Portfolio Management with Cost Model using Multi Objective Genetic Algorithms

A Thesis Submitted to
The Department of Frontier Informatics
by

Claus Aranha
56343

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

Supervisor: Hitoshi Iba

Graduate School of Frontier Sciences
The University of Tokyo
August 2007

Abstract

In this work, we study in depth the problem of Portfolio Optimization, and the application of Genetic Algorithms to solve it. We discuss the limitation of current approaches, that do not take into consideration multiple scenarios, nor transaction costs, and propose a modification of the Genetic Algorithm System for Portfolio Optimization to address these issues.

A Financial Portfolio is a strategy of making investiments in a number of different applications, instead of focusing in a single security. The idea behind the creation of a Financial Portfolio is that by investing in many different assets, the investor can reduce the specific risk from each asset, while maintaining a target return rate. Markowitz, in 1958, proposed a formal model for this strategy, which is called the Modern Portfolio Theory (MPT).

Nowadays, there is a number of optimization techniques that propose to solve the MPT's equations in order to find out the risk/return optimal portfolio for any given market. However, while Markowitz model has a very elegant mathematical construction, when we remove the restrictions that do not hold in the real world, it becomes a very hard optimizaton problem.

A number of numerical techniques have been developed to solve this problem. Among then, we highlight the Genetic Algorithms (GA), which are random search heuristics which have showed themselves very appropriate for such large problems. Current GA-based approaches, however, still tackle a limited version of the problem, without complications like transaction costs.

In this work we extend the current GA-algorithms for Portfolio Optimzation, aiming at a system that is able to overcome some of the incomplete points in its predecessors. More specifically, we wish to address the modeling of a transaction cost measure, and the influence of market changes over time.

In this work we develop two new techniques: Objective Sharing and Seeding, to address the above questions. We perform experiments of these new techniques using historical data from the NIKKEI and the NASDAQ market indexes.

From the results of our experiments we can see that it is possible to get consistently high returns while reducing the distance of the optimal portfolios between scenarios. It becomes apparent that our approach is a fruitful one, and can be used to build models that are closer to the real world than other current techniques.

Acknowledgements

I'd like to thank Professor Hitoshi Iba and the Japanese Ministry of Education for allowing me the opportunity of conducting my studies in the University of Tokyo. This was a unique period of my academic life where I enjoyed many chances to enlarge my horizons.

Also, during these 2 and a half year, I always had the support from all my fellow colleagues at IBA laboratory. In particular, I would like to thank Nasimul Noman, Makoto Iwashita and Yoshihiko Hasegawa for the many many suggestions they gave me regarding my work. Be assured that I took every last one of them into consideration. I would also like to thank Daichi Ando and Shotaro Kamio for all the practical guidance during my first months in Japan. Finally, I would like to extend very special thanks to Toshihiko Yanase, who gave me lots of support, both technical and moral, and made me feel less of a foreigner in the laboratory.

Outside the academia, I would like to thank my many friends which I met in Japan, and made this period much more interesting. Among them I would like to mention the Fabulous Four of Kashiwa Campus (Chaminda, Pamela and Ahmet), Mrs Saito and her friends in the graduate affairs division, and all the many Brazilian exchange student that I met through ASEBEX.

Last but not least I would like to thank all the support my parents gave me, and all the patience and love from Marilia, who was always quick to remind me that reading Slashdot when I was supposed to be writing my dissertation is not procrastination, but relaxation :-).

Kashiwanoha 5-1-5, Kashiwa-shi, Chiba 277-8561 Claus Aranha

August 1st, 2007

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the popularization of the Internet and personal computers in the past decade and a half, the trading of financial instruments has left the realm of the financial institutions, and became much more accessible. Many people nowadays trade stocks from different markets as their secondary (sometimes even primary) means of income. Simulations of the market to the public are also common, and allow people without much knowledge of financial theory to get an intuitive grasp of the workings of the market place.

In this environment, the interest on financial research, and its development outside the academic circles of economists and mathematicians, has also flourished. The improved computer power allows the calculation of a greater amount of data, faster. This leads to the development of computational tools for the aid of the financial trader. For instance, more accurate methods for prediction of future trends (Nikolaev and Iba, 2001), the simulation of market behavior (Subramanian *et al.*, 2006), and more recently, the automation of simpler tasks in the financial market (Jiang and Szeto (2003), Aranha *et al.* (2007)).

In this work, we address the problem of Financial Portfolio Optimization from a Genetic Algorithm perspective. This is a problem that has gathered a lot of attention in recent years, yet poses many open questions. In this dissertation we'll overview the Portfolio Optimization problem, the most relevant work in regards to it, and then address some of those open problems.

1.1 Financial Portfolios

The word *Portfolio* usually means "collection". Thus we can have the portfolio of a company, meaning the different products that this company produces, or the portfolio of an artist, which is the collection of his significant works.

In the financial sense, a portfolio is a collection of different investments. Since any market activity involves a certain amount of risk, it is often a good idea not to put all of your resources into a single security. Also, different securities (like bonds, stocks, options, foreign currency, etc) have different characteristics regarding liquidity, dividends, and valorization rate. Because of this, an investment strategy must take into account the characteristics of each of the many securities, and divide the available resources accordingly.

The idea behind a portfolio spread is that different securities may show different behaviors over time, even when their perceived risk and return are the same. For instance, if two securities belong to two competing industries, some external factor that affects negatively one of them, may affect positively the other one. If we had concentrated all of our capital in one of the two securities, there would be 50% chance that we picked the "right" security, and a 50% chance that we picked the wrong one. On the other hand, by sharing the capital between the two securities would mean that the losses in one of them would be offset by the gains in the other.

For the day trader or other short term investors, it might be more rewarding to take the risk of concentrating their capital in a few lucrative investments. These investors work with many transactions of small value, hoping to profit from local fluctuations of individual securities. However, for longer-term investors and funds, it is better to reduce risk as much as possible (while maintaining desired return rates) so that large investments may be used to reap long term rising trends in the market. Also, having a well diversified portfolio prevents the risk of unexpected events in certain markets that may lead to crashes.

1.2 Portfolio Optimization and Management

Markowitz stated the Modern Portfolio Theory (Markowitz, 1987), which says that the risk of an investment can be diminished by an appropriate diversification of the assets that compose it. This diversification reduces the *specific risk*, leaving only the *systematic risk*.

