mutual stocks 1, 2, and 3, but portfolio A and portfolio C share mutual stocks 6, 7, 8, and 9. Once portfolio A changes its mating partner from portfolio B to portfolio C, the ‘building block’ changes from ‘123’ to ‘6789’. During the evolution, we want to protect good ‘building blocks’ and combine them to form even more highly fit and higher-order ‘building blocks’.

4.4.2 Application

In this section, we apply the Schema Theorem to the index tracking problem. We detail the solution approach named Roulette Genetic Algorithm (R-GA) from three aspects: the selection, crossover, and mutation. We first introduce the selection process named Roulette Wheel selection.

- **Roulette Wheel Selection**

The selection process is implemented regarding the fitness values ($P_i = f_i / \sum f_j$) of the portfolios, more precisely, we create a roulette wheel, where each current individual takes a slot. The size of the slot is based on the fitness value of the individual. The fitter the individuals the bigger size the slot. During the reproduction process, we rotate the roulette wheel to decide which individuals should be selected into the mating pool. As for the mating partners, they are randomly selected. Therefore, stocks of the portfolios with fitness values above the population average are more likely to appear in the next generation. For stocks of portfolios with fitness value below the population average, they are less likely to appear in the next generation and will eventually die out.

- **Crossover**

Semi-crossover operator

For a semi-crossover, we keep the mutual stocks untouched, cut the parent portfolios at the same locus, and exchange the pieces after the locus. Considering a general case, where portfolio A (cardinality is K) is in the mating pool and portfolio B (mating partner) is randomly selected, they share M mutual stocks. What is the probability of a particular stock (H) in portfolio A being survived after the mating? To solve the problem, we should consider the case with two scenarios: the stock is a non-mutual stock and the stock is a mutual stock. We use figure 4.9 for illustration.

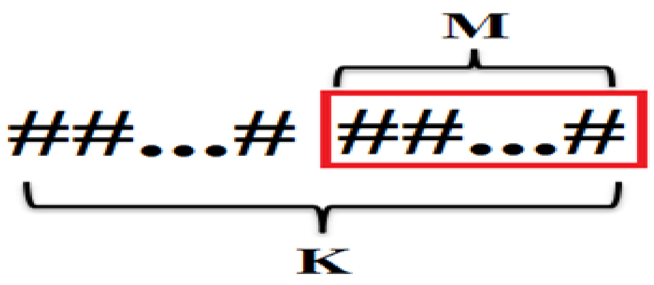


Figure 4.9: Portfolio illustration

- Non-mutual stock. The probability of stock H sitting outside the 'building block' is $P_{out} = \frac{K-M}{K}$, and it has 50% chance of being exchanged by the crossover. So the survival probability of stock H sitting outside the 'building block' is: $P_{out-s} = \frac{K-M}{2K}$
- Mutual stock. Once a stock is sitting inside the 'building block', it would be guaranteed to survive. So the survival probability of stock H sitting inside the 'building block' is: $P_{in-s} = \frac{M}{K}$

Therefore, the total survival probability is $P_s = \frac{K+M}{2K}$, and then the expected number of stock H at generation $g+1$ can be given by the following equation. $\bar{M}$ is the average value of M and $\Delta(H)$ is the disturbance caused by the mating partners. The reason why we introduce $\Delta(H)$ into the equation is because if the mating partner contains a particular stock H, it has a potential chance to be inherited by the offspring.

$$m(H, g+1) \geq m(H, g) * \frac{f(H)}{\bar{f}} * \frac{K+\bar{M}}{2K} + \Delta(H) \tag{4.15}$$

$\Delta(H)$ is affected by three factors: the number of the population, the size of the index, and the cardinality. Generally, its approximate calculation can be given by $\Delta(H) = \frac{n*k}{2N}$, where N is the size of the index. Furthermore, if the crossover occurs by a certain probability, then we have:

$$m(H, g+1) \geq m(H, g) * \frac{f(H)}{\bar{f}} * \left[1 - P_c \frac{K-\bar{M}}{2K}\right] + \Delta(H) \tag{4.16}$$

We broadly categorise the evolution into different stages named as: initial stage, expansion stage, maturation stage, and stagnation stage. At the initial stage, the evolution is most likely to drive populations to move along the fitness landscape in random manner. During that period, few mating pairs have mutual stocks, especially for the large sized indices, where most of the mating pairs don't have any mutual stocks. Hence, $\bar{M}$ is approximately equal to zero, thus the survival probability is around 50% at the initial stage. At the expansion stage, some mating pairs share a few stocks while others may share less or none. Small good 'building blocks' are formed. At the maturation stage, most mating pairs share a considerable number of stocks. Good small ‘building blocks’ are combined and constructed to large ones. At the stagnation stage, most of the individuals have similar combinations to a certain degree. The maximum fitness value maintains the same for several generations, which means that the evolution either has reached the global optimal point or is stuck at a local optimum. Generally, $\bar{M}$ gradually increases along with the evolution and finally reaches K.

The following tables show the results of the simulation (data from the Hang Seng index), where we examine the effect of the semi-crossover operator on stocks given by the above equations. Note here that the data set is from the Hang Seng index. For simplicity, we set the cardinality $K = 4$ and shrink the size of the index to 10 stocks. ($f_i = \frac{1}{f(x)}$, $\bar{M} = 2 \; and \; \Delta(H) = 0.8$)

Table 4-5 (a): Semi-crossover operator simulation – portfolio processing

Portfolio No.	Composing stocks				$f(x)$ Tracking error	f_i Fitness value	Expected number of times selected $f_i/\bar{f}$	Actual times selected from Roulette Wheel
1	1	2	4	8	0.635992	1.57234682	0.83745366	1
2	3	7	9	10	0.655519	1.5255088	0.81250708	0
3	1	4	6	10	0.564189	1.77245568	0.9440344	3
4	1	4	5	9	0.516545	1.93593975	1.03110828	1
5	2	3	4	6	0.445397	2.245188	1.19581817	0
6	2	3	5	8	0.514576	1.94334753	1.03505376	1
7	2	4	5	9	0.435712	2.29509401	1.2223988	2
8	1	3	7	9	0.649325	1.54006083	0.82025769	1
9	1	2	9	10	0.629814	1.58777036	0.84566844	0
10	3	4	5	6	0.424157	2.35761758	1.25569972	1
					Sum	18.7753294		
					Ave	1.87753294		
					Max	2.35761758		

(continued)

Mating pool (stocks)				Mating Partner (stocks)				Offspring (stocks)				$f(x)$ Tracking error	f_i Fitness value
3	4	5	6	8	9	5	6	3	4	5	6	0.424157	2.3576176
1	4	5	9	2	7	5	9	1	2	5	9	0.56885	1.7579327
1	4	6	10	7	4	6	10	1	4	6	10	0.564189	1.7724557
6	1	4	10	2	1	4	10	1	4	6	10	0.564189	1.7724557
1	4	6	10	2	7	8	10	1	6	8	10	0.583497	1.7138049
4	5	9	2	7	8	10	2	2	7	8	9	0.63671	1.5705737
1	2	4	8	3	5	6	8	2	4	5	8	0.553196	1.8076776
1	7	3	9	6	8	3	9	3	7	8	9	0.557986	1.7921597
4	2	5	9	8	2	5	9	2	4	5	9	0.435711	2.2950993
2	5	3	8	6	9	3	8	2	3	8	9	0.540078	1.8515844
												Sum	18.691361
												Ave	1.8691361
												Max	2.3576176

Table 4.5(b) semi-crossover operator simulation – stocks processing

Stocks No.	$m(H,g)$	$F(H)$	$\frac{F(H)}{\bar{f}}$	$\frac{m(H,g)*F(H)}{\bar{f}}$	$\frac{K+\bar{M}}{2K} = 0.75$	$\Delta(H) = 0.8$ / $m(H,g+1)$	Actual times selected
1	5	1.6817147	0.8957045	4.47852	3.36	4.16	4
2	5	1.9287493	1.0272786	5.13639	3.85	4.65	5
3	5	1.9223445	1.0238673	5.11934	3.84	4.64	3
4	6	2.0297736	1.0810855	6.48651	4.86	5.66	5
5	4	2.1329997	1.1360651	4.54426	3.41	4.21	4
6	3	2.1250871	1.1318508	3.39555	2.55	3.35	4
7	2	1.5327848	0.8163824	1.63276	1.22	2.02	2
8	2	1.7578472	0.9362537	1.87251	1.40	2.20	5
9	5	1.9278814	1.0268163	5.13408	3.85	4.65	5
10	3	1.6285783	0.8674033	2.60221	1.95	2.75	3

Usually, the actual count of a particular stock H at generation $g + 1$ matches with the expectation, except that there is one special circumstance, where the actual count of H8 is almost two times its expectation. By further investigation, we find that the special circumstance is caused by the extreme amount of disturbance ($\Delta(H)$=3) caused during the mating process. Generally, the simulation reached the expectation. However, we also discover some issues of the semi-crossover operator. The maximum fitness value (2.3576176) remains constant and it might be lost in the future generations if the crossover site locates on any position of the string '3456'. In addition, the average fitness value slightly decreases from 1.878 to 1.869. It enlightens us that the generational approach may not be suitable for the crossover operator as it is too random that we cannot guarantee the quality of the new generation. We believe that a steady-state-approach is a better choice for the semi-crossover operator and large size of populations must be generated to ensure a certain probability of achieving fit offspring. Thus, we can replace the unfit individuals with the fitter offspring to form a new generation so that the average fitness value could be improved generation by generation. We test the idea with several data sets, but the results still cannot meet the expectations. We notice that the semi-crossover operator makes the mating process extremely fast, but prolongs the whole computing time, because the fitness of the generations improves very slowly. It means that we cannot efficiently construct good 'building blocks', which make the algorithm weak in searching power.

Semi-optimization Operator

The semi-optimization operator can be seen as an extreme case of crossover, where the locus is always chosen at the optimal position. Thus, the parents can most efficiently exchange their pieces of genetic information and the population size is fairly small compared with semi-crossover. However, the computing time for each mating process is much longer than that of the semi-crossover, because each time the solver needs to optimize the non-mutual stocks; the more there is of non-mutual stocks the larger the amount of time the solver takes for optimization. Next, we need to consider the survival probability of a particular stock under semi-optimization. Again, we break the case down to two scenarios: when a particular stock is a mutual stock and when it is a non-mutual stock.

- Non-mutual stock. Once the mating partner is selected, the optimal combination is already determined regardless of whether the breeding process takes place or not. So its survival probability is either: $P_{out-s} = \frac{K-M}{K}\ or\ 0$
- Mutual stock. It is the same as the semi-crossover : $P_{in-s} = \frac{M}{K}$

Therefore, the total survival probability is:

$$\frac{M}{K} \le P_s \le 1$$

Accordingly, we consider the expected number of stock H at generation $g+1$. Neutrally, we can use the average probability, where $P_s = \frac{\frac{M}{K}+1}{2} = \frac{K+M}{2K}$. So equation (4.15) and (4.16) are still feasible for the semi-optimization operator. Alternatively, we may play it more conservatively by using the lower bound of the survival probability. Again, we run simulations on the semi-optimization operator using the population sample generated from the semi-crossover simulation. The following tables show the results:

Table 4-6 (a): Semi-optimization operator simulation – portfolio processing

Portfolio No.	Stocks				$f(x)$ Tracking error	f_i Fitness value	Expected number of times selected $f_i/\bar{f}$	Actual times selected from Roulette Wheel
1	1	2	4	8	0.635992	1.57234682	0.83745366	1
2	3	7	9	10	0.655519	1.5255088	0.81250708	0
3	1	4	6	10	0.564189	1.77245568	0.9440344	3
4	1	4	5	9	0.516545	1.93593975	1.03110828	1
5	2	3	4	6	0.445397	2.245188	1.19581817	0
6	2	3	5	8	0.514576	1.94334753	1.03505376	1
7	2	4	5	9	0.435712	2.29509401	1.2223988	2
8	1	3	7	9	0.649325	1.54006083	0.82025769	1
9	1	2	9	10	0.629814	1.58777036	0.84566844	0
10	3	4	5	6	0.424157	2.35761758	1.25569972	1
					Sum	18.7753294		
					Ave	1.87753294		
					Max	2.35761758		

(Continued)

Mating pool (stocks)	Mating Partner (stocks)	New population (stocks)	$f(x)$ Tracking error	f_i Fitness value
3 4 5 6	8 9 5 6	4 5 6 9	0.41627	2.402287
1 4 5 9	2 7 5 9	2 4 5 9	0.435711	2.2950993
1 4 6 10	7 4 6 10	4 6 7 10	0.536048	1.8655046
6 1 4 10	2 1 4 10	1 4 6 10	0.564189	1.7724557
1 4 6 10	2 7 8 10	2 6 7 10	0.528463	1.8922801
4 5 9 2	7 8 10 2	2 4 5 9	0.435711	2.2950993
1 2 4 8	3 5 6 8	1 3 6 8	0.446564	2.2393207
1 7 3 9	6 8 3 9	1 3 6 9	0.4561979	2.1920311
4 2 5 9	8 2 5 9	2 4 5 9	0.435711	2.2950993
2 5 3 8	6 9 3 8	3 5 6 8	0.447001	2.2371315
			Sum	21.486308
			Ave	2.1486308
			Max	2.402287

Table4-6 (b) semi-optimization operator simulation – stocks processing

Stocks No.	$m(H,g)$	$F(H)$	$\frac{F(H)}{\bar{f}}$	$\frac{m(H,g)*F(H)}{\bar{f}}$	$\frac{K+\bar{M}}{2K}=0.75$	$\frac{\Delta(H)=0.8}{m(H,g+1)}$	Actual times selected
1	5	1.6817147	0.8957045	4.47852	3.36	4.16	3
2	5	1.9287493	1.0272786	5.13639	3.85	4.65	4
3	5	1.9223445	1.0238673	5.11934	3.84	4.64	3
4	6	2.0297736	1.0810855	6.48651	4.86	5.66	6
5	4	2.1329997	1.1360651	4.54426	3.41	4.21	5
6	3	2.1250871	1.1318508	3.39555	2.55	3.35	7
7	2	1.5327848	0.8163824	1.63276	1.22	2.02	2
8	2	1.7578472	0.9362537	1.87251	1.40	2.20	2
9	5	1.9278814	1.0268163	5.13408	3.85	4.65	5
10	3	1.6285783	0.8674033	2.60221	1.95	2.75	3

Generally, we find that the number of stocks in the portfolios above average fitness increases more than expected and those below average fitness decrease more than expected. On one side, it shows that the semi-optimization can successfully protect the good stocks. On the other side, as the growth speed or the decay speed is too fast, we have more risk of obtaining sub-optimal solutions. For the fitness value, the average fitness value reaches 2.149 and the maximum value is increased from 2.358 to 2.402. Both of the values are better than those of semi-crossover. Therefore, we conclude that the semi-optimization operator is the better choice for the R-GA.

- **Mutation**

The mutation method is similar to that of the EIT-GA, where we provide ‘fresh’ genes to the evolution to maintain the diversity. It occasionally happens in the reproduction process with a certain probability. We define the mutation parameter as P_m, if $P_m \geq 0.02$ then select the mating partners from the previous offspring, otherwise, the mutation process is triggered, where the mating partners are randomly generated. The following are the steps of the R-GA.

```
• Generate the initial population
• Repeat
    ▪ Evaluate the fitness of individuals
    ▪ Create the roulette wheel to select individuals into mating pool
    ▪ If $P_m \geq 0.02$
        − Select the mating partners from the previous offspring
    ▪ Else
        − Randomly generate mating partners
    ▪ Mate the parents to produce new offspring
• Until
    ▪ Meet the stopping criteria
```

Note here that, the stopping criteria are the same with that of EIT-GA.

4.4.3 Investigation of population size

The aim of investigating initial population size is to construct reliable relations between three factors: cardinality, initial population size, and the size of the index. It is almost impossible to summarize the relations of the three factors from empirical studies without any framework. Thus, we use the following equation to define the framework, and then we try to search for the best ‘frequency’ for the equation.

$$Initial\ population = \frac{Frequency\ \times Index\ Size}{Cardinality}$$

The ‘frequency’ means the average number of times a particular stock can be picked up when we generate the initial population. We generate five sets with frequency equals to 6, 9, 12, 15, and 18 (write as f6, f9, f12, f15, and f18). The corresponding size of the initial population for each index with different cardinalities is shown in table 4-7 and the computing results are shown in table 4-8. (The reason why we don’t test the algorithm by the last three indices or the Nikkei index with bigger cardinalities is because a single mating process could cost us extraordinary amount of computing time. Thus, we decide to find the most desirable set first, and then use it to generate tacking portfolios for last three indices.)

Table 4-7: Initial population size under different frequencies

Frequency	K	Initial population size				
		Hang Seng	DAX	FTSE	S&P	Nikkei
f6	10	19	51	53	59	135
	15	12	34	36	39	----
	20	----	26	27	29	----
f9	10	28	77	80	88	203
	15	19	51	53	59	----
	20	----	38	40	44	----
f12	10	37	102	107	118	270
	15	25	68	71	78	----
	20	----	51	53	59	----
f15	10	47	128	134	147	338
	15	31	85	89	98	----
	20	----	64	67	74	----
f18	10	56	153	160	176	405
	15	37	102	107	118	----
	20	----	77	80	88	----

Table 4-8: Computational results of R-GA for smaller size indices under different input frequencies

Index	K	$Population = (f6)$		$Population = (f9)$		$Population = (f12)$		$Population = (f15)$		$Population = (f18)$	
		TE	CT	TE	CT	TE	CT	TE	CT	TE	CT
Hang Seng	10	0.101010742	5.7	0.101010742	10.9	0.101010742	16.3	0.101010742	16.7	0.101010742	19.5
	15	0.056886088	3.3	0.056886088	5.3	0.056886088	8.0	0.056886088	8.5	0.056886088	13.3
DAX	10	0.069071471	35.8	0.070037168	111.9	0.067296843	113.0	0.067296843	113.7	0.069161645	145.4
	15	0.037468948	74.4	0.037904515	217.7	0.037312618	302.4	0.037468948	243.9	0.033208033	366.9
	20	0.025376215	126.0	0.022087138	268.5	0.022141031	527.0	0.022383211	623.5	0.019949278	652.4
FTSE	10	0.101422483	80.3	0.099823448	97.0	0.099823448	133.4	0.099823448	169.1	0.11011537	167.1
	15	0.064295024	140.6	0.066674418	359.9	0.05703033	372.3	0.05435597	412.1	0.056244049	678.2
	20	0.038013702	401.6	0.032838238	437.5	0.033113403	863.8	0.030839004	1133.7	0.031466236	1795.2
S&P	10	0.100628635	63.6	0.100628635	162.0	0.099653821	168.3	0.100628635	193.4	0.090899847	283.7
	15	0.053665422	257.8	0.056136286	338.9	0.051351667	709.0	0.053651399	737.8	0.050947878	869.0
	20	0.032541688	196.7	0.027379289	482.5	0.027379289	938.9	0.025649629	1338.8	0.026032438	1537.0
Nikkei	10	0.099245473	303.1	0.080572321	420.9	0.080572321	682.8	0.087102594	963.0	0.087366785	1142.3

Noticeable from table 4-8, it is hard to choose a particular frequency whose tracking errors are always better than those of the others, consequently making it very difficult for comparisons. Thus, we decide to compare the average tracking errors of each index. (For example, the average tracking error of the DAX with frequency 6 is calculated by $(0.069071471 + 0.037468948 + 0.025376215)/3 = 0.0439722)$ Thus, we are able to transfer the above table to an average tracking error table, shown in table 4-9.