Specific Risk is the risk associated to a single asset. It happens due to factors particular to that asset. For instance, a sudden bankruptcy of one company may plummet the values of the stocks associated to that company - the chance of this happening would be considered as the specific risk of those assets. Portfolio Optimization is capable of reducing Specific Risk.

Systematic Risk, on the other hand, is the risk associated with the whole market. Sudden crashes or bubbles may rise or lower dramatically the values of all assets in a market, beyond the predicted values. No amount of diversification is capable of preventing Systematic Risk.

The first challenge to applying the Modern Portfolio Theory is to design a proper model in which to calculate the optimum portfolio. This model must be precise enough to reflect the behavior of the market in the real world. However, it cannot be so complex that it is not possible to analyze it. Over the years, many different models have been proposed in an attempt to strike a balance between these two factors.

After the model has been devised, it is necessary to develop an optimization routine to this model, and decide how to apply the results into the real world. The optimization method will return a weighting for every available asset, so that the investor knows how to distribute his capital in order to maximize the expected return, and minimize the risk of his portfolio.

In short, the problem of Portfolio Optimization can be thought of as a Resource Management problem. The capital is the limited resource that must be distributed between the available assets, in order to maximize the objective function.

Portfolio Management is an extension of the Portfolio Optimization problem. It is defined as the task of achieving a desirable risk/return ratio on a dynamic

market in a continuous fashion, as opposed to the "one scenario" nature of Portfolio Optimization.

There are two added difficulties in the Portfolio Management problem. The first one is that, as the market is dynamic, its proprieties change with time. The risk and return values and characteristics for each asset are not constant. So the method used to optimize the portfolio must be able to adapt itself to the changing characteristics of the market. We find that Genetic Algorithms are quite appropriate for this aspect of the task.

The second difficulty is the issue of cost. If the optimal portfolio at one scenario is too different from the optimal portfolios at the scenarios immediately before and after it, changing back and forth between optimal portfolios might be more costly than picking a solution which is sub-optimal in some, but positive to all scenarios.

1.3 Traditional Optimization Methods

The model described by Markowitz's Modern Portfolio Theory has already been solved in theory. Following that model, it can be proved that the optimal portfolio is the Market Portfolio, which contains all assets available in the market, weighted by capitalization.

However, this basic model has too many assumptions which do not hold in practice. As we remove those assumptions, and add restrictions like cost, trading limits, imperfect information, etc., the model becomes more and more complex, and finding an optimal portfolio becomes harder.

More elaborate models of the problem have been solved by numerical optimization methods, like quadratic programming or linear programming (ping Chen *et al.*, 2002). However, the biggest problem with numerical methods is that their algorithms do not scale very well, and quickly become complex with the number of assets.

For instance, optimization techniques for versions of the MPT require the correlation values between all the assets for each iteration. This is computationally very intensive, and grows exponentially on the size of the data.

In fact, reports on simulations for most Portfolio Optimization techniques limit themselves to 10 or 20 assets at a time. The use of search heuristics is a possible solution to this problem.

1.4 Genetic Algorithms

Genetic Algorithm is a Random Search heuristic which is inspired by biological evolution. The basic setup of a Genetic Algorithm is as follows:

Each possible solution to a problem is considered an individual. A population of random individuals is generated, and each solution is tested for its goodness. The best solutions in the population are mixed together, generating new individuals. The worst solutions are removed from the population.

The heuristic behind Genetic Algorithms is that by mixing the best solutions, we can direct the population to explore better regions of the search space. Genetic Algorithms is a mature technique that has been researched for more than 20 years, with practical applications in many fields.

One of the good characteristics of Genetic Algorithms is that they are able to adapt to changes to the problem they are trying to solve. If the environment changes (for example, if the data values change), the algorithm is able to find a new optimal solution without any changes to its population or parameters, by executing a few more iterations.

This makes Genetic Algorithm specially fit for problems where they must continually optimize a solution in an evolving environment. Like the Portfolio Management problem we have described earlier.

1.5 Proposal

In this work, we study the Portfolio Optimization and the Portfolio Management problem, and the research field of using the GA heuristic to solve those problems.

There were many works in recent years that aim to use the powers of Genetic Algorithm to solve the Portfolio Optimization problem. However, each of these

works seem to approach the problem from a different angle, using GA in different ways, and different models for the Portfolio and the market.

In this dissertation survey the many methods that were developed, and analyze the techniques proposed, and the models assembled. From the results of this analysis, we understand that the current methods for Portfolio Optimization lack a proper approach to transaction costs, and to Portfolio Management.

We propose the use of the adaptive characteristic of GA to improve a simple Portfolio Optimizer into a system able to manage a portfolio through multiple scenarios. Our improvements consist mainly of two new ideas:

1. The insertion of successful individuals from previous scenarios into the current generation. And;

2. The use of a secondary goodness metric: The distance between two portfolios, which we use to measure transaction costs.

In this work we explain with details both approaches, and we perform experiment to understand their effectiveness.

1.6 Outline of the Thesis

Chapter 2 explains the basic financial concepts necessary for the development of this work. In this chapter we discuss our choice for the portfolio model used, and define the calculations of risk, return, and transaction cost. We offer some information on other possible models, and some of the traditional methods on how to approach the portfolio optimization problem. The goal of this chapter is to introduce all the mathematics necessary for this work to the reader without a financial background.

Chapter 3 makes a brief introduction to genetic algorithms, and then explains how they can be used to solve optimization problems. We'll also discuss the concept of Multi-Objective evolutionary algorithms, which can be used to solve problems where there are more than one competing objectives. The goal of this

chapter is to explain the concept of Genetic Algorithms for Optimization Problems for the readers without this familiarity.

Chapter 4 exposes previous works from other researches that have a similar approach as ours. These works have some influence in our current method, and we discuss the more important ones in detail, pointing out their strengths, and where they could be improved.

Chapter 5 explains our approach for portfolio management by use of GA. Here we explain again in details our model, and how it is implemented in the system. We show what are the parameters of the system, and describe how were values decided for these parameters. We describe the difference between using Genetic Algorithms for Optimizing a portfolio, and for Managing a portfolio, and argue on the superiority of the later method.

Chapter 6 shows the experiments that were performed to validate our proposal. We have executed a simulation with real world historical data from Japanese and American markets, and observed good results with the proposed method. The methodology for the experiments is explained, and the results are shown and discussed.

Chapter 7 proposes a reflection on the field of GA-based portfolio optimization based on the results of our work. We explain how the information acquired here can be useful to the field, both as basis for applications, and as insight for future experiments.

Chapter 2

Financial Background

Financial Engineering is the application of economical theories to real-world investment problems. Economic models have been developed describing the behavior of many different kinds of securities under varied conditions. These models propose elegant solutions that generate optimal strategies for investment in many problems.

For instance, Markowitz's Modern Portfolio Theory (Markowitz (1952) and Markowitz (1987)) (MPT) describes the behavior of investors when many assets with different risks and returns are available. The main result is that the optimal strategy is to invest in the *market portfolio*, which is composed of all available assets, weighted by capitalization. In order to achieve higher returns or lower risks, the market portfolio may be combined with a theoretical riskless asset.

From the MPT, the Capital Asset Pricing Model (CAPM) can be derived. The CAPM is a model that allows the evaluation of assets and portfolios with relation to the market portfolio.

Also, the MPT and the CAPM rely on the knowledge of market rates of return and risk for all the assets. That kind of information, however, is not so readily available. While "risk as deviation" is a well known measure of risk, there are other models for modelling risk which can also be used. The choice of model depends on which characteristics of risk the investor finds more important.

Return as well, can be estimated in many different ways. The straightforward moving averages was used for the elaboration of the MPT. But we could use more fancy moving averages, like the weighted moving average or the geometric moving average, or elaborated schemes based on evolutionary computing in order to obtain

a more precise figure on the estimated return.

In this chapter we will introduce the basics of Markowitz's MPT and CAPM, and then discuss some risk and return models.

Both the MPT and the CAPM make strong assumptions about the model. Among those are: the investors have perfect information, they react perfectly to market changes and risk and return are the only relevant characteristics of an asset. Also trading costs, upper and lower bounds to trade volume are not included in the model. By releasing these assumptions, the solutions proposed in their models become much harder computationally.

2.1 Markowitz Modern Portfolio Theory

A financial asset can be represented in terms of its *return*, and its *risk*.

The return of an asset is the relative change of its value over time - it indicates whether the price of the asset has gone up (positive return), or down (negative return). Financial applications are usually based on the idea of the *Estimated return* of an asset. If the estimated return for a certain point in the future is high, it is profitable to buy the asset now, to sell it at a higher value later.

The risk of an asset is the chance that the actual value of the return in the future will be different than the expected return. In the Modern Portfolio Theory, we use the variance of the rate of return as the measure of risk for one asset.

We model the market by defining n tradeable assets. The rate of return for each asset is described as r_n, and the expected return rate is $E[r_n]$. The error for each asset is defined as σ_n.

A portfolio W is a weighted composition with n weights $(w_1, w_2, ...w_n)$. Each weight w_i represents how much of the available capital will be allocated to asset i. There are usually two restrictions for the values of w_i:

1. $\sum_{i=1}^{n} w_i = 1$
2. $w_i \geq 0$

The second restriction means that short-selling is not allowed. Short-selling means selling assets that you have borrowed. It can be shown that the Modern Portfolio Models with and without short-selling are equivalent (Yuh-Dauh-Lyu, 2002), so we'll add this restriction for the sake of simplicity.

In this model, the return of the portfolio W is given as the weighted sum of the assets composing it:

$$r_W = \sum_{i=1}^{n} w_i r_i \tag{2.1.1}$$

And the variance (risk) of the portfolio is calculated by the correlation of the risks of the composing assets:

$$\sigma_W = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \sigma_{ij} \tag{2.1.2}$$

Where $\sigma_{ii} = \sigma_i^2$ is the variance of r_i, and σ_{ij} is the covariance between r_i and r_j.

The calculation for the error of the portfolio in equation 2.1.2 can then be divided in two parts, one containing the variance of the returns, and another containing the covariance of the returns:

$$\sigma_W = \sum_{i}^{n} w_i^2 \sigma_i^2 + \sum_{i \neq j} w_i w_j \sigma_{ij} \tag{2.1.3}$$

The first part of equation 2.1.3 describes the portfolio risk component composed by the risk of individual assets. It is called the *specific risk*. If there is no correlation between the assets in the portfolio (the second part of the equation is 0), we can show that the specific risk can be reduced by *diversification* (adding more assets to the portfolio):

Proof: Diversification reduces specific risk. For a portfolio with n assets, assume that $w_i = 1/n$, and that $\sigma_{ij} = 0$ for every i and j. The risk of the portfolio is then:

$$\sigma_W = \frac{\sum \sigma_i^2}{n^2}$$

While the mean of the risks is:

$$\bar{\sigma} = \frac{\sum \sigma_i}{n}$$

Make the risk of all assets equal to the highest risk, $/sigma_{max}$. Then as n increases (as we add assets to the portfolio), the portfolio's risk will decrease towards zero, as the mean of the risks remains the same:

$$\begin{aligned} \frac{\sum \sigma_{max}^2}{n^2} &\leq \frac{\sum \sigma_{max}}{n} \\ \frac{n\sigma_{max}^2}{n^2} &\leq \frac{n\sigma_{max}}{n} \\ \frac{\sigma_{max}^2}{n} &\leq \sigma_{max} \end{aligned}$$

For increasing values of n. □

The second part of equation 2.1.3 describes the portfolio risk component composed by the risk shared by the assets. It is called *systematic risk*. If the variance is different then zero, then the systematic risk cannot be reduced below the average by adding more assets:

Proof: Systematic risk cannot be reduced by Diversification. For a portfolio with n assets, assume that $w_i = 1/n$, $\sigma_{ii} = s$ and $sigma_{ij} = a$ for every i and j. Then the risk of the portfolio becomes:

$$\begin{aligned} \sigma_W &= \frac{\sum_i^n s^2}{n^2} + \frac{\sum_{i \neq j} a}{n^2} \\ &= \frac{ns^2}{n^2} + n(n-1)\frac{a}{n^2} \\ &= \frac{s^2}{n} + a - \frac{a}{n} \\ &= a + \frac{s^2 - a}{n} \end{aligned}$$

So that even if we increase n (the number of assets), the risk won't become less than the average covariance, a. □

2.1.1 Efficient Portfolios

The MPT makes two assumptions about the behavior of investors which can be used to decide the efficiency (optimality) of a portfolio.