Table 4-9: Average Tracking Error

Index	Average Tracking Error				
	f6	f9	f12	f15	f18
Hang Seng	0.078948	0.078948	0.078948	0.078948	0.078948
DAX	0.043972	0.043343	0.042250	0.042383	0.040773
FTSE	0.067910	0.066445	0.063322	0.061673	0.065942
S&P	0.062279	0.061381	0.059462	0.059977	0.055960
Nikkei	0.099245	0.080572	0.080572	0.087103	0.087367
Total Average	0.070471	0.066138	0.064911	0.066017	0.065798

After comparing the above average tracking errors for each index, we first ruled out the sets f6 and f9, because they have the worst solutions of the five at most of the time. For the other three sets, f18 have the best average tracking errors for DAX and S&P, f15 have the best one for FTSE, and f12 have the best one for Nikkei. Among the three, f12 have the smallest *total average tracking error* which shows its strong stability. In addition, we compare the computing time associated with the three sets. From figure 4.10, we could see that f12 requires the least amount of computing time. Therefore, we conclude that f12 is the most desirable frequency we are looking for. Although the result cannot be proved rigorously, we believe that f12 is the most suitable choice.

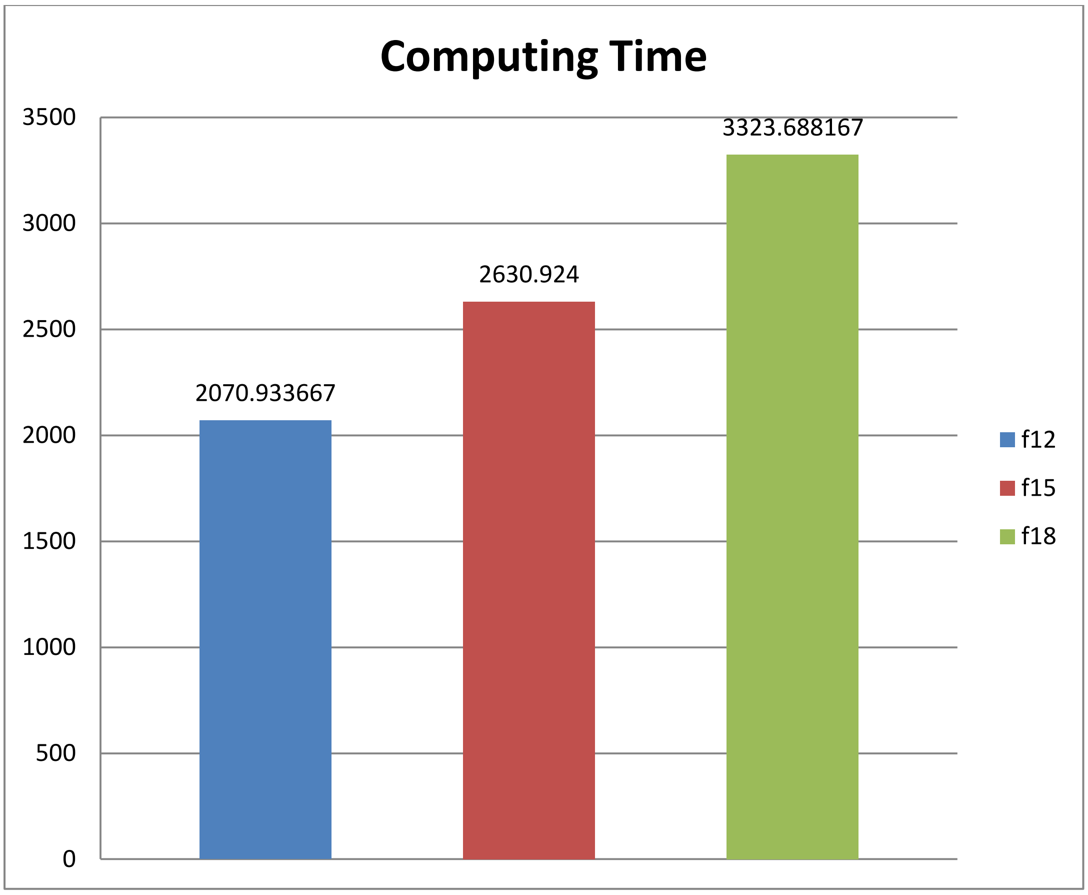


Figure 4.10: Computing time comparison of f12, f15, and f18

Then, the approximate expression for the relations among cardinality (K), population size (n), and the size of an index (N) can be given by:

$$n \approx \frac{N * 12}{K}$$

4.4.4 Enhancement

Before we use f12 to implement the rest of the tests, we would like to make some improvements on the R-GA. We find that the R-GA still has potential room to improve the solution qualities while not largely increasing the computing time. We can achieve this goal by adjusting the P_m rate to a very high value when the stopping criteria are almost met. We explain it using the following figure.

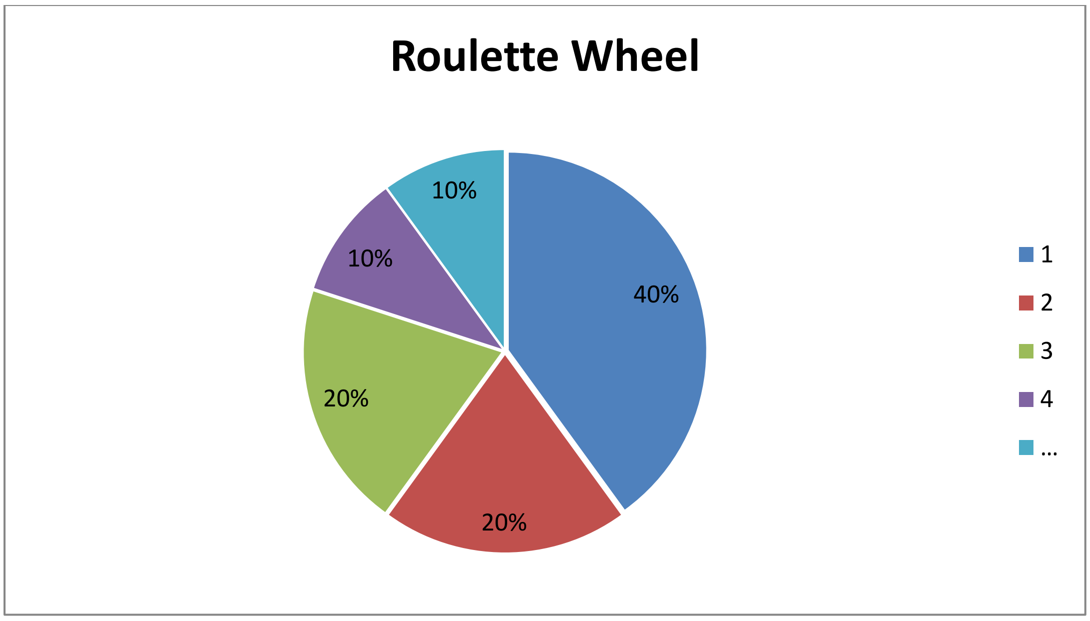


Figure 4.11: Roulette wheel

When the best tracking portfolio is unchanged for two generations, the roulette wheel would be like the above figure. The majority part of the wheel would be taken by the best tracking portfolio, and then followed by the second best portfolio, and so on. At that moment, if the P_m rate is increased to a very high value such as 0.9, most of the mating partners would be randomly generated. These randomly created mating partners can bring back the eliminated stocks and preserve diversity. Therefore, we might further improve the solution values. If the best solution is changed, we would set the mutation rate back to 0.2; otherwise, the algorithm would be terminated. We call the improved algorithm the ‘Enhanced Roulette Genetic Algorithm’ (ER-GA).

4.4.5 Computational Results

We test the ER-GA on all the 8 market indices. As for the last three indices and Nikkei index with bigger cardinalities, we set a limitation for each single mating process. For this reason, the solver would provide us the best solution it can find during that limited duration. The solving time limitation for Nikkei 225 with bigger cardinalities (15, 20, 25, and 30) and S&P 500 is 1s and it is 0.5s for the two largest indices, Russell 2000 and Russell 3000. The following table shows the comparison between the computing results of EIT-GA and ER-GA.

Table 4-10: EIT-GA vs ER-GA

Index	K	EIT-GA		ER-GA	
		TE	CT	TE	CT
Hang Seng	10	0.10101	16.5	0.10101	17.5
	15	0.05689	13.9	0.05689	8.7
DAX	10	0.06730	617.8	0.06730	115.2
	15	0.03486	271.9	0.03515	311.8
	20	0.01924	1036.7	0.02021	540.4
FTSE	10	0.09982	615.1	0.09982	134.8
	15	0.05436	401.7	0.05664	429.0
	20	0.02858	2359.7	0.02906	980.2
S&P 100	10	0.09629	437.2	0.09965	175.4
	15	0.04755	2102.3	0.05040	626.0
	20	0.02789	3200.4	0.02617	1009.1
Nikkei 225	10	0.08581	1748.2	0.08057	784.3
	15	0.04477	3721.6	0.04269	2197.5
	20	0.02512	3312.0	0.02690	1041.2
	25	0.00913	3725.0	0.01125	2336.5
S&P 500	10	0.06296	607.0	0.07260	1443.6
	15	0.03989	1854.0	0.03502	4369.1
	20	0.02166	2233.0	0.01738	3779.4
	25	0.01157	1933.0	0.01021	3140.1
	30	0.00640	3442.0	0.00924	3633.1
Russell 2000	30	0.01214	1969.0	0.01072	2639.9
	40	0.00490	476.0	0.00401	1808.8
	50	0.00116	3650.0	0.00043	1147.7
	60	0	277.0	0	534.7
	70	0	286.0	0	456.1
Russell 3000	30	0.01449	477.0	0.00794	5727.3
	40	0.00455	240.0	0.00198	5874.7
	50	0.00039	1703.0	0.00059	1850.8
	60	0	277.0	0	892.9
	70	0	389.0	0	770.8
	80	0	482.0	0	1358.4

We separate the above table into two parts by the size of the indices. The first part contains the computational results of the first four indices and the second part contains those of the rest four indices. There are 11 comparisons in the first part. In terms of solution quality, EIT-GA beats ER-GA 6 times, it loses to ER-GA 1 time, and they draw for 4 times. For the computing time, EIT-GA costs about 11073 time units to solve the first four indices while ER-GA only costs about 4348 time units. There are 20 comparisons in the second part. In terms of solution quality, EIT-GA beats ER-GA 5 times, and it loses to ER-GA 10 times, and they drew for 5 times. For the computing time, EIT-GA costs about 32802 time units while ER-GA costs 45787 time units.

Based on the above fact, we conclude that the EIT-GA works better for small size indices while ER-GA is more suitable for the large size ones. However, the computing results of ER-GA for the large indices can prove that it is able to construct sound relations between the cardinality, the size of the index, and the size of the populations. Overall, we conclude that ER-GA is the better choice of the two solution approaches in general.

Chapter 5

5 .Ant Colony Optimization

5.1 Introduction

Ant colony optimisation (ACO) is a meta-heuristic solution approach for approximate optimisation. The ACO is inspired by the observation of the natural foraging behaviour of real ants. The core of the foraging behaviour is the indirect communication between the ants. This indirect communication is facilitated by the chemicals produced by the ants, called pheromones. When ants go out to search for food, they initially explore the surrounding area in a random manner. While walking from their nest to the food resources and vice versa, they deposit pheromone trails on the ground. Other ants can smell the pheromones and tend to choose, probabilistically, paths marked by the strong pheromone concentrations. In the long run, the shortest path is more likely to have the strongest pheromone concentrations. Thus, this indirect means of communication that enables the ants to find the optimal path between the nest and the food resources is known as 'stigmergy'. The ACO meta-heuristic was first formalised by Dorigo and his colleagues in 1999. In the book '*Ant Colony Optimisation*' (Dorigo M, Stützle T, 2004), they give a comprehensive description about the ACO Meta-heuristic, which can be broken down to three parts: problem representation, the ants' behaviour and the general structure of the ACO meta-heuristic.

Firstly, they provided a formal characterisation of the class of problems to which the ACO can be applied. They considered a general minimization problem (S, f, Ω), where S represents the set of candidate solutions, f represents the objective function, and Ω represents a set of constraints. The aim is to find a globally optimal solution s^*, where $f(s^*) \leq f(s), s \in S$. Then, they designed an artificial ant which is a stochastic constructive procedure that builds solutions by moving randomly on a completely connected construction graph $G_c = (C, L)$, where C and L are denoted as the set of the nodes and the set of connections on the graph respectively.

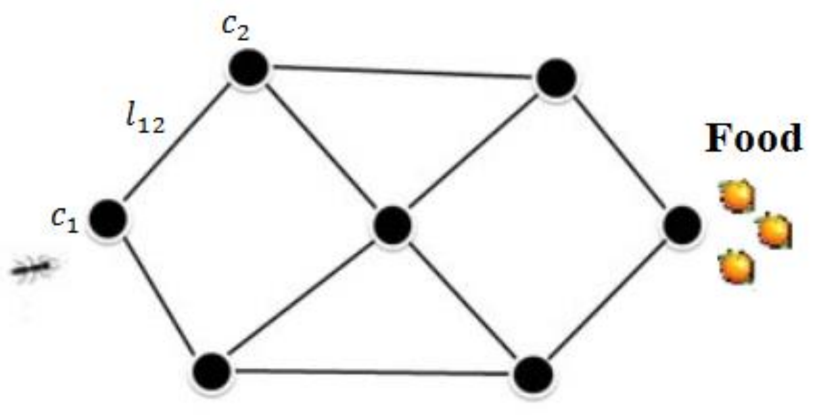


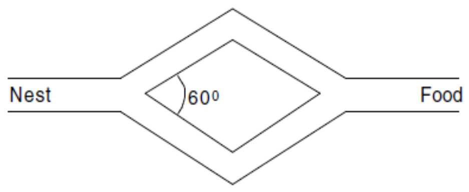
Figure 5.1: Construction graph

The components $c_i \in C$ and the connections $l_i \in l$ are associated with pheromone trails τ_i and τ_{ij} and heuristic value, which is also called heuristic information, η_i and η_{ij} respectively. The pheromone trail encodes the memory about the entire searching process, and is updated by the ants themselves. In contrast, the heuristic information is provided by other sources, usually this is the estimated cost of adding components or connections to the solution under construction. The ants use these values to make probabilistic decisions on how to walk on the graph. The ACO algorithm can be broke down into three steps: Construct Ants Solutions, Update Pheromones and Daemon Actions.

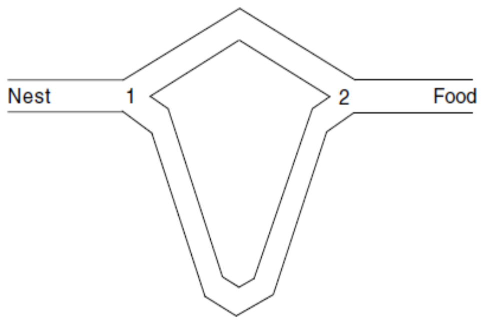
- **Construct Ants Solutions**

This manages colonies of ants that walk on the construction graph of the considered problem. The moving of ants is guided by stochastic decisions based on the pheromone trails and the heuristic information. In this fashion, the ants gradually build the solutions for the problem. Deneubourg and his colleagues (Deneubourg, J.-

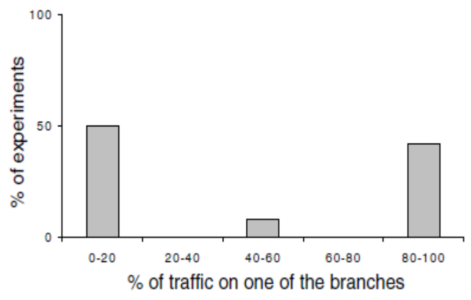
L., Aron, S., Goss, S., &Pasteels, J.-M., 1990) (Goss et al, 1989) designed an experiment, where they used a double bridge to connect the nest and the food resources. They ran the experiments by varying the ratio $r = l_l/l_s$, where l_l represents the length of the longer bridge and l_s represents the length of the shorter one. In the experiments, they initially set $r = 1$. At the beginning, as there are no pheromones on either of the bridges, the ants do not have any preference so they select the either of the two bridges with equal probability; yet, due to the random fluctuation, a few more ants will select one bridge over the other. Thus, one bridge starts to accumulate more pheromone concentrations than the other. The extra amount of pheromone in turn will stimulate more ants to choose that bridge. The outcome is that, although initially the ants randomly selected the path, eventually all the ants used the same bridge. In the second experiment, the researchers set $r = 2$, so the long bridge is twice as long as the short one. In this case, almost all the ants choose the short path as the pheromone accumulates much faster on the short path than it does on the long one. However, there is a small percentage of ants still choose the longer bridge, and this was explained as a type of 'path exploration'.



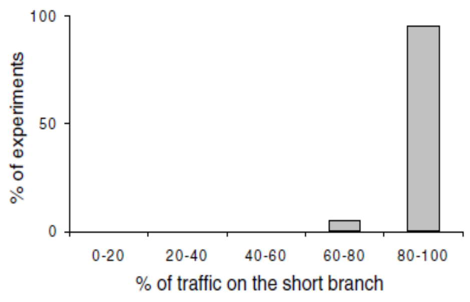
(a)



(b)



(c)



(d)

Figure 5.2: Double bridge experiment

- **Update Pheromones**

This is a process that modifies the current pheromone concentrations. The pheromone values can either be increased by depositing new pheromone on the nodes or connections the ants walked through or decreased by evaporation. The depositing of new pheromone can be achieved in two ways. Either the nodes or the connections are used by many ants, or are only used by one ant that produces a really good solution. Conversely, the evaporation plays a role of causing the past information to be forgotten: it helps the ACO algorithm to avoid early convergence to sub-optimal solutions. However, a too high evaporation rate would also lead to rapid convergence. Deneubourg and his colleagues carried out an additional experiment that reveals the natural disadvantage of ACO, where only the long bridge is initially offered to a colony of ants and after 30 minutes a short bridge is added. In this case, the short bridge is selected by few ants while the majority of the ants are trapped using the long bridge. They explained the reason for this as the high pheromone concentration on the long bridge and the slow evaporation rate.

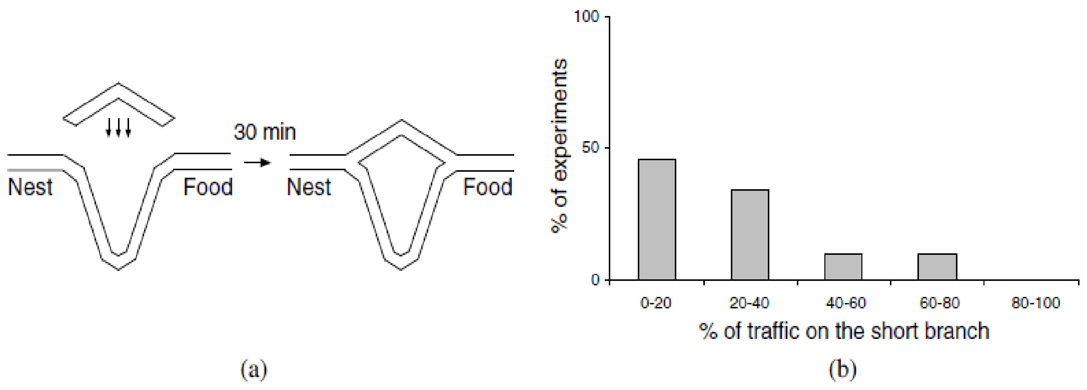


Figure 5.3: Additional experiment

- **Daemon Actions**

This is a step which is biased toward the searching process. For example, the daemon can deliberately select one or a few ants that build good solutions in the algorithm iterations, and then deposit additional pheromone on the nodes and connections they used. In other words, the daemon actions are used to magnify the effect of good solutions.

5.2 Application

The way that we apply the ACO algorithm to the Index Tracking problem follows exactly the three steps: Construct Ants Solutions, Update Pheromones and Daemon Actions. In order to improve the ACO algorithm to solve the Index Tracking problem, we use artificial ants which maintain the characteristics of real ants while they have some additional capabilities. These capabilities are:

1. Probabilistically construct paths by pheromone trails. The ants are determined to have two working modes: forward and backward. When they are in the forward mode, they are moving from the nest toward the food resources, and when they are in the backward mode, they are heading back from the food resources to the nest. When they move forward, ants build the solutions by probabilistically choosing the nodes and connections to move.