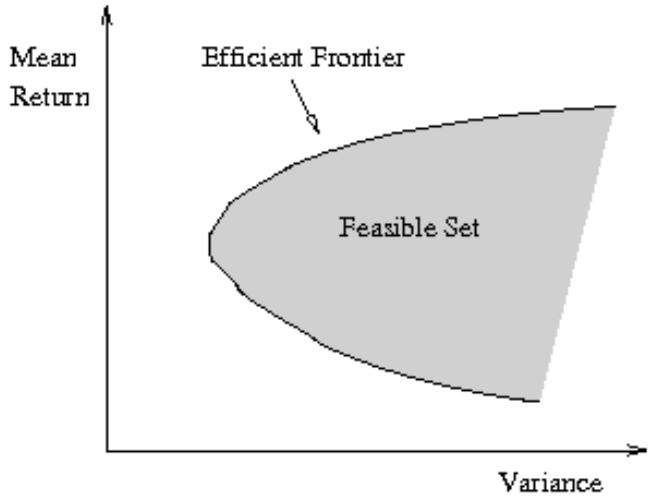

Figure 2.1: The feasible and efficient sets of portfolios, according to the MPT's mean-variance model.

The first one is that they only care about risk and return. This means that, even if the composing assets are different, two portfolios with the same measures of risk and return are equal in the MPT.

The second one is that the investors are risk-averse. This means that given two portfolios with the same mean return, the one with smaller risk/variance will be considered more efficient.

Based on these two assumptions, we can define an efficient (optimal) portfolio, according to the MPT model. Consider a two dimensional mapping where the horizontal axis denotes the risk associated with a given portfolio, and the vertical axis denotes the mean of its return (see figure 2.1).

Each point inside the shaded area indicates one of the possible portfolio configurations. We can assume that this region, which we will call the *feasible set* is solid and convex to the left (Yuh-Dauh-Lyu, 2002). If we pick an arbitrary point, moving up in the Y axis from that point means that we find another portfolio with a higher return for the same risk rate. Conversely, if we move left from a given point, we'll find a portfolio which has a lower risk for the same mean return.

In other words, portfolios that are located more to the left, or to the top of

the diagram are considered to be better, in that they provide a higher return for a lower risk. We say that a given risk/return combination represented in this way is *efficient*, if there is no other portfolio that offer a higher return for the same amount of risk, or less risk for the same return.

We call the set of efficient combinations of risk/return in a scenario the *efficient frontier*. In figure 2.1, this combination is the upper left border of the feasible set. Portfolios whose representation falls on the efficient frontier are said to be *efficient portfolios*.

2.1.2 Riskless Asset

Normally, it is not possible to design a portfolio with a better return/risk ratio than that in the efficient frontier. However, when we add a riskless asset to the model, this restriction may be lifted.

A riskless asset, as the name implies, is a security which's deviation of the mean return is 0. While perfectly riskless assets are a theoretical construct, in practice we can get similar results with very low risk assets, like government bonds.

The presence of the riskless assets allows for the investor to combine it with a portfolio at the efficient frontier. According to Markowitz's 2-fund theorem, the combination of two portfolios is *linear*: This mean that if we have a portfolio W_c composed of two portfolios ($W_c = (1-\gamma)W_1 + \gamma W_2$), the possible return and risk representations of W_c are $r_c = (1-\gamma)r_1 + \gamma r_2$ and $\sigma_c = (1-\gamma)\sigma_1 + \gamma\sigma_2$.

Figure 2.2 describes this relationship more clearly. As can be seen, by combining the riskless asset and a portfolio in the efficient frontier, it is possible to hold portfolios outside the feasible set.

In particular, if we find the line that is tangent between the riskless asset and the feasible set, we have a new efficient frontier. If holding short positions (borrowing) the riskless asset is allowed, this line may extend beyond the tangent point, allowing for higher returns with similar risk amounts when compared with the feasible set.

In this way, finding the efficient frontier becomes the problem of finding the tangent portfolio between the riskless asset and the feasible set.

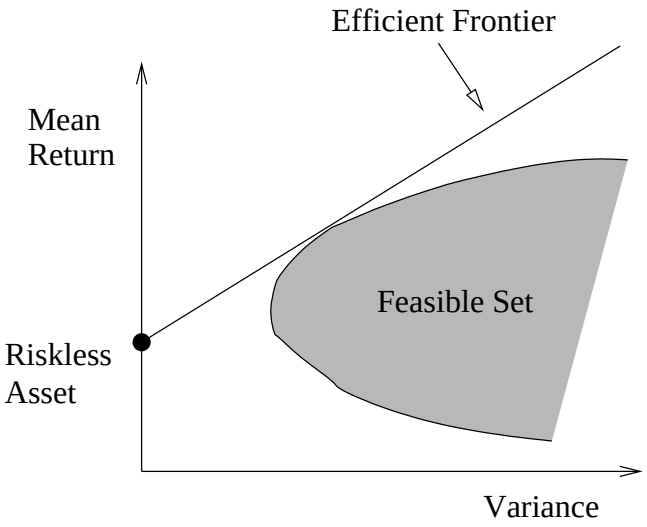

Figure 2.2: Possible combinations between a riskless asset and an efficient portfolio

2.1.3 Capital Asset Pricing Model

If we imagine that all the investors in a market are rational, and possess the same information about the risk and return values from the assets, we can draw some conclusions about the nature of the efficient portfolio.

From the previous subsection we know that the efficient portfolio is the one portfolio located in the tangent between the efficient frontier and the feasible set. This line is composed by the points representing different balances between the efficient portfolio and the riskless asset.

The slope of the line between the riskless asset and a portfolio is given by:

$$Sr = \frac{r_p - r_f}{\sigma_p} \tag{2.1.4}$$

Where r_p and σ_p are the portfolio's mean return and risk, respectively, and r_f is the riskless asset rate of return. This values tells us the rate at which the risk will rise when we increase the return. In other words, it can be thought of as the price of risk.

This rate is also known as the *Sharpe Ratio*. It can be used as an analysis tool to determine how good is a portfolio, since it measures the amount of risk and

return that can be traded by balancing the portfolio with the riskless asset (Lin and Gen (2007), Subramanian *et al.* (2006), Yan and Clack (2006)).

By definition, the positions located in the efficient frontier hold the best risk/return ratios. Consequently, we can assume that all the investors in the market will want to hold only some amount of the riskless asset and the efficient portfolio. Since all assets in the market must be held by somebody, a consequence of the MPT is that the efficient portfolio is composed by all the assets available in the market, weighted by their capitalization. We'll call this portfolio the *market portfolio*.

Using the this model, we can also identify the contribution of each individual asset i to the risk of the market portfolio.

$$\beta_i = \frac{\sigma_{i,M}}{\sigma_M^2} \tag{2.1.5}$$

Where σ_M^2 is the risk of the market portfolio, and $\sigma_{i,M}$ is the covariance between asset i and the market portfolio. This beta can be extended to symbolize the risk relationship between any portfolio and the market portfolio:

$$\beta_p = \sum_i w_i \beta_i \tag{2.1.6}$$

Beta is another CAPM indicator that can be used to measure the risk/return efficiency of a portfolio (Lipinski *et al.*, 2007).

2.2 Financial Engineering

In order to extract effective predictions from models like the MPT, we need numerical methods to extract information from the data. We'll call the trading and financial decision systems based on this kind of methods *Technical Trading*.

In the Portfolio Optimization problem, two important technical measures are the Return Estimation and the Risk Estimation. The return estimation must be made based on the past values of the return. A precise calculation of this estimation will allow the model to make more realistic predictions.

The risk estimation is based on the past variance of the asset, and represents how reliable is one asset with relation to its estimated return. There are different

methods of calculating this measure, which carry with them different beliefs about the nature of the risk.

2.2.1 Return Estimation

Moving averages are often used in technical analysis as a tool to analyze time series data.

Moving average series can be calculated for any time series. In finance, it is often applied to estimate future returns or trading volumes. Moving averages smooth out short-term fluctuations, thus highlighting longer-term trends or cycles. The threshold between short-term and long-term depends on the application, so that the size of the moving average must be set on a case-by-case basis.

A *simple moving average* is the unweighted mean of the previous n data points. For instance, if r_{t-1}, r_{t-2}, ... r_{t-n} are the values for the returns for the last N months, the simple moving averages for these returns would be

$$SMA_t = \frac{r_{t-1} + r_{t-2} + ... + r_{t-n}}{N} \tag{2.2.1}$$

A good characteristic of moving averages in general is that it is very simple, given the value of the moving average for a point in time, to calculate the moving average for the next point in time.

$$SMA_{t+1} = SMA_t + \frac{r_t - r_{t-n}}{N} \tag{2.2.2}$$

A problem with the simple moving average is that it introduces some "lag" into the the estimation, due to its nature of using past data points. This lag will be greater as the earlier data points are more divergent from current data.

One alternative to reduce this problem is to add weights to the Moving Averages. For example, the *weighted moving average* adds linear weights to the sum, where more recent values have the highest weight, and older values have smaller weights:

$$WMA_t = \frac{nr_{t-1} + (n-1)r_{t-2} + ... + 2r_{t-n-1} + r_{t-n}}{n + (n-1) + ... + 2 + 1} \tag{2.2.3}$$

However, this weighting strategy makes calculating the next moving average a bit more complicated.

$$T_{t+1} = T_t + r_t - r_{t-n} \tag{2.2.4}$$

$$Num_{t+1} = Num_t + nr_t - T_t \tag{2.2.5}$$

$$WMA_{t+1} = \frac{Num_{t+1}}{n + (n-1) + ... + 2 + 1} \tag{2.2.6}$$

Another possible way to calculate the Moving Averages is the *Volume Weighted Moving Averages*. In this version, we weight each return variable based on the volume at that time. If v_i is the volume at time i, we have:

$$VWMA_t = \frac{v_{t-1}r_{t-1} + v_{t-2}r_{t-2} + ... + v_{t-n}r_{t-n}}{v_{t-1} + v_{t-2} + ... + v_{t-n}} \tag{2.2.7}$$

2.2.2 Risk Estimation

Differently from the Return Estimation, the method chosen for the risk estimation carries with it some suppositions about the model. By using different methods to calculate the Risk, we are changing the underlying assumptions of the model.

In this way, the risk method must not be chosen purely according to the numerical complexity of the calculations. The underlying ideas behind the risk definition must also be carefully considered.

Variance

In the Modern Portfolio Theory, Markowitz uses the variance of the return of an asset as its risk measure. The variance of a random variable, in this case, the return over time of an asset, is the measure of how large the differences are between the values of that variable.

In other words, if the Estimated Return of an asset is its Moving Average, the Variance risk means how much error we can expect from this estimate. The variance is defined as:

$$\sigma^2 = E[(R - E[R])^2] \tag{2.2.8}$$

Which, in practice, can be defined for a limited data sample as:

$$\bar{\sigma}_r^2 = \frac{1}{n-1} \sum_{i=t-n}^{t-1} (r_i - \bar{r}) \tag{2.2.9}$$

Where r_i is the value of the return at time i, and $\bar{r}$ is the average return of the period (i.e., the moving average).

The way to calculate the Variance-risk of a portfolio is to use the weighted co-variance between all the assets presents in the portfolio. This is described in eq. 2.1.2. The co-variance between two assets, r_1 and r_2 is calculated as follows:

$$Cov(r_1, r_2) = \frac{\sum (r_1^i - \bar{r_1})(r_2^i - \bar{r_2})}{N-1} \tag{2.2.10}$$

Where $\bar{r_1}$ and $\bar{r_2}$ are the moving averages. This calculation increases in size with the number of lines in the data, and must be done for every pair of assets in a portfolio. So, for a portfolio with many assets, and a large historical dataset, the calculation of the covariances becomes expensive quickly.

Value at Risk

Value at Risk (VaR) is another measure of risk used for financial assets and portfolios. It gives a measure of how much of the value of the asset might decrease at a certain probability over a certain time period.