2. Memorise the paths they have gone along. The artificial ants can memorize the nodes they have visited to avoid visiting the same node twice. Therefore they can never be trapped in a loop.

3. Go through extra nodes and choose the best K out of $K + extra$ to build optimal solutions. When the cardinality is K, it means that an ant must walk through K nodes on the construction graph to build up a solution. Our ants are more intelligent, they can go through $K + extra$ nodes and evaluate the solution values of each possible combination, and then select the best K nodes to construct an optimal path. In our research, the number of the extra nodes is user specified, but it should be restrictively controlled. By setting too high a number, the computer time would be affected. We set this number equal to 2 in our research.

4. Update pheromones while moving backward to the nest. The artificial ants deposit pheromone on a path biased toward the solution values. The smaller the solution value of a path the more pheromone is deposited. In addition, we use a steady-state approach to update the pheromones, where we sort the solution values of the two recent colonies from the smallest to the largest and only use the better half to update the pheromone information.

In our research, we treat each portfolio as a path and each stock as a node on the construction graph. In the following, the meaning of the terms should become clear if we switch them from time to time. Below are the details about our ACO solution approach:

We first generate a number of portfolios and calculate their solution values, and then we sort the solution values from the smallest to the largest and use these sorted solution values to update the pheromone information. The pheromone information updating compromises two parts: heuristic information and pheromone trails. Equations (5.1) to (5.3) are used to update the heuristic information.

$$H_j = \frac{1/OB_j}{\sum_j 1/OB_j} \qquad j = 1..n \tag{5.1}$$

$$PI_i = \sum_j H_i * Prop_{j,i} \qquad i \in N \tag{5.2}$$

$$P_i = \frac{PI_i}{\sum_i PI_i} \qquad i \in N \tag{5.3}$$

In equation (5.1), OB_j represents the solution value of portfolio j and H_j represents the weight of portfolio j in a colony, the smaller a solution value the bigger the weight of a portfolio. In equation (5.2), $Prop_{j,i}$ represents the weight of stock i in portfolio j and PI_i represents the weight of stock i in a colony. After achieving PI_i, we use equation (5.3) to calculate the normalised probability of each stock, and then we use the normalised probabilities to select the first stock in a portfolio. From another point of view, equation (5.3) determines the first movement of an ant on the construction graph of the considered problem. After the first step, an artificial ant still has to move '$K + extra - 1$' nodes to build a solution. As for the rest of the movement, it is no longer guided by the heuristic information, but by the pheromone trails between the nodes. We construct a table to measure the amount of attraction between each pair of the stocks.

$$P_{j,i,i1,g} = \frac{1}{OB_j} \qquad j = 1..n \ (Prop_{j,i} <> 0, Prop_{j,i1} <> 0) \tag{5.4}$$

$$P_{j,i,i1,g} = 0 \qquad i = i1 \tag{5.5}$$

$$P_{i,i1} = \sum_{j}\sum_{g} P_{j,i,i1,g} \tag{5.6}$$

Equation (5.4) represents the attraction value between stock i and stock $i1$ in portfolio j in colony g. Generally, the smaller the solution values of a portfolio the bigger the attraction between its stocks. Equation (5.5) shows that there is no attraction between a stock and itself. The attraction only occurs between two different stocks. Equation (5.6) gives the latest total amount of attraction between stock i and stock $i1$, and $\mathrm{P}_{\mathrm{i,i1}}$ is updated after each colony. The pheromone trail updating is vitally important for the ACO algorithm as it can directly affect the quality of the solutions. The following are two requirements for good pheromone information:

$$\frac{\sum_{i\in Optimal} PI_i}{\sum_i PI_i} > \mathrm{M} \qquad \mathrm{M} \in (0,1)$$

$$P_{i\in Optimal,\ i1\in Optimal} > P_{i\in Optimal,\ i1\in Rest} > P_{i\in Rest,\ i1\in Rest} \qquad i \neq i1$$

Note that, here, *Optimal* is defined as a set of stocks in the 'optimal' tracking portfolio or very good tacking portfolios and *Rest* is defined as a set of the rest of the stocks in an index. In the first expression, *M* must be big enough to maintain a certain probability of choosing a stock in the *Optimal* set. The second expression shows that the attraction value between '*Optimal' stocks* must be bigger than any other pairs of stocks. Furthermore, the amount of attraction between an '*Optimal'* stock and a '*Rest'* stock should be greater than that of between two '*Rest'* stocks. Therefore, it ensures that if somehow the '*Rest'* stocks are picked, we still have the opportunity to construct a good tracking portfolio. By simply using the equations (5.4) ~ (5.6) are not sufficient enough, because if we randomly generate a colony, the difference between the smallest solution value and the largest solution value may not be big enough to be distinguishable. Consequently, the attraction table is not able to offer instructive guidance. Table 5-1 shows such an example. (For simplicity, we only generate 20 portfolios and set the cardinality to 4)

Table 5-1: Simulation

Portfolio No.	Stocks				Tracking error OB_j	Attraction $P_{j,i,i1}$
1	4	9	7	8	0.564493	1.771501
2	2	7	9	3	0.566167	1.766263
3	3	1	9	4	0.567521	1.762049
4	2	4	1	7	0.570928	1.751534
5	5	6	8	7	0.571562	1.749591
6	5	9	7	2	0.574703	1.740029
7	1	2	5	4	0.579622	1.725262
8	10	6	9	7	0.588624	1.698877
9	7	1	4	3	0.595866	1.67823
10	5	2	6	10	0.603175	1.657894
11	5	4	1	3	0.607775	1.645346
12	1	9	8	5	0.612161	1.633557
13	2	5	8	6	0.615245	1.625369
14	5	10	2	4	0.616589	1.621826
15	8	1	2	9	0.622338	1.606844
16	5	1	6	2	0.622885	1.605433
17	5	1	7	4	0.625538	1.598624
18	2	10	9	1	0.629814	1.58777
19	8	1	4	2	0.635992	1.572347
20	2	4	10	1	0.638456	1.566279

As shown above, the difference between the largest attraction value and the smallest attraction value is very small, only about 0.21. Therefore, if we construct an attraction table based on the above figures, it could not provide constructive guidance for the selection of stocks. In order to enlarge the differences, we standardize the solution values of a colony, and then we only enlarge the attraction of stocks in a portfolio, whose solution value lies within $(-\infty, -2\sigma]$, $(-2\sigma, -\sigma]$, and $(-\sigma, 0]$, by 50 times, 7 times and 3 times respectively, see figure 5.4. (Note that the numbers 50, 7 and 3 are set based on experience of numerical experiments)

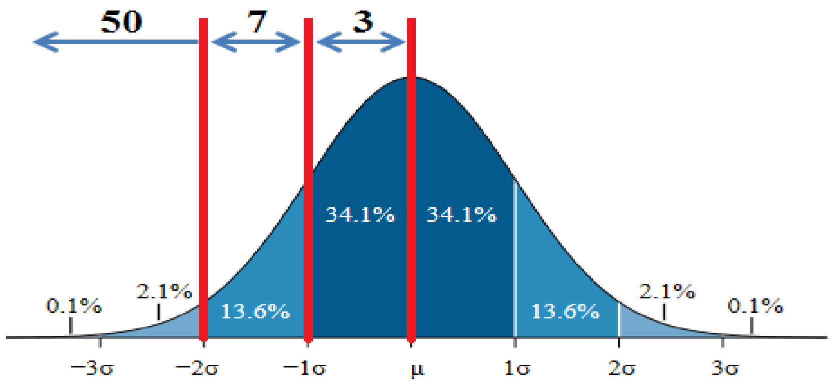


Figure 5.4: Enlargement method

Table 5-2 and table 5-3 below show the attraction table before and after the amplification.

Table 5-2: Attraction table before amplification

	1	2	3	4	5	6	7	8	9	10
1	0	11.4	5.09	13.3	8.21	1.61	5.03	4.81	6.59	3.15
2	11.4	0	1.77	8.24	9.98	4.89	5.26	4.8	6.7	6.43
3	5.09	1.77	0	5.09	1.65	0	3.44	0	3.53	0
4	13.3	8.24	5.09	0	6.59	0	6.8	3.34	3.53	3.19
5	8.21	9.98	1.65	6.59	0	6.64	5.09	5.01	3.37	3.28
6	1.61	4.89	0	0	6.64	0	3.45	3.37	1.7	3.36
7	5.03	5.26	3.44	6.8	5.09	3.45	0	3.52	6.98	1.7
8	4.81	4.8	0	3.34	5.01	3.37	3.52	0	5.01	0
9	6.59	6.7	3.53	3.53	3.37	1.7	6.98	5.01	0	3.29
10	3.15	6.43	0	3.19	3.28	3.36	1.7	0	3.29	0

Table 5-3: Attraction table after amplification

	1	2	3	4	5	6	7	8	9	10
1	0	25.4	19	41.2	11.7	1.61	18.9	4.81	17.2	3.15
2	25.4	0	12.4	22.2	23.9	4.89	36.8	4.8	27.7	6.43
3	19	12.4	0	19	1.65	0	17.4	0	24.7	0
4	41.2	22.2	19	0	10	0	31.3	14	24.7	3.19
5	11.7	23.9	1.65	10	0	17.1	26	15.5	13.8	3.28
6	1.61	4.89	0	0	17.1	0	17.3	13.9	5.1	6.75
7	18.9	36.8	17.4	31.3	26	17.3	0	24.6	42	5.1
8	4.81	4.8	0	14	15.5	13.9	24.6	0	15.6	0
9	17.2	27.7	24.7	24.7	13.8	5.1	42	15.6	0	6.68
10	3.15	6.43	0	3.19	3.28	6.75	5.1	0	6.68	0

Comparing the above two tables, table 5-3 is more instructive than table 5-2. For example, in table 5-2, the attraction value of stock 4 and stock 9 is 3.53 and the attraction value of stock 4 and stock 10 is about the same. However, stock 4 is actually much more attractive to stock 9 than to stock 10, as they are included in two of the three best tracking portfolios. In contrast, the situation is largely improved in table 5-3. The attraction value between stock 4 and stock 9 has increased to 24.7 while the attraction value between stock 4 and stock 10 remains the same. Thus, the attraction table can provide sound guidance for the selection of stocks. After constructing the attraction table, we categorise all the stocks into two types: picked stocks and non-picked stocks, and then we use the following equations to select the remaining stocks:

$$Ppick_{i1 \in Npicked} = \sum_{i \in picked} P_{i,i1} \tag{5.7}$$

$$W = \sum_{i1 \in Npicked} Ppick_{i1} \tag{5.8}$$

$$NPpick_{i1 \in Npicked} = \frac{Ppick_{i1}}{W} \tag{5.9}$$

Equation (5.7) states that the selection of a stock is based on the previous stocks that have been selected. As long as the number of selected stocks is greater than one, then the selection of the next stock has to incorporate the combined attraction. This combined attraction is a result of summing up all the attractions between the current picking stock and the already selected stocks (stated another way, except for the very first step, the rest of the moves of the ants are not independent and are strongly related to the nodes they have gone through), e.g., if stock 1 and stock 2 are selected, then the probability of picking stock j is given by the following:

$$Ppick_j = P_{1,j} + P_{2,j} \tag{5.10}$$

After calculating all the probabilities of picking the non-picked stocks, we normalise them using equations (5.8) and (5.9), and then we use the normalised probability to pick the rest of the stocks to generate a tracking portfolio. By repeatedly doing this for n times, we can create a new colony. Besides the above basic steps, we need to insert mutation and stopping criteria into the ACO algorithm to ensure it can produce good solutions in reasonable time.

- **Mutation**

As we stated Above, the movements of the artificial ants are guided by the pheromone information. There is always a potential risk that such information would cause the algorithm to converge to a local optimum. In order to avoid an early convergence, it is necessary to implement mutation. We decide to do the mutation as soon as an artificial ant reaches the food source. We need to consider the following two problems:

1. The sequence of the nodes that the ants go through cannot affect the objective values of the routes, so we can not implement the mutation by swapping nodes.
2. Randomly changing a node or several nodes in a route would break their inner attraction that makes the solutions even worse.

Considering the above issues, we decide to carry out the mutation by moving more extra nodes, but these nodes are no longer affected by the pheromone information. Instead, they are specially selected from a set of nodes, which can be illustrated by the following figure:

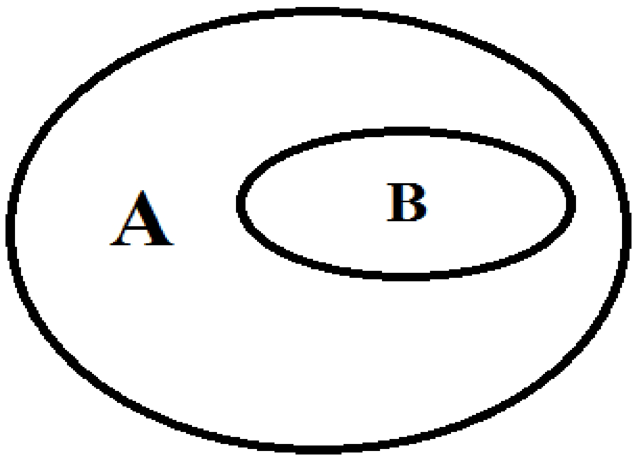


Figure 5.5: Mutation of ACO

In figure 5.5 above, A is the universe of all the nodes between the nest and the food resources and B is a set which contains the nodes that an artificial ant has gone through. The nodes used for mutation are selected from the set C which can be defined as $C = A \cap \bar{B}$. To control the computing time, we normally select five nodes for the mutation. As for the mutation rate, it should not be set either too large or too small, as it either dramatically increase the computational time or does not have any substantial influence on the computing results. For the convenience of comparison, we set the mutation rate to be the same as that of GAs (0.02).

- **Stopping Criteria**

The main purpose of building the ACO algorithm for solving the Index Tracking problem is to find reasonably good solutions in reasonable time. Theoretically, the algorithm should be terminated when all the ants in a colony follow the same path to commute between the nest and the food sources. Practically, this is not feasible, as it usually takes a tremendous amount of time for full covergence, especially when the size of the colony is large as well. Therefore, our stopping criteria only checks for partial convergence. To be more specific, the algorithm would be terminated immediately, when the top solutions of a colony converge. The reason why we only consider the top solutions is that they have the most powerful influence on the updating of pheronome information. Our the stopping criteria enable interaction between the user and the computer, and let the user decide where to stop. The stopping criteria comprise two parts:

1. Best Solution. When the best solution achieved so far has not changed for three colonies, the program would be stopped temporarily and the interaction part would be triggered.

2. Interaction. The better half of the two recent colonies are listed. If the convergence is clear, then the user can stop the program permenantly, otherwise, the user can give the computer ‘carry on’ instruction and refresh the stopping criteria.

One can notice that the only difference between the stopping criteria of ACO and thoes of GAs is the interaction part. Although the *Best Solution* is usually good enough, we cannot entirely rely on it, becuase sometimes we would ternimate the algorithm too early. The following table 5-4 shows an example, which is extracted from the computing results of the FTSE-15.

Table 5-4: Stopping criteria illustration

Ants No.	Colony			
	12th	13th	14th	15th
1	0.078392	0.078392	0.078392	0.078392
2	0.08318	0.08318	0.0788	0.0788
3	0.0839	0.0839	0.0818	0.0818
4	0.084177	0.084177	0.08318	0.08318
5	0.08458	0.08458	0.0839	0.083641
6	0.087442	0.087442	0.084047	0.083648
7	0.08771	0.08771	0.084177	0.0839
8	0.087832	0.087832	0.08458	0.084047
9	0.091045	0.090299	0.088751	0.087966
⋮	⋮	⋮	⋮	⋮
n	0.105788	0.103961	0.100299	0.098511

Although the best solution is unchanged from the 12th colony to the 15th colony, the 15th colony has not converged yet. If the algorithm is terminated at the 15th colony, we would get a very poor tracking portfolio.

Our ACO algorithm comprises five elements in total: Construct Ants Solutions, Update Pheromones, Daemon Actions, Mutation, and Stopping Criteria. The following are the steps of the ACO algorithm:

- *Generate an initial ant colony*
- *Evaluate the objective value of each route used by the colony*
- *Deposit pheromones biased to objective values*
- *Repeat*
  - *Use the heuristic information to choose the first move*
  - *Use the attraction table to instruct the rest of the moves*
  - *Mutation*

  - *Update the pheromone information*

- *Until*
  - *The user decides when to stop and report the best solution achieved*

5.3 Parameters Investigation

When we use the ACO algorithm to solve the Index Tracking problem, we notice that the changing of some parameters can have a big influence on the computing results. Particularly, there are two parameters: the colony size and the evaporation rate which play decisive roles in searching good tracking portfolios. In the following, we investigate on the size of the colony and the evaporation rate, and illustrate how they affect both the solution qualities and the computing time. Note here that, the investigations are only implemented for the first five indices as they are sufficient enough to give general conclusions.

- **Colony Size**

In the investigation, we keep other parameters unchanged and observe how the computing results change as the colony size changes. We use the same way to generate colonies as we generate the initial population for the R-GA. This time we generate seven groups: f6, f9, f12, f15, f18, f21, and f24. The details are given below (we set the evaporation rate equal to 0.5):

Table 5-5: Colony size for each index under different frequencies

Frequency	Cardinality	Colony Size				
		Hang Seng	DAX	FTSE	S&P	Nikkei
f6	10	19	51	53	59	135
	15	12	34	36	39	----
	20	----	26	27	29	----
f9	10	28	77	80	88	203
	15	19	51	53	59	----
	20	----	38	40	44	----
f12	10	37	102	107	118	270
	15	25	68	71	78	----
	20	----	51	53	59	----
f15	10	47	128	134	147	338
	15	31	85	89	98	----
	20	----	64	67	74	----
f18	10	56	153	160	176	405
	15	37	102	107	118	----
	20	----	77	80	88	----
f21	10	65	179	187	206	473
	15	43	119	125	137	----
	20	----	89	93	103	----
f24	10	74	204	214	235	540
	15	50	136	142	157	----
	20	----	102	107	118	----

The following table shows the computing results of the above groups. For the reason of space, we round up the tracking errors to the fifth decimal place and the computing time to the first decimal place.

Table 5-6: Computational results under different input frequencies

Index	K	Computational Results													
		$population = f6$		$population = f9$		$population = f12$		$population = f15$		$population = f18$		$population = f21$		$population = f24$	
		TE	CT	TE	CT	TE	CT	TE	CT	TE	CT	TE	CT	TE	CT
Hang Seng	10	0.10101	5.2	0.10425	7.8	0.10215	18.7	0.10101	23.3	0.10101	22.5	0.10101	23.6	0.10101	49.0
	15	0.05689	8.5	0.06046	11.2	0.05689	13.4	0.05689	11.6	0.05689	15.1	0.05689	16.8	0.05689	27.7
DAX	10	0.07979	35.7	0.08350	32.7	0.06856	42.3	0.06907	85.3	0.06907	78.7	0.06730	119.3	0.06907	170.0
	15	0.04699	26.9	0.03747	36.0	0.03824	46.6	0.03827	73.7	0.04202	66.9	0.03860	90.3	0.03515	118.2
	20	0.02848	43.6	0.03114	54.7	0.03109	68.6	0.02613	95.5	0.02631	156.3	0.02593	145.9	0.02507	161.8
FTSE	10	0.13625	31.3	0.12290	47.1	0.11264	77.2	0.12243	94.6	0.11421	133.2	0.12142	109.3	0.11517	168.0
	15	0.08286	35.8	0.07670	64.1	0.06674	80.0	0.05756	73.2	0.06845	159.1	0.05791	139.2	0.05101	194.0
	20	0.04412	33.4	0.04653	52.3	0.04333	81.0	0.04260	116.9	0.03733	116.6	0.04599	181.9	0.03376	174.8
S&P	10	0.11213	44.6	0.11833	55.9	0.11331	92.8	0.10736	132.9	0.11443	88.7	0.11048	178.0	0.10000	177.3
	15	0.05528	40.9	0.06232	57.4	0.06762	117.5	0.06022	67.0	0.06552	154.9	0.06165	142.4	0.06094	182.7
	20	0.04191	33.1	0.05063	70.5	0.03871	113.4	0.04058	122.9	0.03649	94.4	0.04130	132.2	0.03317	181.2
Nikkei	10	0.12301	79.0	0.11379	182.9	0.10457	161.5	0.09182	333.3	0.09908	329.0	0.09259	254.2	0.09259	434.8

Generally, the larger the colony size the better the solution we can achieve. For example, the best solutions of DAX-20, FTSE-20, and S&P-20 are all achieved from f24, which are 0.025066372, 0.033761488, and 0.033167268, respectively. On the downside, f24 costs the most amount of computing time among the seven groups. We conclude theoretically the colony size for each problem should be set as large as possible, but practically it should be acceptably large so that the computing time and the solution quality can be balanced.

- **Pheromone Evaporation Rate**

The evaporation rate can also influence the pheromone information, as it has the function of forgetting the past information. In real ant colonies, the density of pheromones decreases over time by evaporation but it does not play an important role in finding the shortest path. However, the evaporation is very critical for the artificial ants in searching for the optimal solution. The idea of investigating the evaporation rate is enlightened by the Extend Double Bridge Experiment (Dorigo M, Stützle T, 2004). In this experiment, the ants start from the 'nest' node. They can choose between the upper and the lower parts of the graph. If they choose the upper part of the graph, they can always reach the destination by going along a path of eight lengths, while if they choose the lower part, they may find paths shorter than eight lengths, but at the same time they may go along a path of more than eight lengths.

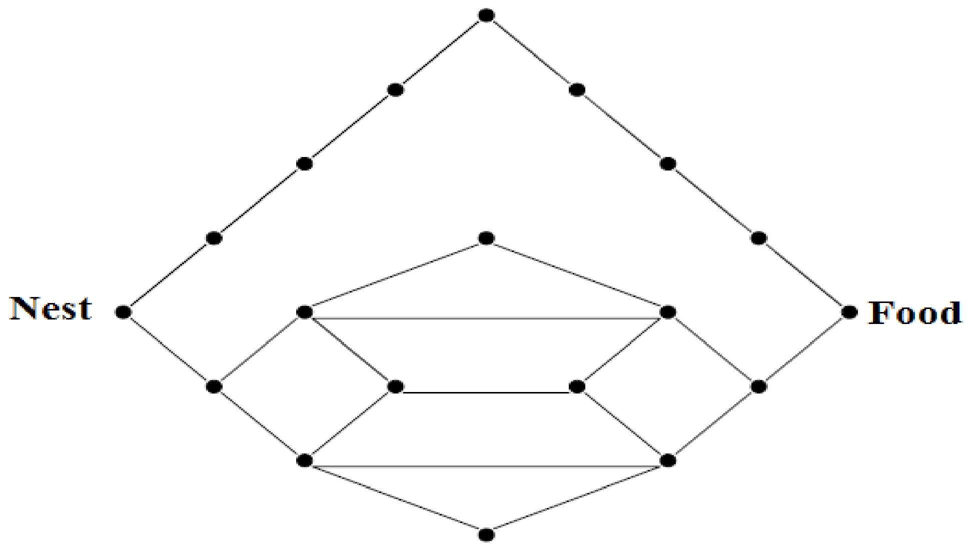


Figure 5.6: Extended double bridge experiment

During the experiment, the researchers ran it with different evaporation rate $\rho = \{0, 0.1, 0.01\}$ and the following graph in figure shows the relationship between the number of completed path (x-axis) and the moving average of the length of the path (y-axis). When the evaporation rate is zero, the algorithm does not converge, because the moving average is about 7.5, which is not related to either of the paths plotted. If the evaporation rate is 0.01 or 0.1, the algorithm converges to a single path. However, the evaporation rate is the greatest of the three, the moving average is 6 which is a sub-optimal solution. In contrast, when the evaporation rate is 0.01 the algorithm converges to the shortest route in all trials. In the investigation, we run the algorithm with four different evaporation rates $\rho = \{0.05, 0.1, 0.5, 0.9\}$ to find out how they affect the computing results. For the colony size, we use the data from f24. Computing results are given below. For the reason of space, we round the tracking errors up the fifth decimal place and the computing time to the first decimal place.

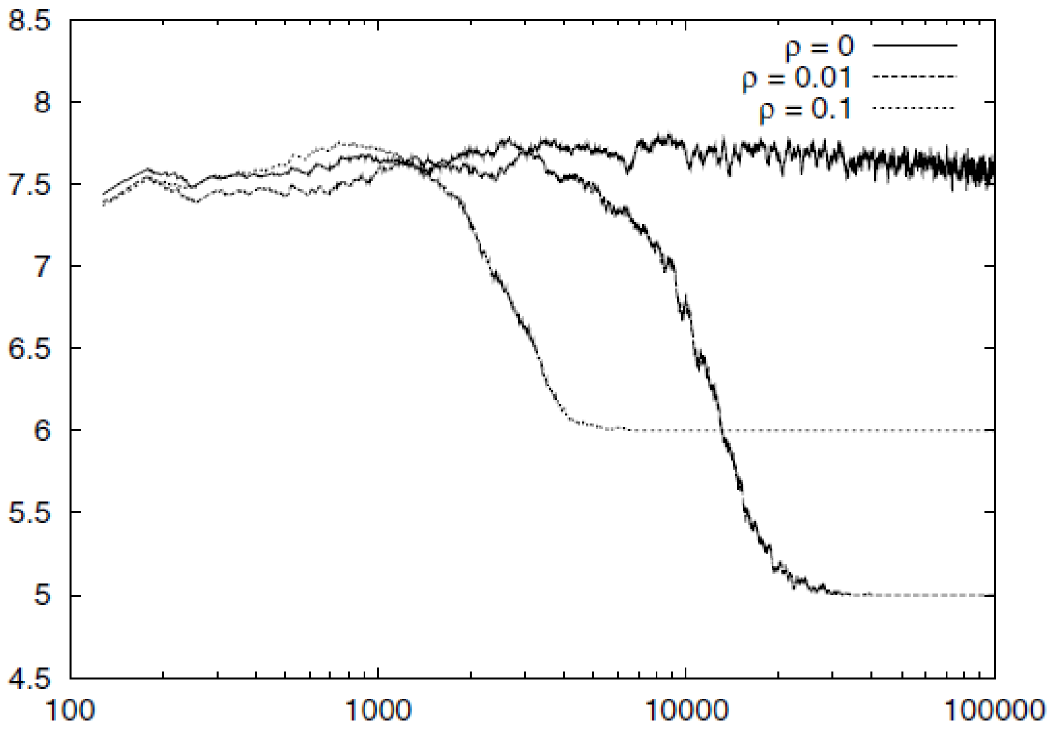


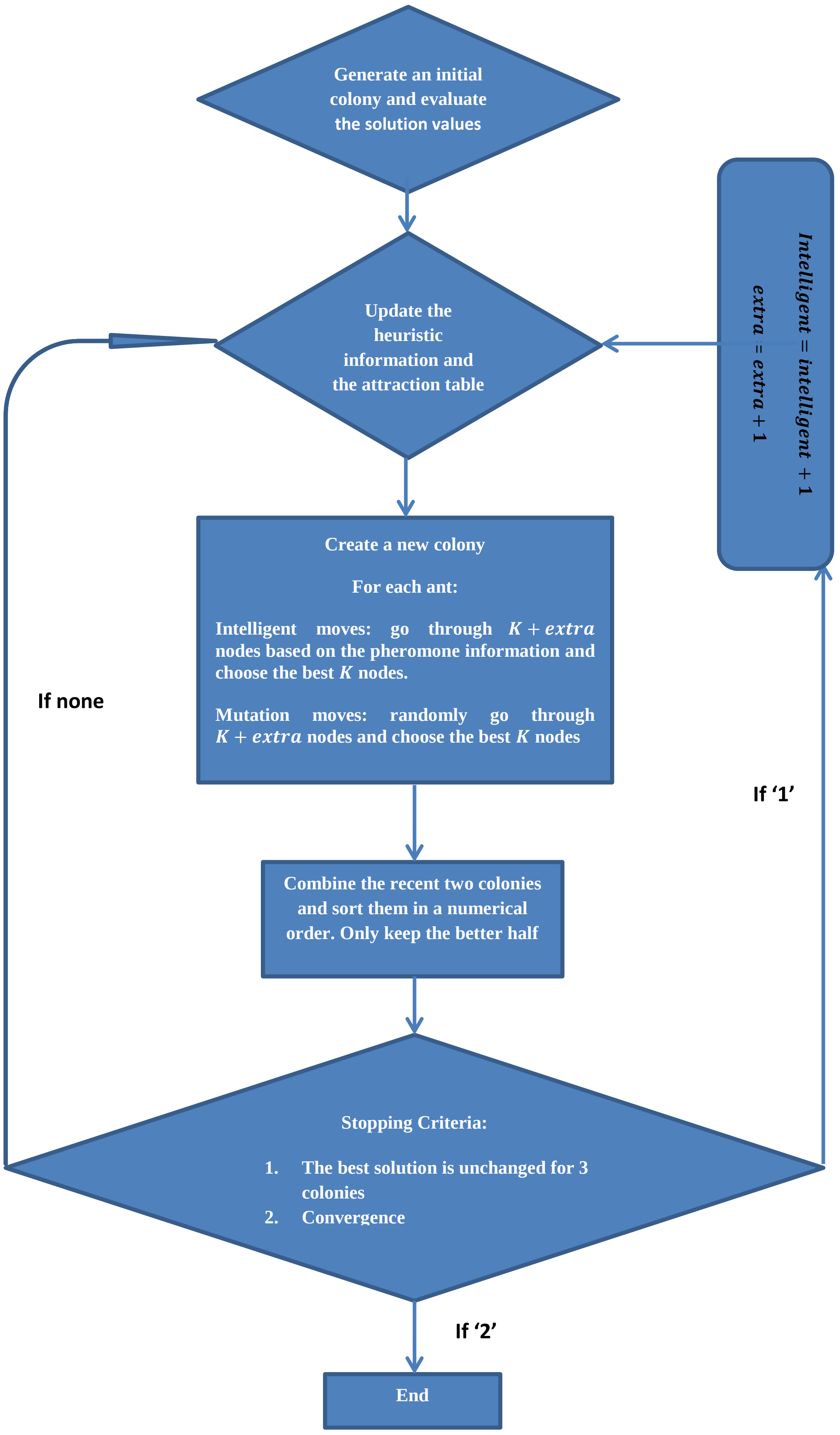
Figure 5.7: The results of the extended double bridge experiment

Table 5-7: Computational results under different evaporation rates

Index	K	Computational results							
		0.05		0.1		0.5		0.9	
		TE	CT	TE	CT	TE	CT	TE	CT
Hang Seng	10	0.10101	41.0	0.10101	34.8	0.10101	49.0	0.10101	36.4
	15	0.05689	36.4	0.05689	37.9	0.05689	27.7	0.05689	25.5
DAX	10	0.07361	286.1	0.07413	339.4	0.06907	170.0	0.06907	112.2
	15	0.03790	229.1	0.04032	285.5	0.03515	118.2	0.03747	138.7
	20	0.02052	366.2	0.02036	258.2	0.02507	161.8	0.02627	92.5
FTSE	10	0.11517	310.3	0.11517	261.7	0.11517	168.0	0.11801	130.1
	15	0.07411	313.0	0.07240	247.7	0.05101	194.0	0.07710	131.1
	20	0.03650	414.7	0.03341	320.7	0.03376	174.8	0.03373	159.9
S&P	10	0.10000	219.3	0.09965	261.2	0.10000	177.3	0.09965	110.0
	15	0.06022	651.1	0.06022	477.4	0.06094	182.7	0.06220	128.8
	20	0.03983	478.1	0.02956	456.9	0.03317	181.2	0.03111	103.4
Nikkei	10	0.09460	552.4	0.09259	391.4	0.09259	434.8	0.09722	260.7

When we set the evaporation to 0.9, the algorithm converges very quickly and costs the least amount of the computing time, but it usually produces the worst solutions. If we set the evaporation rate to 0.5, the algorithm is still able to converge at most of the time, but it only can find good solutions for the two smallest indices: Hang Seng and DAX. As for the three larger indices, the solutions are poor. When we set the evaporation rate to 0.1, the algorithm often produces good solutions. Particularly, it finds the best solutions among the four for the tracking portfolios with big cardinalities such as: DAX-20, FTSE-20, and S&P-20. When the evaporation rate is 0.05, the algorithm hardly converge and costs most amount of computing time. Regarding both the solution quality and the computing time, we conclude that the evaporation rate should be set to 0.1.

Even after finding the most suitable evaporation rate, however, the computing results still do not meet our expection. In order to improve the solution quality, we decide to make several changes on the mutation rate and the stopping criteria. Originally, the mutation process is a stochastic event that happens with a probability of 0.02, but we adjust the mutation rate to 1 to make it a deterministic event. From another point of view, each ant is able to make an extra number of moves on the construction graph by twice, which make them even more powerful in finding good soltuions. In the first time, the extra number of moves is guided by the pheromone trails (named intelligent moves), and in the second time, the extra number of moves are purely random (named mutation moves). In addition, the two times of extra moves play different functions in the algorithm. The intelligent moves can correct the wrong decisions which ants probabilistically make based on the pheromone concentrations. The mutation moves are used to maintain the diversity in the algorithm. For the stopping criteria, we add more features in the interaction part, where we not only can examine the convergence, but also can increase the the number of intelligent moves and mutation moves. The following shows the flowchart of the ACO.



Generate an initial colony and evaluate the solution values
Update the heuristic information and the attraction table
$intelligent = intelligent + 1$
$extra = extra + 1$
Create a new colony
For each ant:
Intelligent moves: go through $K + extra$ nodes based on the pheromone information and choose the best K nodes.
Mutation moves: randomly go through $K + extra$ nodes and choose the best K nodes
If none
If '1'
Combine the recent two colonies and sort them in a numerical order. Only keep the better half
Stopping Criteria:
1. The best solution is unchanged for 3 colonies
2. Convergence
If '2'
End

5.4 Computational Results

As the artificial ants become more powerful, we are able to play a trade-off between the size of the colony and the solution quality. Therefore, we use f15-10 (frequency 15 with cardinality of 10) as the colony size for the first five indices, which are 47, 128, 134, 147, and 338 respectively. For the last three indices, as the attraction tables are too large, which contain $457 \times 457, 1318 \times 1318$ and 2151×2151 elements respectively. Therefore, we have to further cut the colony size. By emperical experience, we set the colony size to 150, 75, and 30 for S&P 500, Russell 2000, and Russell 3000 respectively. Table 5-8 shows the comparisons between the computing results of ER-GA and ACO. (*Intelligent* and *Mutation* represent the number of intelligent moves and mutation moves respectively and *size* represents the colony size.

Table 5-8: ER-GA vs ACO

Index	K	ER-GA			ACO				
		TE	CT	Population	TE	CT	Intelligent	Mutation	size
Hang Seng	10	0.10101	17	37	0.10101	21	2	5	47
	15	0.05689	9	25	0.05689	16	2	5	47
DAX	10	0.0673	115	102	0.0673	127	2	5	128
	15	0.03515	312	68	0.03515	354	2~4	5~7	128
	20	0.02021	540	51	0.02036	302	2~5	5~8	128
FTSE	10	0.09982	135	107	0.09842	98	2~3	5~6	134
	15	0.05664	429	71	0.05703	272	2~4	5~7	134
	20	0.02906	980	53	0.02858	351	2~4	5~7	134
S&P	10	0.09965	175	118	0.09935	366	5~6	5~6	147
	15	0.0504	626	78	0.0504	573	5~8	5~8	147
	20	0.02617	1009	59	0.02873	690	5~7	5~7	147
Nikkei	10	0.08057	784	270	0.09259	490	7	7	338
	15	0.04269	2197	180	0.04736	1374	7~9	7~9	338
	20	0.0269	1041	135	0.02461	1721	7~9	7~9	338
	25	0.01125	2336	108	0.02283	2413	7~12	7~12	338
S&P 500	10	0.0726	1444	548	0.08201	517	10~11	10~11	150
	15	0.03502	4369	366	0.03214	1541	10~11	10~11	150
	20	0.01738	3779	274	0.02145	2375	10~12	10~12	150
	25	0.01021	3140	219	0.01059	2300	10~13	10~13	150
	30	0.00924	3633	183	0.00846	2957	10~17	10~17	150
Russell 2000	30	0.01072	2640	527	0.00762	4714	15~17	15~17	75
	40	0.00401	1809	395	0.00478	1380	15~21	15~21	75
	50	0.00043	1148	316	0.00056	5650	15~21	15~21	75
	60	0	535	264	0	2797	15~16	15~16	75
	70	0	456	226	0	785	15	15	75
Russell 3000	30	0.00794	5727	860	0.00674	1210	25	25	30
	40	0.00198	5875	645	0.00155	3305	25~26	25~26	30
	50	0.00059	1851	516	0.00009	3853	25~27	25~27	30
	60	0	893	430	0	1283	25	25	30
	70	0	771	369	0	741	25	25	30
	80	0	1358	323	0	1819	25	25	30

In the above table, expressions such as ‘2~3’ / ‘5~6’ mean that initially ants are allowed to take 2 ‘intelligent’ moves and 5 ‘mutation’ moves, and the number of moves are increased through the searching process until they finally reach 3 and 6 respectively.

Again, we separate the comparison by two groups: the first four indices and the last four inidces. In the first group, ACO beats or loses to ER-GA three times each, and they draw 5 times, in terms of solution quality. The two solution approaches are equally good for solving small size indices, but ACO is slightly better, because it spends about 1176 time units less that the ER-GA does. In the second group, ACO beats ER-GA seven times, and it loses to ER-GA eight times, and they draw five times, in terms of solution quality. The two solution approaches spend almost the same amount of computing time.

Overall, we conclude that the ACO algorithm works slightly better for the small size index while the ER-GA is more productive for the large size index. Generally, the two solution approaches are equally good. In addition, we find that the ACO alogrithm works better either with powerful ants and small size colonies or less powerful ants and large size colonies.

Chapter 6

6 .Post Analysis

6.1 Tracking Portfolios Investigation

6.1.1 Investigation Method

In the previous chapters, we use several solution approaches to deal with the Index Tracking problem. Whatever solution approaches used, we actually try to solve two fundamental questions.

1. Which stocks should be selected to form a good tracking portfolio for a given index?
2. How much proportion should be given to the stocks in the tracking portfolio?

To answer the above questions, we need to use the following illustration.