To calculate the VaR, we need to draw a probability distribution function of the possible return values of the portfolio over the next time period (1 day, 10 days, 1 month, etc.). From this probability distribution, we choose our target probability, and find out how much value at most may be lost at that probability.

For example, Consider a trading portfolio that is reported to have a 1-day VaR of US$4 million at the 95% confidence level. This means that, under normal market conditions, we can expect that, with a probability of 95%, a change in the value of its portfolio would not result in a decrease of more than $4 million during 1 day, or, in other words, that, with a probability of 5%, the value of its portfolio will decrease by $4 million or more during 1 day.

To calculate the return distribution, there are many different numerical methods. Some of the most common are the variance-covariance model, historical

simulation, and Monte Carlo simulation (Uludag *et al.*, 2007).

The *variance-covariance* methods starts from the assumption that the portfolio returns follow some given distribution, like normal, or T-normal. The distribution and weights of the assets are calculated to find out the best-fitting parameters for the distribution model. Once the values of the distribution model are found, the loss for the desired probability is easily taken.

The *historical simulation* method is more transparent, if a large amount of historical data is available, but is also more computationally intensive. It involves running the current portfolio across a set of historical price changes to yield a distribution of changes in portfolio value, and computing a percentile (the VaR). The benefits of this method are its simplicity to implement, and the fact that it does not assume a normal distribution of asset returns.

The *Monte Carlo* simulation consists of generating a large number of possible scenarios based on the current data, and some market model. The varied results from this simulation are ordered, and the lowest 5% (in a 95% VaR example) are taken as the value at risk. It is then important to choose the model properly.

Monte Carlo simulation is generally used to compute VaR for portfolios containing securities with non-linear returns (e.g. options) since the computational effort required is non-trivial. For portfolios without these complicated securities, such as a portfolio of stocks, the variance-covariance method is perfectly suitable and should probably be used instead.

Chapter 3

Evolutionary Algorithm-based Optimization

Evolutionary Algorithms (EA) are a group of heuristics for random searches that are based on inspirations from the the biological evolutionary process. It intend to uses the same mechanism that is described in the theory of evolution: a population of individuals, through crossover and mutation of it's gene pool, is capable of adapting to changes in the environment.

In EAs, the environment is the *problem* to be solved, and the *population* is a group of solutions, subset of the search space. The best individuals in a population (dictated by a problem-specific *fitness function*) are mixed, generating new individuals (see figure 3.1). In this way, the search is biased towards the direction of the best individuals in the current population.

Algorithms inspired in the evolutionary theory were first proposed around 1965, as a numerical optimization technique (Schwefel, 1981). A formulation more similar to what is known today as genetic algorithms was proposed later in 1975 (J.H.Holland (1975),Jong (1975)).

While the first uses of Genetic Algorithms were for numerical optimizations, one of the main good characteristics of GA is its ability to adapt itself to changes in the problem conditions. The population in the genetic algorithms can easily change itself to adapt to changes in the fitness function. A good example of this can be seen in the many approaches to co-evolution, where two evolving populations compete between them (Rosin and Belew, 1997).

Besides Genetic Algorithms, another technique which also fit under the Evolutionary Algorithm paradigm is Genetic Programming. Both Genetic Algorithms

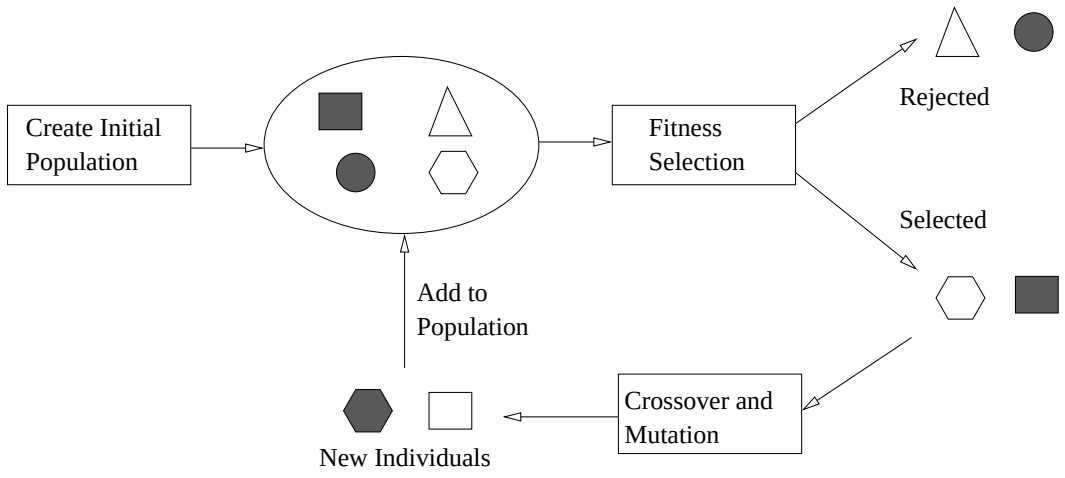

Figure 3.1: A simple description of the Evolutionary Algorithm framework

and Genetic program have in common the fact of being population based, and the use of genetic operators (crossover, mutation) in order to mix the solutions and produce new ones. Genetic programming evolves tree-structured individuals while Genetic algorithm focuses in array based representation of solutions.

This difference leads to many significant changes in the philosophy behind crossover and generation of individuals between the two techniques. In this work, we concentrate ourselves to Genetic Algorithms, although the use of Genetic Programming to the portfolio problem might yield interesting results.

Another limitation in the original EAs is related to the fitness function. The fitness function guides the population towards the goal of the search. However, when the search has multiple goals, specially when these goals are independent, or contradictory, it becomes difficult do create a fitness function that can properly bias the population to both objectives.

Portfolio Management, which we address in this thesis, is an example of such a problem. It has three goals: return, risk and transaction costs, of which the first two are contradictory, and the third is independent. To address this problem, there is the development of Multi Objective GA methods (MOGA).

In this chapter we describe the basic concepts of Genetic Algorithms, and

Multi Objective Genetic Algorithms that were needed for the development of our system. The first section describes the genetic algorithm and brakes it down to its composing parts, concentrating on the choices that have to be made to apply GA to a given problem. Then in the next session we describe some basic techniques of Multi Objective Genetic Algorithms, and show our choice for the current work.

3.1 Genetic Algorithms

A Genetic Algorithm is a black box heuristic for a Random Search. It can be defined by a search space S with its candidate solutions s, a subset of the search space P (the population), a fitness function $f(s)$, a selection operator, and the mixing operators $m(s)$ and $c(s_1, s_2)$, the mutation and crossover, respectively.

Each individual in the Population represents a solution to the problem. How to represent the solution is an important question in the genetic algorithm. For instance, if the solution for the problem is a set of real-valued parameters, a common representation is to turn each parameter into its binary representation, and one individual would be composed by the concatenated binary strings. Others would create an individual as an array of real values, each value being considered a "gene" in the individual. It is not yet known whether the binary or the real valued representation have any search advantage over the other (Yao, 1999).

A rough sketch of a generic genetic algorithm was drawn in figure 3.1. The first step is to generate the initial subset P, also called the initial generation, according to some rule. Then each member of the population is evaluated by f, which generates a fitness value for every member. Next, the fitness values are used by the selection operator, which will remove individuals with worse fitness values from the population. Finally, the remaining individuals are used by the mixing operators to generate new individuals, which are used to complete P.

The heuristic behind the genetic algorithm is that by mixing and changing the solutions with good performance, we can find better solutions than by randomly searching through the solution space. In this regard, the selection operator is used to prune bad solutions, the crossover operator generate new solutions that have

the characteristic of existing good solutions, and the mutation operator has the role of introducing new information into the system (Vose, 1999).

While Genetic Algorithms have met with good practical results in a variety of areas, whether they are particularly good for a particular class of problems or not is a question that has not been properly answered yet. So, most applications of genetic algorithms require extensive experimentation with Selection, Crossover and Mutation strategies to tune the evolutionary system to the problem.

3.1.1 Selection Strategies

The selection operators performs the "survival" role in the evolutionary mechanism. The goal of this operator is to select from the current population the individuals which will be used to construct the next population.

This selection is based according to the fitness value of each individual in the population. Individuals with a higher fitness score must have a higher probability of being chosen to be recombined than individuals with a lower fitness score. The difference between the reproduction probability of a low fitness individual and a high fitness individual in a given selection strategy is called the *selection pressure*.

If the selection pressure is too high, then there is a risk that the population will be dominated by a few individuals that are not optimal but have a higher fitness than the rest of the current population, leading the search to a local optima. On the other hand, if the selection pressure is too low, the search will move too randomly through the search space (Yao, 1999).

A simple selection strategy is the *Fitness Proportional* selection, also known as *Roulette Wheel*. The probability for each individual in the population to be selected is proportional to the relation between its fitness f_i and the sum of the fitness of all individuals in the population. While this method is quite simple, it can build too much selection pressure if there are a few individuals with a relatively high fitness in the population.

To address this problem, we have the *Rank-based* selection. In this selection method, the individuals in the population are ordered by fitness, and the probability of selection is proportional to the individual's rank. This removes the influence

of relative fitness values, and allows for a finer control of the selection pressure by.

Both Rank-based and Fitness proportional Selection methods compare the whole population at once. *Tournament Selection*, on the other hand, works with small subsets of the population. A parametric number of elements is chosen at random, and them their fitness is compared, and the winner is chosen for reproduction. The winning criteria of the tournament can be either deterministic (the highest fitness is always the winner) or probabilistic (winning chance is proportional to fitness). The tournament selection is appropriate for a parallel implementation, since it breaks the selection problem into many small independent parts, and can in this way can speed up the processing of a genetic algorithm.

Finally, *Elite* selection is not a selection method per se, but a common addition to other selection policies. The application of an elite policy means that, regardless of mutation or crossover, one or more of the best individuals from the current population are copied without changes into the next population. It can be used to keep the current best solutions when using aggressive crossover and mutation strategies.

3.1.2 Crossover Strategies

The role of the crossover operator is to perform exploitation of the search space, by testing new solutions with characteristics of two good individuals (Vose (1999), Yao (1999)).

Crossover is an operator that is applied to two members of the population (the *parents*), to create one (or two) offspring with characteristics from both its parents. The offspring individuals contain informations from both parents.There are three basic forms that the crossover operator usually takes.

The *k-point crossover* is one such form. From the genetic representation k points are chosen; $p_1, p_2, ...p_k$. If x and y are the parents' genetic vectors, the offspring vector, c, will be composed by alternately copying elements from x and y, and changing from one parent to the other at each of the k points (see figure 3.2.

The *Linear crossover* is (also known as *Uniform crossover* is similar to the

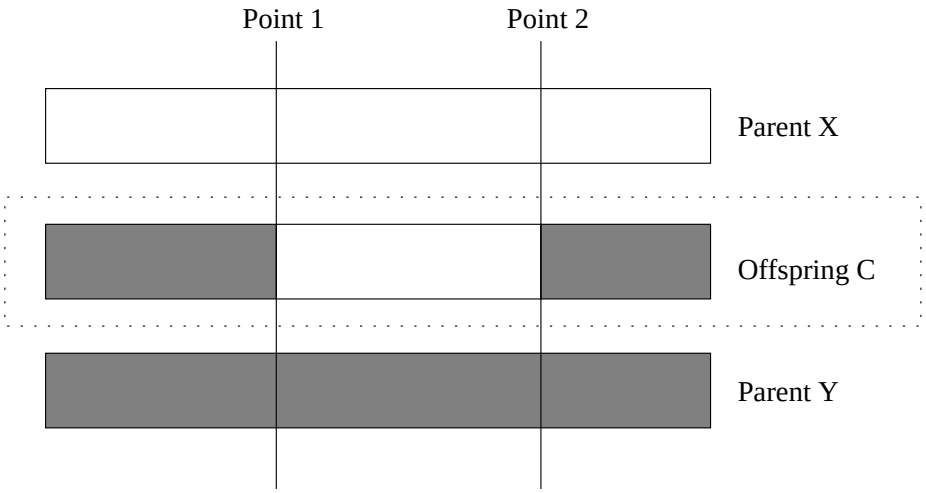

Figure 3.2: Example of k-points crossover with $k = 2$

k-points crossover. However, instead of copying blocks of elements from each parent into the offspring, in the linear crossover each element is tested against a probability of being copied from either parent (usually 50%).