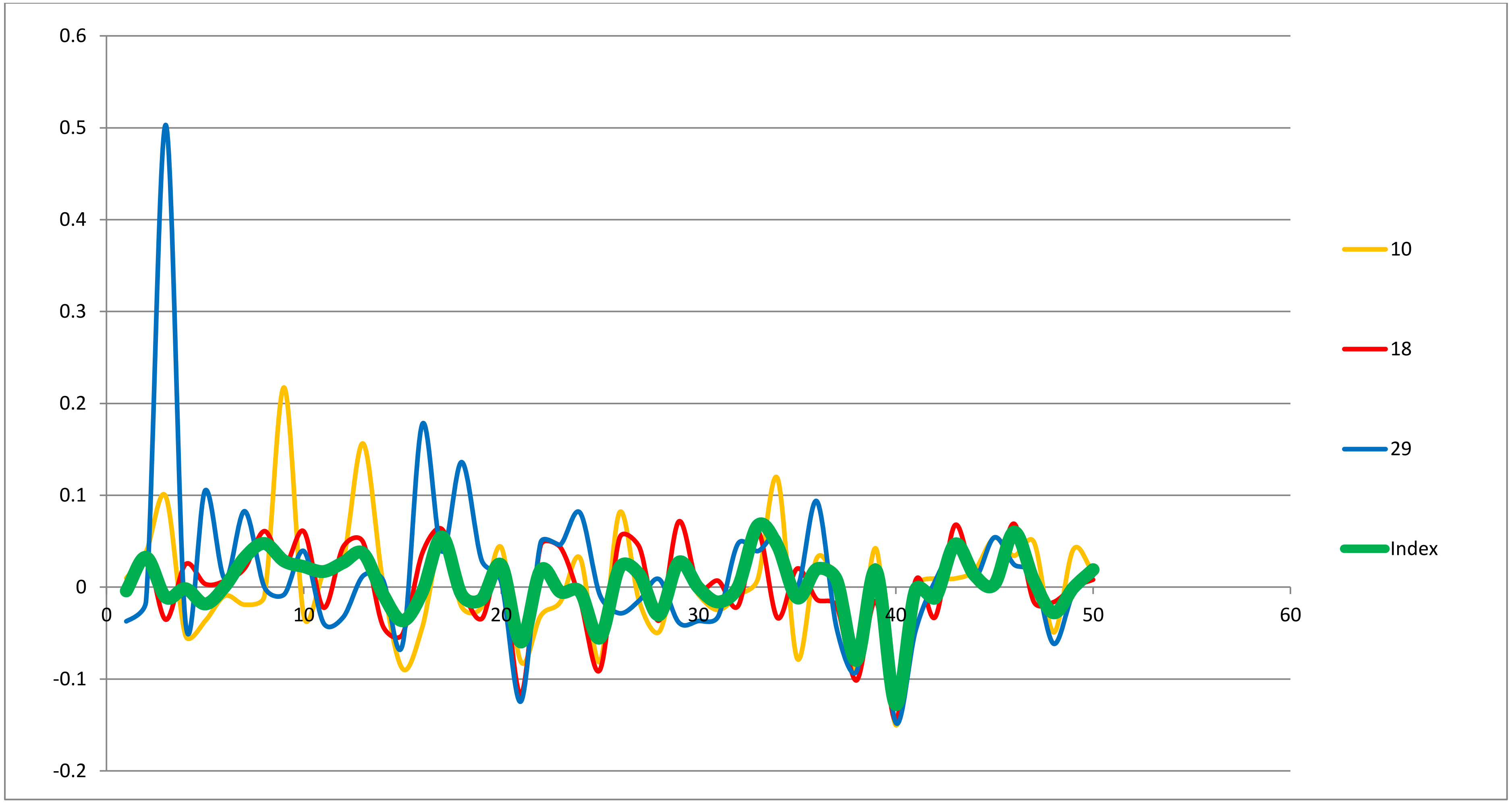


Figure 6.1: The return of Hang Seng index and its stocks

(The data used for the above illustration is from the Hang Seng index. For the reason of space and clarity, we only depict return of the index ($T = 1 \dots 50$) and compare it with three stocks from the index return, stock 10, stock 18, and stock 29.) Considering an extreme case, where a tracking portfolio is only composed by a single stock, which stock would you pick among the three? Arguably, everyone would pick stock 18, because stock 18 has the most similar curve to that of the index. Consider another case, where we need to select K stocks to form a tracking portfolio. Assume that we know a particular stock H whose curve is extremely close to that of the index. How should we pick the stocks? It might cost us tremendous amount of time to find the optimal tracking portfolio, but to find a fairly good one, we could select the stock H and assign it with a very large weight, and then randomly pick the rest stocks. Base on the above assumption, if we can find the stock H in an index, then we can form a good tracking portfolio. Therefore, we investigate all the stocks in an index. Before introducing our investigation method, we need to define two notations:

$D_{i,t}$	The difference between the return of the index and the return of the stocks i at time t	
$E_i = \sum_t \lvert D_{i,t} \rvert$	$t = 1 \dots T$	(name it tracking error of stocks i)

In our investigation, we first calculate the tracking error of each single stock in an index and sort their tracking errors from the smallest to the largest, and then we investigate each best tracking portfolio achieved to answer the following questions:

1. Is a good tracking portfolio mainly composed of stocks whose return curves are close to that of the index (name this kind of stocks ‘close’ stocks)?
2. If yes, are they assigned with large weights?
3. How does the number of close stocks change as the cardinality changes?

We investigate the Hang Seng index first. The close stocks and the optimal portfolios found by MIP are shown in table 6-1 and table 6-2, respectively.

Table 6-1: Close stocks of the Hang Seng index

Sequence	Assets	Tracking Error
1	30	0.836184044
2	27	0.863627405
3	12	0.906704067
4	11	0.927586793
5	28	0.936674507
6	22	0.937887916
7	24	0.943844523
8	26	0.989549926
9	3	1.023770649
10	21	1.030483113
11	13	1.037880974
12	31	1.051200381
13	18	1.053314154
14	6	1.057315272
15	4	1.067125077
16	5	1.067304409
17	14	1.082638143
18	20	1.106248613
19	15	1.115634801
20	7	1.226111461
21	8	1.245387753
22	2	1.25439917
23	25	1.328835313
24	17	1.398357754
25	19	1.415774641
26	1	1.4180224
27	9	1.428918746
28	10	1.640887392
29	23	1.779078158
30	16	2.184850543
31	29	2.485006209

Table 6-2: Optimal portfolio of the Hang Seng index with K=10 and K=15

K=10		K=15	
assets	Weight	assets	Weight
11	0.142085359	4	0.056649571
12	0.126630675	11	0.128981463
15	0.129587587	12	0.086145825
18	0.094112224	13	0.057174344
21	0.143762182	14	0.021298088
22	0.130762849	15	0.123751574
23	0.036352913	18	0.061180918
25	0.043006333	21	0.092843515
26	0.071053615	22	0.076711982
27	0.082646264	23	0.024972322
		25	0.026960195
		26	0.063606179
		27	0.077692952
		28	0.058045392
		31	0.043985679

Six of the stocks in the optimal portfolio are from the top 10 ‘close’ stocks, which are stocks 27, 12, 11, 22, 26, and 21 (write this as $CS_{10} = 6$ and name it the ‘close’ stock effect). The total weight of these stocks takes 70% of the whole portfolio. The portfolio tacking error is about 0.101, which is almost eight times smaller than that of the best ‘close’ stock, stock 30. Therefore, it proves that the combination of the stocks produces positive effects to reduce the total tracking error of the portfolio, so we declare that the *combination effect* is 8 (write it as $B_e = 8$). As the cardinality increased to 15, eleven of the stocks in the optimal portfolio are from the top 15 ‘close’ stocks ($CS_{15} = 11$). Moreover, it inherits all the stocks from the previous optimal portfolio with cardinality of 10. Thus, we say the *inheritance effect* is 100% ($I_e = 100\%$). The total weight of them takes 80% of the whole portfolio, and the *combination effect* is almost 15.

6.1.2 Investigation Findings

Only investigating the Hang Seng index is not sufficient to summarize a general conclusion, because the size of the index is too small. Therefore, we would like to carry on our investigation on other four indices: DAX, FTSE, S&P, and Nikkei. For the reason of space, we do not list the table of the tracking error of each stock in the indices. The following table shows the *close stock effect*, *the combination effect*, and the *inheritance effect* of each index under different cardinalities. Notably, the following figures are derived from the best tracking portfolios we have achieved so far.

Table 6-3: Tracking portfolios investigation

Index	No. stocks	K	CS_n/W				B_e	I_e
			$n = K$	$n = N * 30\%$	$n = N * 50\%$	$n = N * 80\%$		
Hang Seng	31	10	6/70%	6/70%	7/79%	9/96%	8.3	-----
		15	11/80%	7/58%	11/80%	14/98%	14.7	100%
DAX	85	10	3/46%	6/78%	10/100%	10/100%	6.4	-----
		15	4/48%	6/70%	10/87%	13/96%	12.3	40%
		20	9/67%	9/67%	16/93%	18/96%	22.2	67%
FTSE	89	10	1/17%	3/45%	7/77%	10/100%	7.9	-----
		15	2/21%	3/30%	8/65%	14/97%	16.3	70%
		20	5/30%	6/38%	12/62%	19/97%	27.8	73%
S&P	98	10	2/30%	4/56%	7/82%	8/89%	6.7	-----
		15	5/41%	8/56%	10/72%	14/94%	12.8	30%
		20	7/48%	8/52%	10/66%	18/94%	23.8	100%
Nikkei	225	10	1/18%	5/63%	7/76%	10/100%	9.2	-----
		15	2/21%	8/70%	8/70%	13/92%	17.3	10%
		20	0/0%	6/40%	12/77%	17/92%	30.8	13%
		25	2/11%	6/40%	10/53%	18/75%	66.0	20%

CS_n/W represents the *close stocks effect* and the total percentage of weight the close stocks take of the tracking portfolio. Let's use the FTSE index for example; when the cardinality is 10, 1/17% means that only one stock is selected from the top K close stocks to form the tracking portfolio and that close stock takes 17% of the weight of the portfolio; 3/45% means that three stocks are selected from the top 27 ($89 \times 30\% \approx 27$) close stocks to form the tracking portfolio and they take about 45% weight of the portfolio. From the above table, we find that a good tracking portfolio is mainly composed of the top 80% 'close' stocks of an index. Averagely, these stocks take about 94% of the total weight of the tracking portfolio, which leaves a tiny proportion for the last 20% 'close' stocks. Among the close stocks of each index, the higher the rank of a close stock the larger the weight it is assigned with. For example, for the FTSE index with the cardinality of 10, only one stock is selected from the top 10 close stocks, but it takes about 17% of the total weight of the tracking portfolio; also, for the Nikkei index with cardinality of 10, the stock selected from the top close stocks takes 18% of the total weight. In contrast, the last 20% of the close stocks are given almost nearly the minimum weight (1%), or sometimes they are not even selected. (In reality, a close stock is more likely to be the stock that takes large percentage of capitalization of an index. For example, *Shell*, *BP*, and *HSBC* are more likely to be the close stocks in the FTSE index. However, our data do not specify the details about the companies and the industry sectors of the indices, so we are not able to make further discussion here.)

I_e represents the *inheritance effect*. For example, in the Dax-15, the 40% inheritance means that 40% of the stocks of the tracking portfolio with cardinality 10 are inherited by the tracking portfolio with cardinality 15.

B_e represents the *combination effect*, which shows that the bigger the cardinality the more the combination effect. Each time the cardinality increases by five units, the effect is increased by 1.89 times averagely. We believe the *combination effect* can be used to examine the quality of a tracking portfolio. For the same index, assume that the combination effect of a tracking portfolio with cardinality 15 is only 1.5 times bigger than that of a tracking portfolio with cardinality of 10, and then we could conclude that it is a poor tracking portfolio.

6.1.3 Further Application

Here, we would like to detail the *inheritance effect*, as we would like to apply it to our heuristic solution approaches. For the GAs, we could use it to generate tracking portfolios in a certain manner so that the evolution is able to start at a higher stage (maturation stage) instead of the initial stage. For example, if we have achieved a fairly good tracking portfolio of the DAX-10, to search for a considerable good tracking portfolio for the DAX-15, we can initially generate the tracking portfolios by combining the 10 stocks achieved from the DAX-10 with another random five stocks. The same method could be applied to the ACO to improve the searching ability of the algorithm, where we make the searching process 'half deterministic' and 'half probabilistic', see figure 6.2 below.

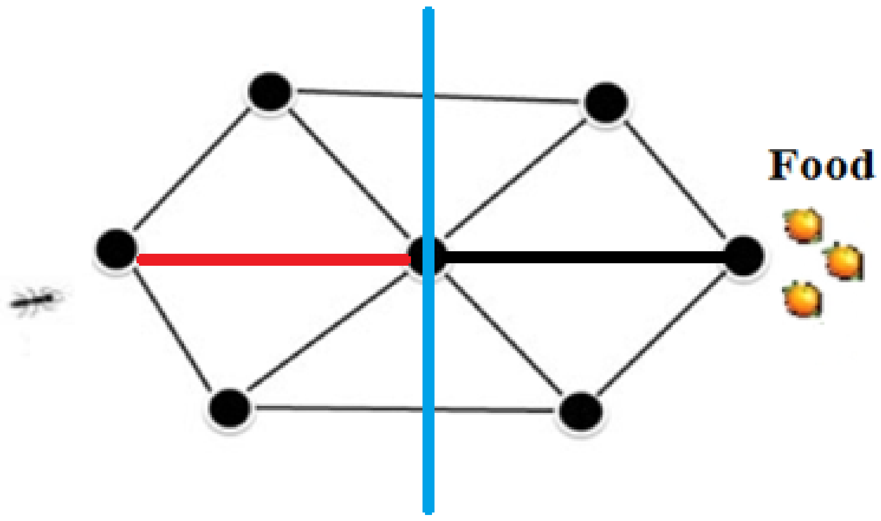


Figure 6.2: Half deterministic and half probabilistic searching

We cut the graph into two parts. Originally, the ants should probabilistically select the nodes on both parts of the graph; however, if we know that there is a good path (highlighted in red) on the left part, we could force the initial colony to follow that path to narrow down the searching space. By using the *inheritance effect*, we believe that we could further shorten the computing time for both of the solution approaches. On the other hand, we have more risk to achieve sub-optimal solutions.

6.1.4 Computational Results

We run the ER-GA and the ACO to test the above assumption. In the test, we keep all of the parameters the same using only the inheritance effect to search tracking portfolios with bigger cardinalities. Note here that, we do not implement the test on the Hang Seng index, because the computing time for both of the two scenarios is already very short. The following tables show the comparison of the computing results.

Table 6-4: Results comparison for GA

Index	K	ER-GA		Inherited ER-GA	
		TE	CT	TE	CT
Hang Seng	10	0.101010742	17.488	------	------
	15	0.056886088	8.735	------	------
DAX	10	0.067296843	115.232	------	------
	15	0.035154218	311.794	0.035154218	105.811
	20	0.020212963	540.419	0.023789522	57.375
FTSE	10	0.099823448	134.79	------	------
	15	0.056638474	429.034	0.048657473	146.794
	20	0.029058669	980.184	0.031018156	75.718
S&P	10	0.099653821	175.407	------	------
	15	0.050399037	626.029	0.053651399	53.679
	20	0.026170453	1009.138	0.030197377	232.575
Nikkei	10	0.080572321	784.257	------	------
	15	0.042686755	2197.451	0.042379989	172.668
	20	0.026897044	1041.193	0.02216469	153.71
	25	0.011254701	2336.498	0.009039572	459.677
S&P 500	10	0.072604735	1443.628	------	------
	15	0.035021586	4369.138	0.032574883	307.751
	20	0.017379756	3779.407	0.017645178	450.34
	25	0.010214859	3140.091	0.008291947	451.853
	30	0.009241603	3633.144	0.00420194	472.662
Russell 2000	30	0.010717636	2639.91	------	------
	40	0.004013996	1808.83	0.00363944	1157.31
	50	0.000427548	1147.72	0.00015756	1799.29
	60	0	534.66	0	149.114
	70	0	456.101	0	131.223
	80	0	606.544	0	140.771
Russell 3000	30	0.007936652	5727.32	------	------
	40	0.001983568	5874.664	0.002030017	2213.019
	50	0.000585532	1850.839	0.000324913	3853.34
	60	0	892.931	0	467.402
	70	0	770.754	0	231.437
	80	0	1358.35	0	948.711

Table 6-5: Results comparison for ACO

Index	K	size	ACO				Inherited ACO			
			TE	CT	Inteligent	extra	TE	CT	Inteligent	extra
Hang Seng	10	47	0.10101	21	2	5	------	------	------	------
	15	47	0.05689	16	2	5	------	------	------	------
DAX	10	128	0.06730	127	2	5	------	------	2	5
	15	128	0.03515	354	2~4	5~7	0.03515	163	2~3	5~6
	20	128	0.02036	302	2~5	5~8	0.02392	168	2~4	5~6
FTSE	10	134	0.09842	98	2~3	5~6	------	------	------	------
	15	134	0.05703	272	2~4	5~7	0.05611	111	2~3	5~6
	20	134	0.02858	351	2~4	5~7	0.03301	134	2~3	5~6
S&P	10	147	0.09935	366	5~6	5~6	------	------	------	------
	15	147	0.05040	573	5~8	5~8	0.05365	279	5~6	5~6
	20	147	0.02873	690	5~7	5~7	0.03429	111	5	5
Nikkei	10	338	0.09259	490	7	7	------	------	------	------
	15	338	0.04736	1374	7~9	7~9	0.05050	287	7	7
	20	338	0.02461	1721	7~9	7~9	0.02507	557	7	7
	25	338	0.02283	2413	7~12	7~12	0.01256	335	7	7
S&P 500	10	150	0.08201	517	10~11	10~11	------	------	------	------
	15	150	0.03214	1541	10~11	10~11	0.04554	316	10	10
	20	150	0.02145	2375	10~12	10~12	0.02372	304	10	10
	25	150	0.01059	2300	10~13	10~13	0.01114	290	10	10
	30	150	0.00846	2957	10~17	10~17	0.00511	518	10	10
Russell 2000	30	75	0.00762	4714	15~17	15~17	------	------	------	------
	40	75	0.00478	1380	15~21	15~21	0.00543	1918	15~18	15~18
	50	75	0.00056	2014	15~21	15~21	0.00127	985	15~16	15~16
	60	75	0	980	15~16	15~16	0	133	15	15
	70	75	0	454	15	15	0	148	15	15
Russell 3000	30	30	0.00674	1210	25	25	------	------	------	------
	40	30	0.00155	3305	25~26	25~26	0.00224	486	25	25
	50	30	0.00009	967	25~27	25~27	0.00070	613	25	25
	60	30	0	1283	25	25	0	89	25	25
	70	30	0	741	25	25	0	210	25	25
	80	30	0	1819	25	25	0	209	25	25

As shown in table 6-4, after changing the way in which the initial populations are generated, the Inherited ER-GA is superior in terms of computing time, as the total computing time is shortened by about 25463 time units. We also find that we do not have to limit each solving time for the Nikkei-225 and S&P-500, because *the inheritance* enables the evolution to start from the maturation stage, where large building blocks are already formed. For the solution quality, although the ER-GA works better with small indices (DAX, FTSE, and S&P), its computing time is four times as long as that of Inherited ER-GA. Also, Inherited ER-GA is better both in terms of solution quality and computing time when dealing with large indices.

The Inherited ACO also saves tremendous amount of computing time; 21802 time units less than that of the ACO, but the solution quality is generally compromised, shown in table 6-5.

Overall, we conclude that the *inheritance effect* is more suitable for the GA, as it could help the GA to largely shorten the computing time while producing better solutions for larger size indices.

6.2 Out-of-sample Performance

In the widespread research, many findings could simply be the product of a process called data mining, also known as data snooping. Data mining can unintentionally be misused, and can then produce results which appear to be significant while it actually cannot predict future behaviour and run on a new sample of data may bear little use. Simply by asking ourselves a question, can we look back on the data and find a portfolio that would almost perfectly re-perform the index? The answer is very positive. However, will this portfolio provide the same performance in the future? Perhaps not. It is always possible that enough data snooping can detect a portfolio that would have worked in the past by chance and fail to perform in the future. In order to check the stability of our solution approach, we decided to run simulations on the out-of-sample data. As stated in the previous part, our observation period is 0~50. We would like to know how our model performs out-of-sample. We choose observation times 51~60 as the out-of sample data to test if there is a huge discrepancy between the model prediction and the real index. The following figures show the performance:

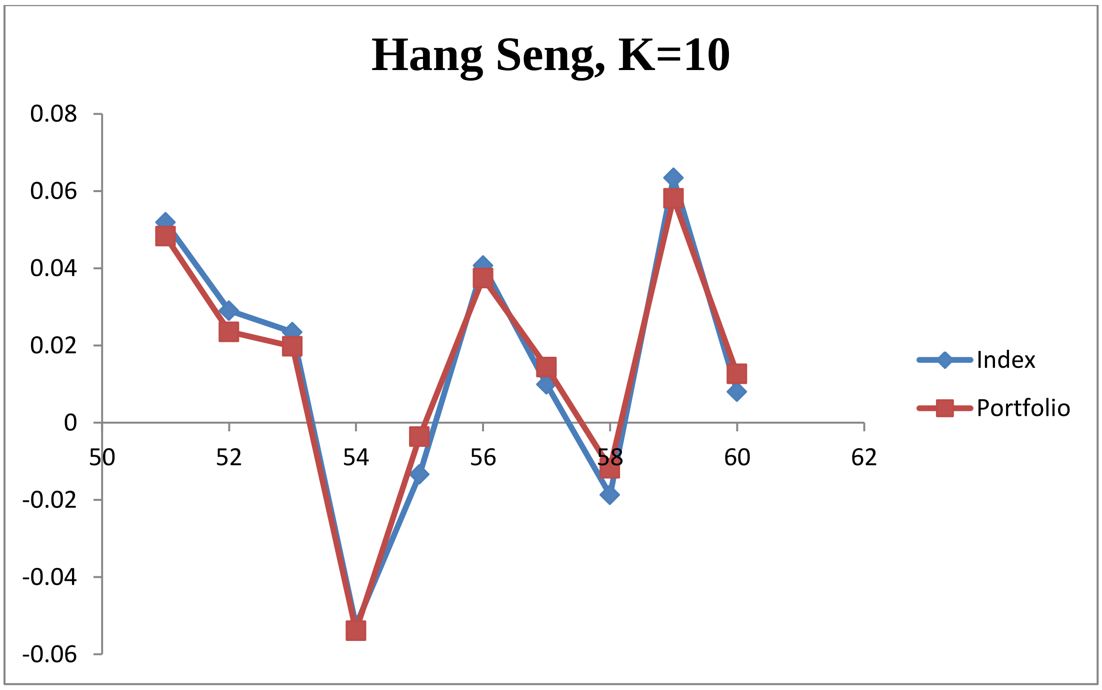


Figure 6.3: Hang Seng, K=10

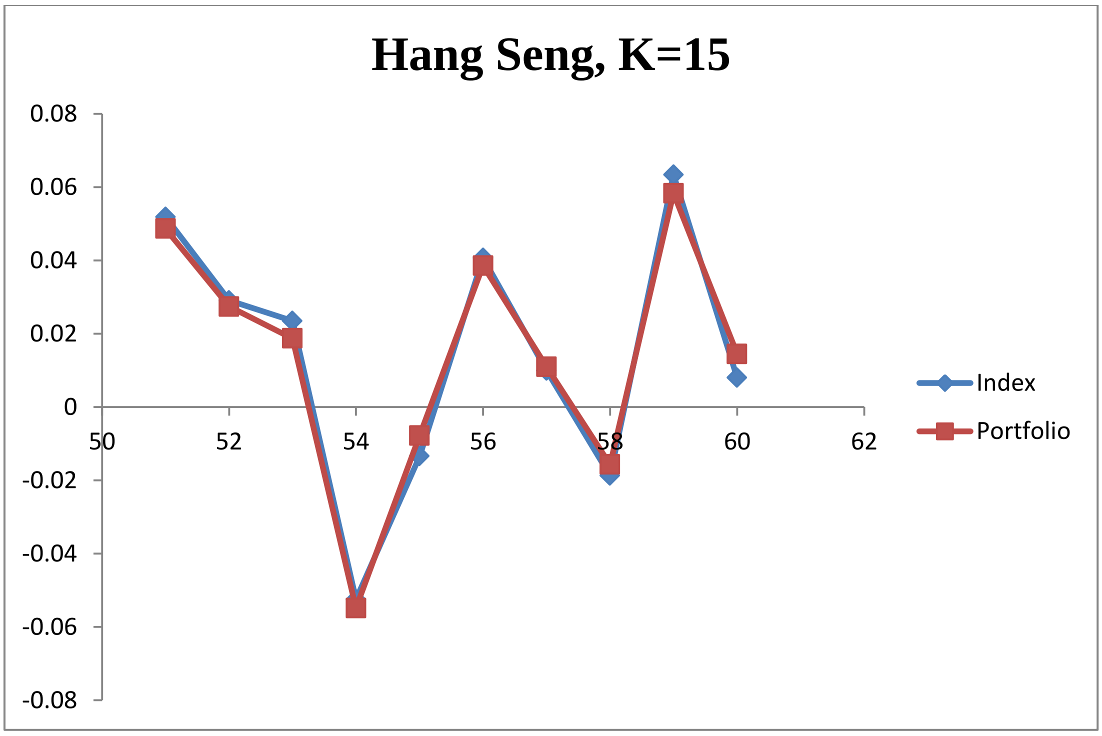


Figure 6.4: Hang Seng, K=15

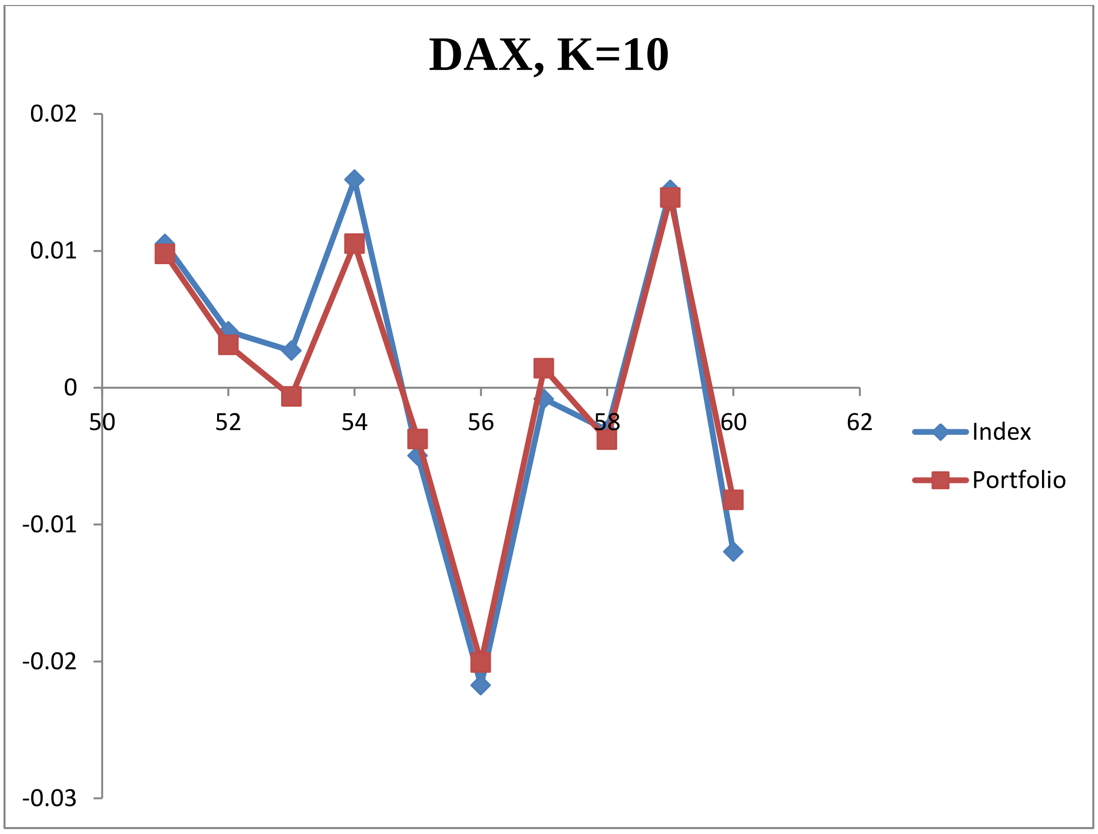


Figure 6.5: DAX, K=10

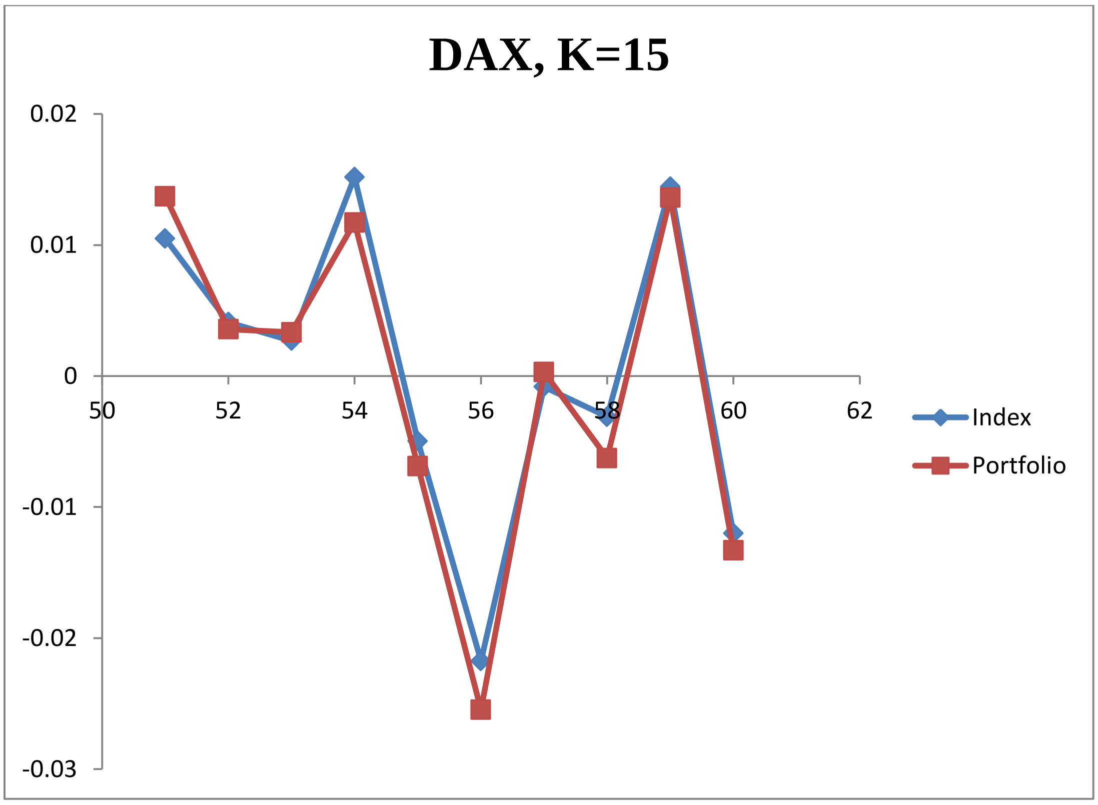


Figure 6.6: DAX, K=15

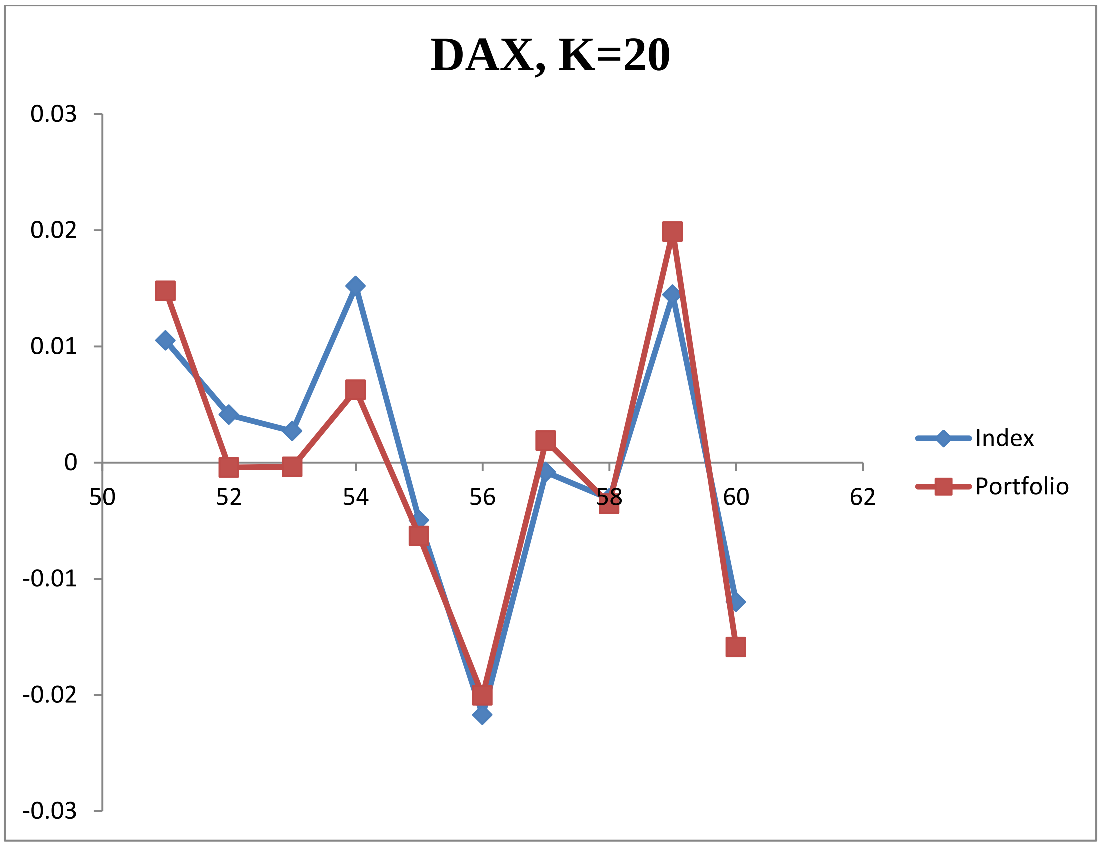


Figure 6.7: DAX, K=20

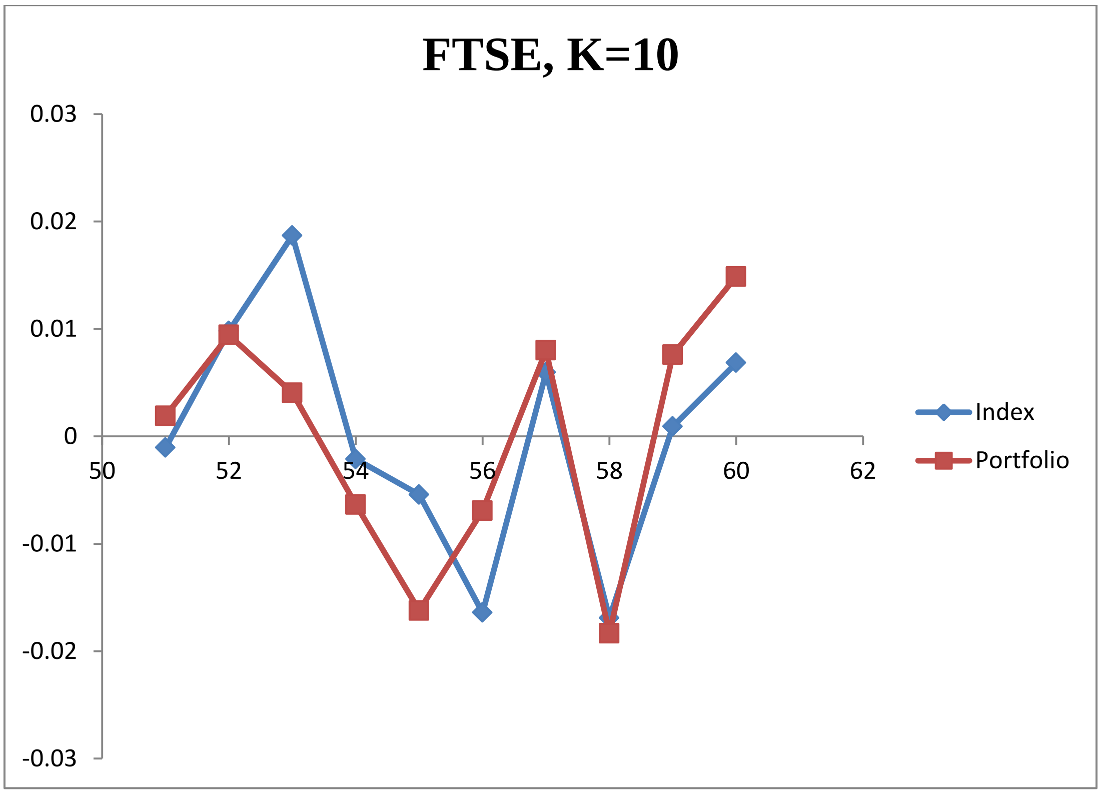


Figure 6.8: FTSE, K=10

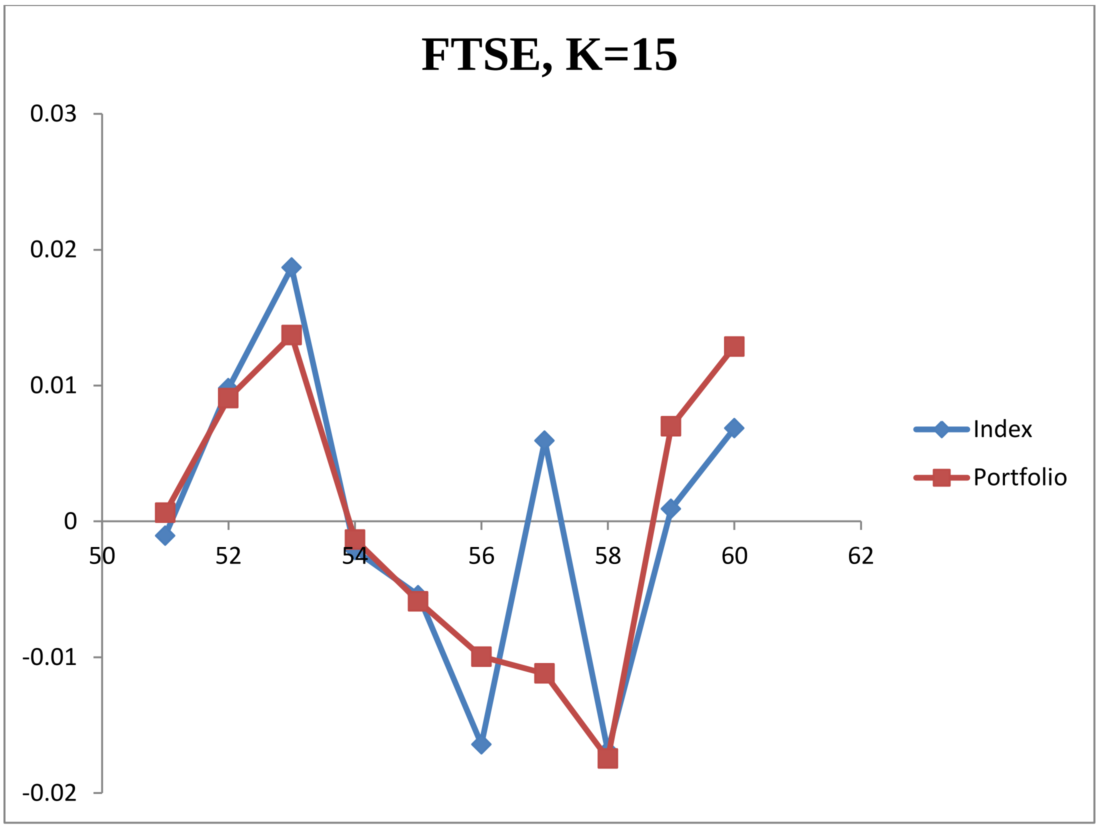


Figure 6.9: FTSE, K=15

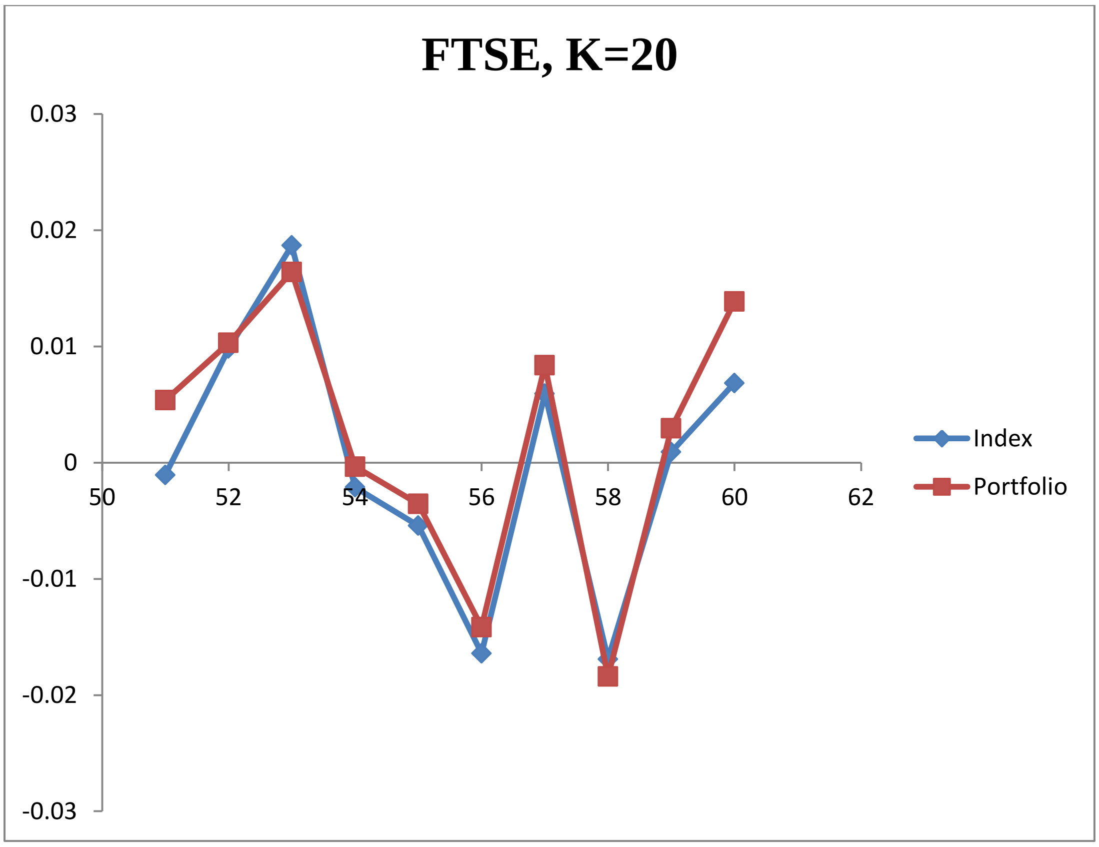


Figure 6.10: FTSE, K=20

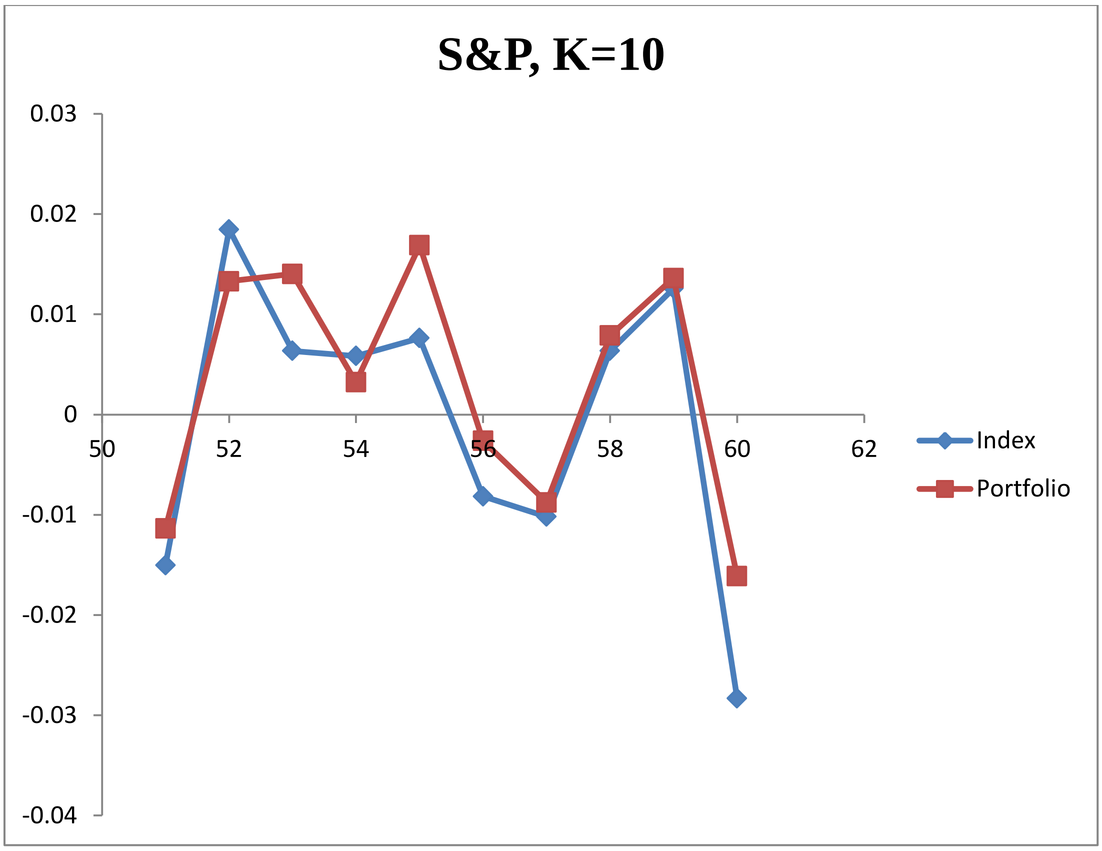


Figure 6.11: S&P, K=10

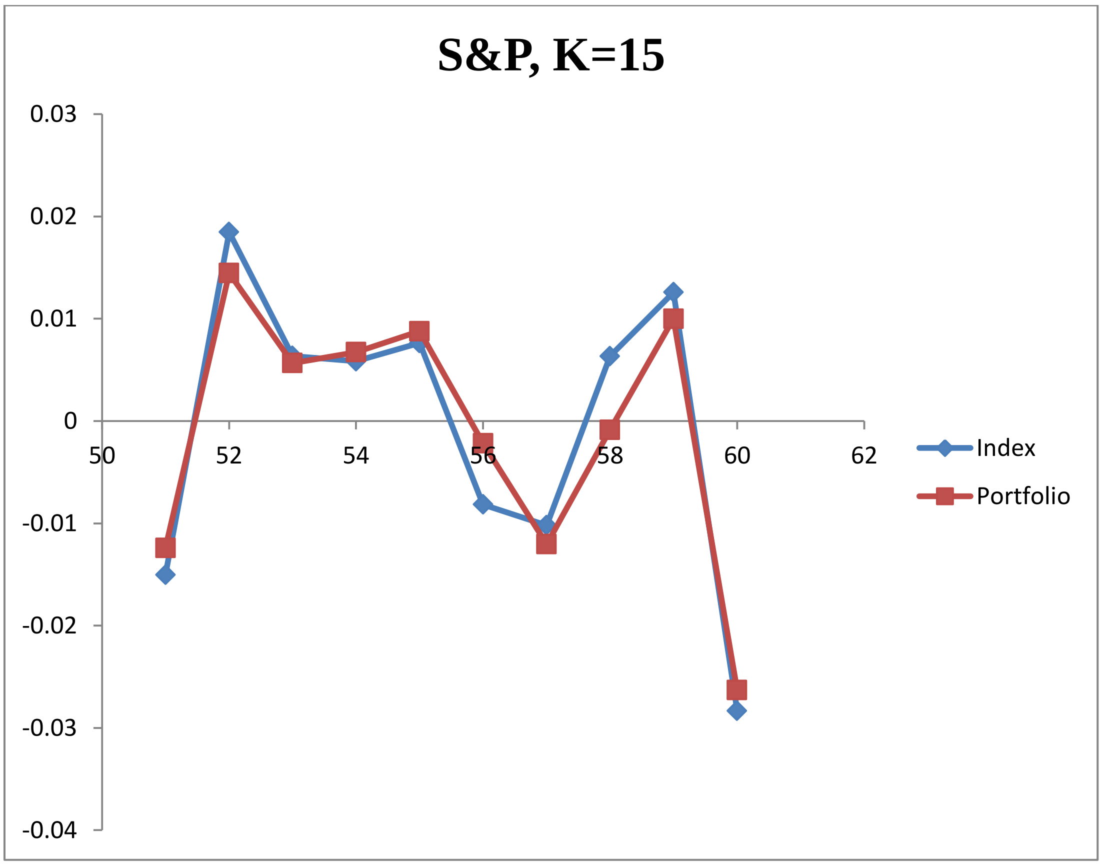


Figure 6.12: S&P, K=15

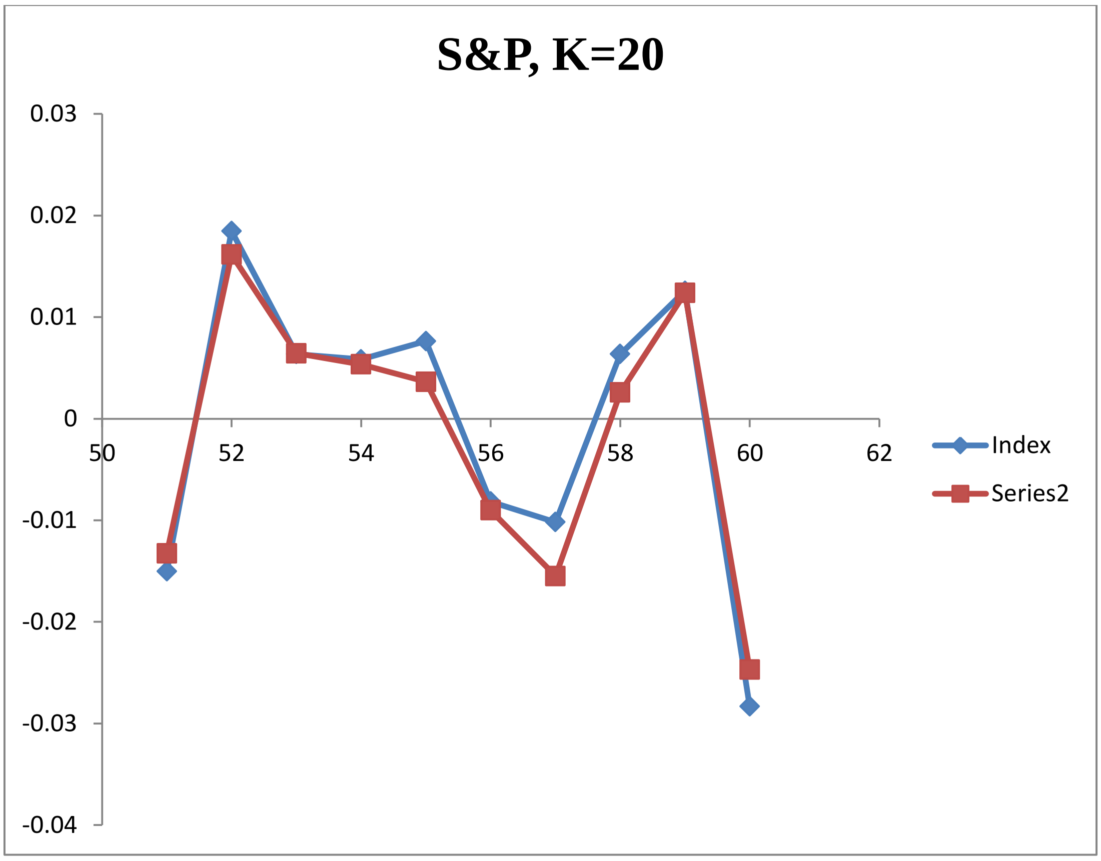


Figure 6.13: S&P, K=20

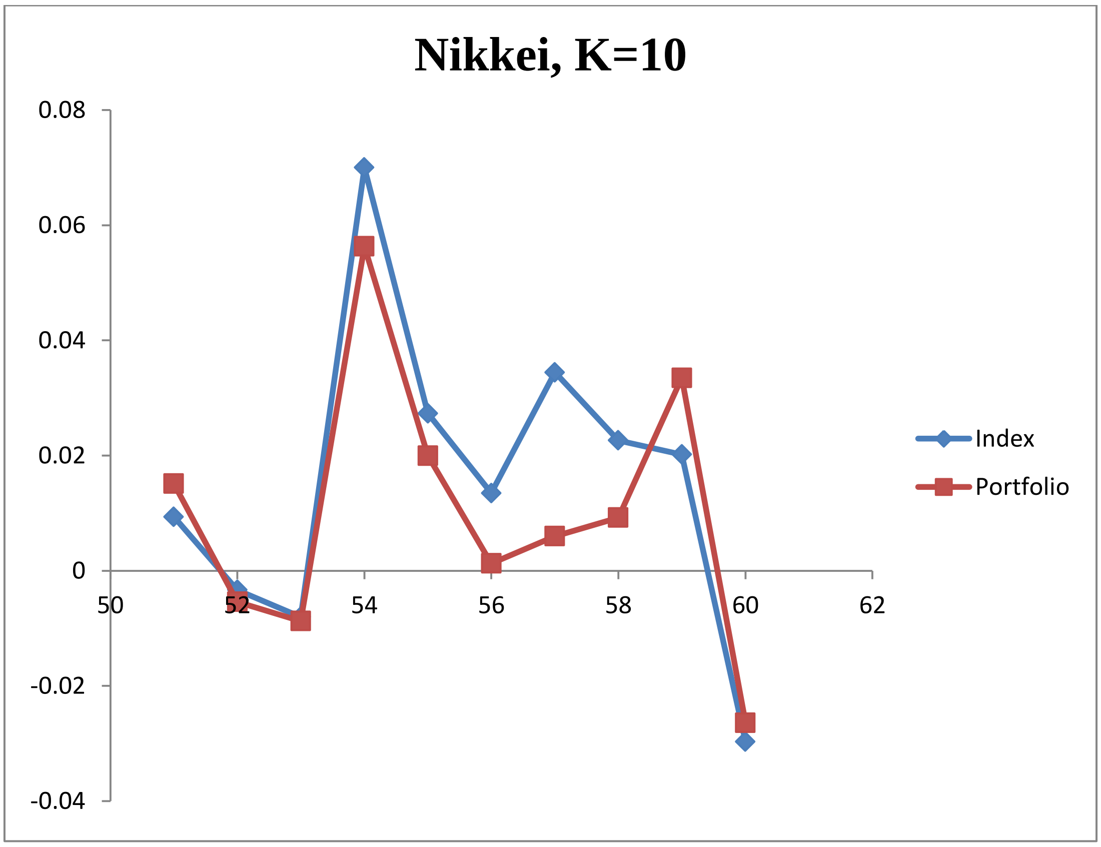


Figure 6.14: Nikkei, K=10

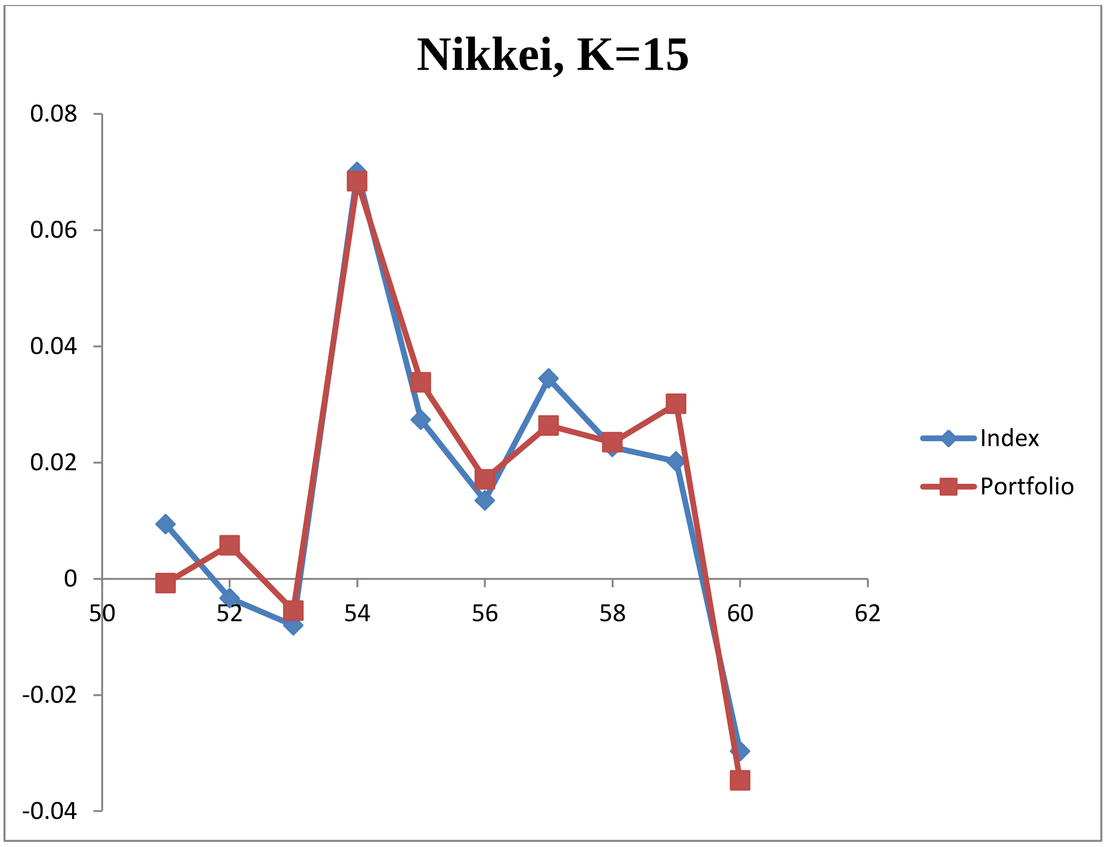


Figure 6.15: Nikkei, K=15

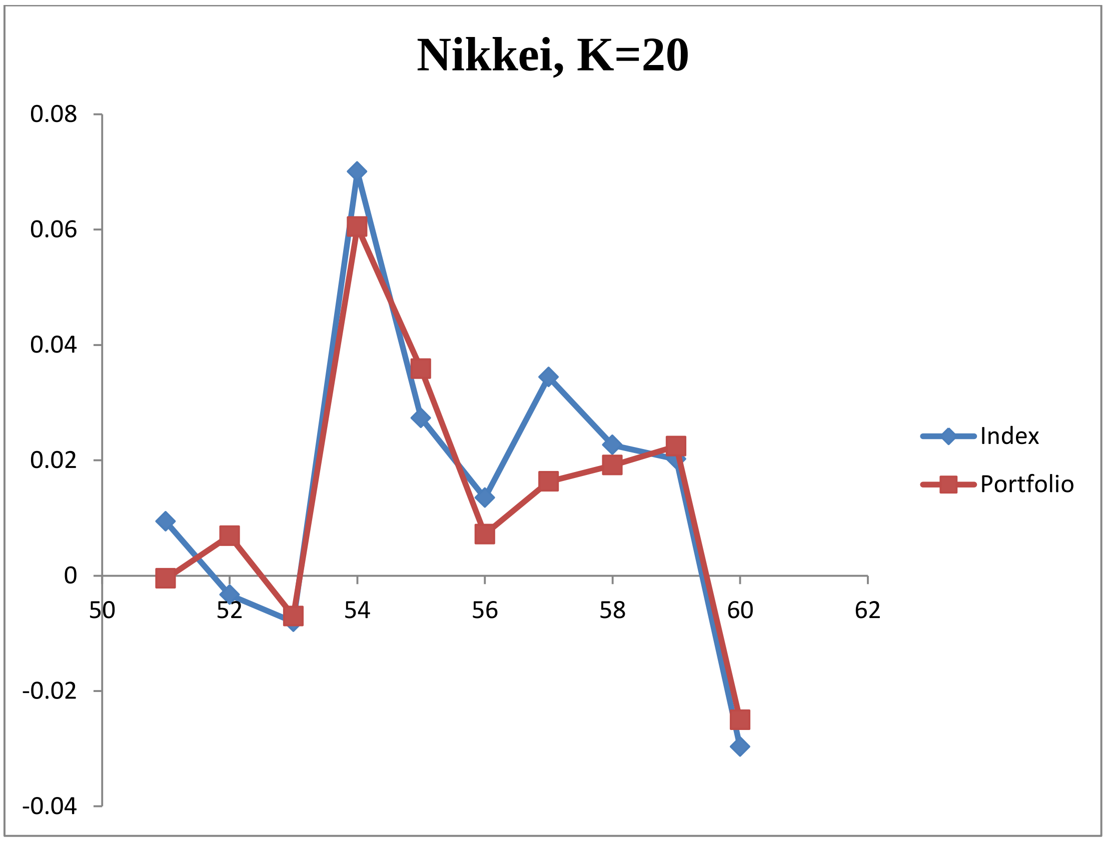


Figure 6.16: Nikkei, K=20

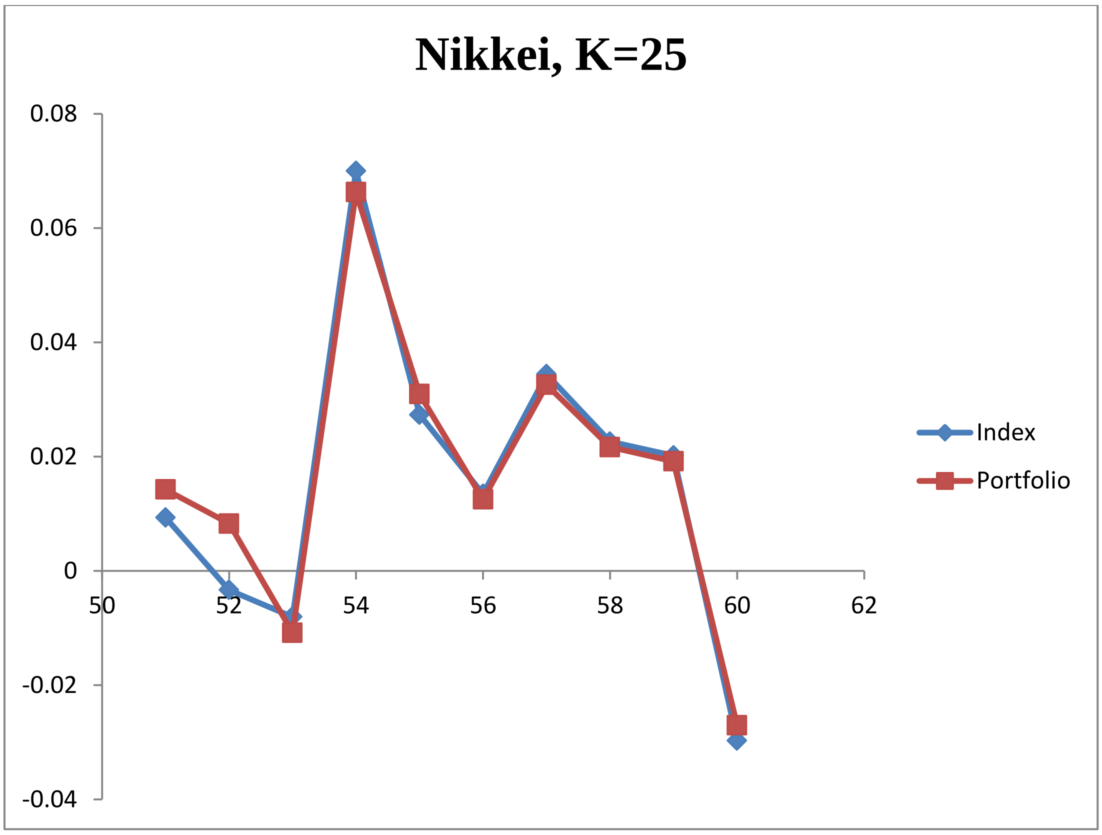


Figure 6.17: Nikkei, K=25

In the above figures, the out-of-sample performance of Hang Seng-10&15, DAX-15, FTSE-20, and Nikkei 225-25 is outstanding, because majority parts of the tracking portfolio curve can match with that of the corresponding index. The rest of the out-of-sample performance is acceptable, but there are four exceptions: FTSE-10&15, S&P-10, and Nikkei 225-10, where large out-of-sample variations occur. By further observation, we notice that the large variations usually happen when the cardinality is small, and when the cardinality is increased the out-of-sample performance would become better, such as the curves of FTSE-20, S&P-20, and Nikkei 225-25 are very close to the curves of their corresponding indices. Overall, we conclude that our model is able to produce acceptable results to track the future performance of an index.

Chapter 7

7 .Conclusion

7.1 Summary

In this thesis, we have developed solution approaches for the Index Tracking problem. We develop two solution approaches to solve the problem, which are Genetic Algorithms, and Ant-colony Optimization. We have presented mathematical formulations for each of the solution approaches, along with the computational results. The computing results of the MIP solution approach are used as a benchmark to measure the performance of the other approaches.

In Chapter 2, we reviewed the previous works on the Index Tracking problem, as well as the portfolio management theory in general. We observe that the genetic algorithms used for the Index Tracking problem often consider only one problem or a limited number of small size problems. In addition, we find that the genetic algorithms still have space to be improved from several aspects, such as the reproduction and mutation operators. We also observe that very few researchers have used the Ant-colony optimization to solve the index tracking problem. We believe it is a good candidate.

In Chapter 3, we consider the Mixed Integer Programming solution approach. We first review BMC’s MIP solution approach. Based on their work, we build our MIP approach and present its formulation. We limit the total computing time to 7200 time units and present the computational results for the eight market indices. We find that, except for the Hang Seng index, the optimal solution cannot be achieved within the given time limitation.

In Chapter 4, we initially present the genetic algorithm from three aspects: initial population generation, reproduction and crossover, and mutation. Then we present the basic IT-GA and the EIT-GA. We use the EIT-GA to solve the eight market indices and compare the computational results with those of the MIP. We conclude that generally the EIT-GA is better than MIP both in terms of solution quality and in computing time. However, the EIT-GA is not able to build reliable relations among the cardinality, the initial population size, and the size of the index. Therefore, we consider the R-GA. We first introduce the Schema Theorem which serves as the foundation for the R-GA. Then we investigate the initial population size and find the most suitable frequency to construct the solid relations among the cardinality, the initial population size, and the size of the index. We also enhance the algorithm and use the enhanced algorithm ER-GA to solve the eight market indices. In the end, we compare the ER-GA with the EIT-GA. We conclude that the ER-GA is preferable, because we can immediately generate a suitable population size for a given problem to help the algorithm find fairly good solutions.

In Chapter 5, we consider the Ant-colony Optimization for the Index Tracking problem. Initially, we give an introduction about the capabilities of our artificial ants. Then we build a basic algorithm by following three steps: construct ant solutions, update pheromone information, and daemon action. We use the basic algorithm to implement investigations on two parameters: colony size and evaporation rate. Regarding the findings from the investigations, we build the ACO algorithm to solve the eight market indices. In the end, we compare the computing results of the ACO with the ER-GA. We conclude that the two solution approaches are equally good. Moreover, we find that the ACO alogrithm works better either with powerful ants and small size colonies or less powerful ants and large size colonies.

In Chapter 6, we implement the post analysis on the computing results. We first investigate the stocks and the best tracking portfolios of the first five indices. We discover several new findings for our research such as the *close stock effect*, the *combination effect*, and the *inheritance effect*. In addition, we state how we can put these findings into practice. Particularly we use the *inheritance effect* to improve the ER-GA to build a more powerful algorithm called Inherited Roulette GA. We find that it not only shortens the computing time largely, but also finds better results for the large size indices. We also apply the *inheritance effect* to the ACO algorithm and conclude that the *inheritance effect* is not suitable for the ACO algorithm. Finally, we test the model by out-of-sample data. The out-of-sample performance shows that the model has strong stability in tracking the index out-of-sample.

7.2 Contribution to Knowledge

Chapter 2 proves that we are familiar with the relevant previous works of both the portfolio management theory and the Index Tracking. The solution approaches presented in Chapters 3-5 are, to the best of our knowledge, original works. The Inverse Triangle GA, the ACO algorithm, and the Inherited Roulette GA/ACO have not been presented elsewhere in the literature. In the following, we list the main contributions of our thesis:

1. The solution approaches developed can deal with a variety of data sets. The smallest data set is the Hang Seng index which contains only 31 stocks while the largest one is the Russell 3000 which contains 2151 stocks.

2. We developed the IT-GA, which finds the best solutions by continuously narrowing down the searching space. It plays a trade-off between the diversity of the algorithm and the computational time. The empirical studies show that it is able to produce sound solutions.

3. For the Roulette GA, we successfully construct solid relations among the cardinality, the population size, and the size of the index. We believe by using the equation: $n \approx \frac{12*N}{K}$ we are able to immediately generate a suitable size of initial population for a given problem to help the algorithm produce fairly good solution result. We also proved that the 'semi-optimization' operator used for breeding is better than the crossover operator. Additionally, instead of using a fixed 'maximum generation' as the stopping criteria, we

created a more flexible one, which is used to check the changing the best solution of an algorithm.

4. We are one of the pioneers to use the ACO algorithm to target the index tracking problem. Through the empirical experiments, we find the most suitable evaporation rate (0.1) for the algorithm to deliver high quality solution in good computational time. We also find that the ACO works equally well either with powerful ants and small size colonies or less powerful ants and large size colonies.

5. In the post analysis, we discover the close stock effect, the combination effect, and the inheritance effect. Particularly, we apply the inheritance effect to the ER-GA and make the algorithm even more powerful in searching for good solutions.

We would like to point that, the solution approaches are tested by different sizes of indices, which has previously been done by very few people in the field. Therefore, the solution approaches are reliable, and we encourage people to use them in real-case practice. Also, we find out that Nigam and Agarwal (Ashutosh Nigam and Yogesh K. Agarwal, 2013) used the ACO to solve the index fund problem after we have completed the ACO solution approach. We declare that the two works are independent from each other.

7.3 Future Ideas and Directions

In this thesis we have presented and evaluated a number of solution approaches for the Index Tracking. However, there are a number of extensions and enhancements that could be considered:

We can use the close stock effect to select stocks to generate portfolios as we know that most of the stocks in the optimal or good tracking portfolios are selected from the top 80% of the close stocks. Usually, the top 20% close stocks are the most important ones, as they often constitute the largest portion in the portfolios.

Therefore, when we generate portfolios, we enable the top close stocks to have more chances of being selected.

The inheritance effect shows that there are inner connections between the good tracking portfolios with different cardinalities. One could use a ‘build-up’ approach, where the cardinality is slightly increased from 1 to the targeting number K, to solve the index tracking problem. For example, once the best close stock is found, one could use it to find a good tracking portfolio which includes two stocks, and then use these two stocks to find a good tracking portfolio with three stocks, etc., until it reaches the targeting cardinality K. Alternatively, one could take a hybrid solution approach, where it combines the MIP with GAs or ACO. The MIP is used to find portfolios with small cardinalities, such as k is equal to 3 or 5, and then use the finding stocks to search for good portfolios with larger cardinalities by using the heuristic methods.

Reference

Alexander, C., 2001. *Market models: A guide to financial data analysis.* s.l.:John Wiley & Sons.

Ashutosh Nigam and Yogesh K. Agarwal, 2013. Ant colony optimization for index fund problem. *Journal of Applied Operational Research,* 5(3), pp. 96-104.

B. Miller and D. Goldberg, 1995. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, pp. 193-212.

Beasley, J.E., Meade, N. and Chang. T.J., 2003. An evolutionary heuristic for the index tacking problem. *European Journal of Operational Research 148,* pp. 621-643.

Beasley, J. E., 2013. Portfolio optimisation: models and solution approaches. *In Tutorials in Operations Research ,* Volume 10, pp. 201-221.

Canakgoz, N. and Beasley, J., 2009. Mixed-integer programming approaches for index tracking and enhanced indexation. *European Journal of Operational Research,* Volume 196, pp. 384-399.

Carhart, M., 1997. On Persistence in Mutual Fund Performance. *Journal of Finance,* 52(1), pp. 57-82.

Chang-Chun Lin, Yi-Ting Liu., 2008. Genetic algorithms for portfolio selection problems with minimum transaction lots. *European Journal of Operational Research*, Volume 185, pp. 393-404.

Charles Thomas, Peter Westaway and Todd Schlanger, n.d. *Vanguard UK Adviser.* [Online] Available at: https://www.vanguard.co.uk/adviser/adv/home
[Accessed 4 11 2014].

Chen, C. and Kwon, H. R., 2012. Robust portfolio selection for index tracking. *Computers & Operations Research,* Volume 39, pp. 829-837.

Coleman, T.F., Henninger, J., and Li, Y., 2006. Minimizing tracking error while restricting the number of assets. *Journal of Risk*, Volume 8, pp. 33-56.

David Colwell, Nadima El-Hassan, and Oh Kang Kwon, 2007. Hedging diffusion processes by local risk minimization with applications to index tracking. *Journal of Economic Dynamics & Control*, Volume 31, pp. 2135-2151.

Deneubourg, J.-L., Aron, S., Goss, S., &Pasteels, J.-M., 1990. The self-organizing exploratory pattern of the Argentine ant. *Journal of Insect Behaviour* , Volume 3, pp. 159-168.

Dorigo M, Stützle T, 2004. *Ant Colony optimization.* Cambridge: MA: MIT Press.

Elton, E., 2007. *Modern portfolio theory and investment analysis.*. New York: Wiley.

Fama, E. F., & French, K. R., 1993. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, Volume 33(1), pp. 3-56.

Fama, E. F., & French, K. R., 1996. Multifactor explanations of asset pricing anomalies. *The Journal of Finance,* Volume 55(1), pp. 55-84.

Fama, E. F., and French, K. R., 1992. The Cross-Section of Expected Stock Returns. *Journal of Finance,* 47(2), pp. 427-466.

Fang, Y. and Wang, S.Y., 2005. A fuzzy index tracking portfolio selection model. *Lecture Notes in Computer Science,* Volume 3516, pp. 554-561.

Focardi, S.M. and Fabozzi, F.J., 2004. A methodology for the index tracking based on time-series clustering. *Quantitative Finance,* Volume 4, pp. 417-425.

Francesco Corielli and Massimiliano Marcellino, 2006. Factor based index tracking. *Journal of Banking & Finance,* Volume 30, pp. 2215-2233.

Goldberg, D. E., 1989. *Genetic Algorithm in Search, Optimization, and Machine Learning.* s.l.:Addison Wesley Longman.

Goss et al, 1989. Self-organised shortcuts in the Argentine ant. *Naturwissenschaften,* Volume 76, pp. 579-581.

Guastaroba, G., & Speranza, M. G., 2012. Kernel search: An application to the index tracking problem. *European Journal of Operational Research,* 217(1), pp. 54-68.

Haugen, R.A. and Baker, N.L., 1990. Dedicated stock portfolios. *Journal of Portfolio Management,* 16(4), pp. 17-22.

Holland, J., 1975. *Adaptation in Natural and Artificial Systems: An Introduction Analysis with Application in Biology, Control, and Artificial Intelligence.* s.l.:MI.

Jeurissen, R. and Van Den Berg, J., 2005. *Index Tracking Using a Hybrid Genetic Algorithm.* s.l., Computational Intelligence Methods and Applications Conference.

Kyong Joo Oh, Tae Yoon Kim, and Sungky Min, 2005. Using genetic algorithm to support portfolio optimization for index fund management. *Expert Systems with Applications,* Volume 28, pp. 371-379.

Lai, K. K., Yu, L., Wang, S. Y. and Zhou, C. X., 2006. A Double-Stage Genetic Optimization Algorithm for Portfolio Selection. *Springer-Verlag* , pp. 928-937.

Larsen Jr., G.A. and Resnick, B.G., 1998. Empirical insights on indexing. *Journal of Portfolio Management,* 25(1), pp. 51-60.

Lee, J.Y., Kim, T.Y. and Min, S., 2005. *Portfolio Optimization for Index Fund Management based on Genetic Algorithm: A Stratified Approach.* s.l., 대한산업공학회 2005 추계학술대회 논문집.

Lin, C.M. and Gen, M., 2007. An Effective Decision-Based Genetic Algorithm Approach to Multiobjective Portfolio Optimization Problem. *Applied Mathematical Sciences,* Volume 5, pp. 201-210.

Lintner, 1965. Security Prices, Risk, and Maximal Gains from Diversification. *Journal of Finance,* 20(4), pp. 587-615.

Maringer, D., 2008. Constrained index tracking under loss aversion using differential evolution. *Studies in Computational Intelligence,* Volume 100, pp. 7-24.

Markowitz, H., 1952. Portfolio Selection. *Journal of Finance 7 (1),* pp. 77-91.

Markowitz, H., 1959. *Portfolio selection: Efficient Diversification of Investments.* New York: Wiley.

Markowitz, H., 1991. *Portfolio Selection: Efficient Diversification of Investments.* New York: Wiley.

Mitchell, M., 1998. *An Introduction to Genetic Algorithms.* 6 ed. London: Massachusetts Institute of Technology.

Mossin, 1966. Equilibrium in a Capital Asset Market. *Econometrica,* 34(4), pp. 768-783.

Okay, N. and Akman, U., 2003. Index tracking with constraint aggregation. *Applied Economics letters*, Volume 10, pp. 913-916.

Orito, Y., Tamamoto, H. and Yamazaki, G., 2003. Index Fund Selections with Genetic Algorithms and Heuristic Classifications. *Computers & Industrial Engineering*, Volume 45, pp. 97-109.

Pandari, A.R., Azar, A., and Shavazi, A.R., 2012. Genetic algorithms for portfolio selection problem with non-linear objectives. *African Journal of Business Management*, Volume 6, pp. 6209-6216.

Papadimitriou and Steiglitz, 1998. *Combinatorial optimization: algorithms and complexity*. New York: Dover.

Roll, R., 1977. A Critique of the Asset Pricing Theory's Tests Part 1: On Past and Potential Testablility of the Theory. *Journal of Financial Eonomics*, 4(2), pp. 129-176.

Ross, S., 1976. The Arbitrage Theory of Capital Asset Pricing. *Journal of Ecnomic Theory* , 13(3), pp. 341-360.

Rudd, A., 1980. optimal selection of passive portfolios. *Financial Management*, pp. 57-66.

Ruiz-Torrubiano, R., & Suarez, A., 2009. A hybrid optimization approach to index tracking. *Annals of Operations Research*, 166(1), pp. 57-71.

Shapcott, J., 1992. *Index Tracking: Genetic Algorithms for Investments Portfolio Selection*, s.l.: EPCC-SS92-24.

Sharpe, W., 1964. Capital asset prices: A theory of market equilibrium under conditions of risk. *Journal of Finance 19 (3)*, pp. 425-442.

Sharpe, W. F., 1966. Mutual fund performance.. *The Journal of Business*, Volume 39(1), pp. 119-138.

Sharpe, W. F., 1975. Adjusting for risk in portfolio performance measurement. *Journal of Portfolio Management*, Volume 1(2), pp. 29-34.

Sharpe, W. F., 1994. The Sharpe ratio. *Journal of Portfolio Management*, Volume 21(1), pp. 49-58.

Soam, Palafox, and Iba, 2012. *Multi-Objective Portfolio Optimization and Rebalancing Using Genetic Algorithms with Local Search*. Japan, The University of Tokyo.

Treynor, J., 1961. Towards a theory of market value of risky assets. *Unpublished Manuscript.*

Tun-Jen Chang, Sang-Chin Yang, Kuang-Jung Chang, 2009. Portfolio optimization problems in different risk measures using genetic algorithm. *Expert Systems with Applications*, Volume 36.

Wang, M. H., 2012. A mixed 0–1 LP for index tracking problem with CVaR risk constraints. *Springer Science+Business Media,* Volume 196, pp. 591-609.

Wilmott, P., 1998. *Derivatives: The theory and practice of financial engineering.* Chichester: John Wiley and Sons Ltd.

Wright, S., 1984. *Evolution and the Genetics of Populations: Genetics and Biometric Foundations v. 1 (Genetic & Biometric Foundations).* s.l.:New Edition. University of Chicago Press.

Yao, D.D, Zhang, S, and Zhou, X.Y., 2006. Tracking a financial benchmark using a few assets. *Operations Research,* Volume 54, pp. 232-246.

Yu, L., Zhang, S., and Zhou, X.Y., 2006. A downside risk analysis based on financial index tracking models. *Stochastic Finance,* pp. 213-236.