The *Intermediate Crossover* is usually found only on real-valued GA. In this form of crossover, each element is the average from the respective elements in both parents. Thus, for an offspring c composed from parents x and y we have:

$$c_i = \alpha x_i + (1 - \alpha) y_i$$

Where α is an weight between both parents (usually 0.5).

The choice of the crossover function influences what kind of information is kept from parent to child individual. For instance in the k-point crossover, long blocks of information is copied from parent to offspring - if two elements who are represented side by side are related, this relationship will be passed to the next generation. Linear crossover, on the other hand, assumes that the position of the genes holds no information.

Sometimes, the crossover function has to be modified to fit with specific requirements of the problem. For instance, in (Aydemir *et al.*, 2006) the crossover function has severe restrictions on what places in the genome representation can

be “cut” to perform the k-point crossover. This is because the genome represents a series of continuous movement by a robotic motor. In this situation, if the crossover happens at any location it is possible to generate an “invalid” genome.

While crossover is traditionally performed before mutation, if both operations are independent, there is no statistical difference between doing one or the other first (Vose, 1999).

3.1.3 Mutation Strategies

The mutation operator introduces new information into the population in the form of new genes. While the crossover operation performs the exploitation role of the genetic algorithm, the mutation performs the exploration.

Mutation usually takes place by having a parametrical chance of mutating each gene in an individual. If the representation is binary, the mutated genes are flipped. If the representation is real-valued, the mutated genes are perturbed by a value given by some distribution (for instance a small random value from a normal distribution).

3.2 Multi Objective Genetic Algorithms

The main decision that must be made when applying a Genetic Algorithm to a problem is the design of the fitness function. A good fitness function will guarantee that eventually the optimal desired solution will be reached, and that the algorithm will not be locked into a local optima of the search space.

However, for many practical problems, there are multiple goals to be achieved, and this goals are often partially independent. When this is the case, a single fitness function will not be able to accurately represent the efficiency of the solution for every objective.

For example, in the Portfolio Problem as discussed in this word, we find potentially three different objectives: The maximization of the estimated return of the portfolio, the minimization of the risk, and the minimization of transaction costs.

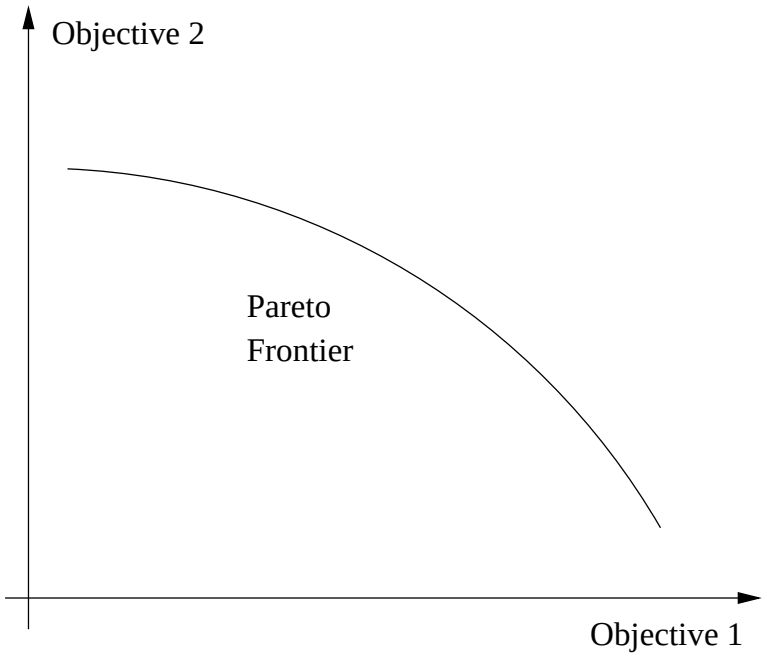

Figure 3.3: The Pareto Optimal Set for an hypothetical problem

It is actually very difficult in these problems to find a single solution which will reach optimal values for every objective. Usually, we have to abandon the concept of a single "optimal" solution in favor of a series of trade-offs between each goal.

This new definition of optimality is called the *Pareto Optimum*, and can be defined as follows. A vector of decision variables $x^* \in F$ is Pareto-optimal if there does not exist another $x \in F$ such that $f_i(x) \leq f_i(x^*)$ for all $i = 1, ..., l$ and $f_j(x) < f_j(x^*)$ for at least one j.

In other words, a Pareto optimal solution is one where the value of at least one of the objectives is higher than any other solution in the search space. From this definition we can also derive the concept of a *Pareto Optimal Set*, which is the group of all Pareto-optimal solutions to a problem. Figure 3.3 illustrates the Pareto Optimal Set for a bi-objective problem.

Genetic Algorithms have been regarded as an appropriate technique to multi-objective problems. This is because GA is less sensitive to concave or discontinuous search spaces, and Pareto sets (Coello Coello, 2001). There are different approaches, though, as to how to use GA with multi-objective problems. Each method has different characteristics regarding complexity, efficiency, and fine-tuning requirements. Here we will discuss the most popular ones.

3.2.1 Aggregate Methods

Aggregate methods try to solve the multiple objective scenario by designing a single fitness function representative of all the goals in the problem. The most common example of an Aggregate Method for Multi Objective GA would be weighting.

In the weighting scheme, a fitness function is designed for each goal in the problem, and the final fitness function is a weighted sum of the problem-specific functions. This method is extremely simple to implement, and do not require any changes to the basic logic of the Genetic Algorithm System.

The problem with weighting, and other aggregate methods is twofold. First, by summing the values of different metrics together into a single fitness function, it becomes very difficult to generate members of the Pareto Optimal Set when this set is concave (Das and Dennis, 1997).

Second, it is not intuitive how the weights must be attributed to the different fitness components. Specially in problems where the different objectives are not related, or have very different value scales. For instance, in the problem discussed in this work, The risk metric and the distance metric are not related, and thus there is not much meaning in trying to sum the two values together.

3.2.2 Objective Sharing Methods

Objective Sharing Methods modify the selection strategy of GA, by having the different fitness functions, related to the different goals, affect the selection separately.

In VEGA (Schaffer, 1985), for instance, the population is separated into a number of subgroups equal to the number of objectives. Each subgroup undergoes

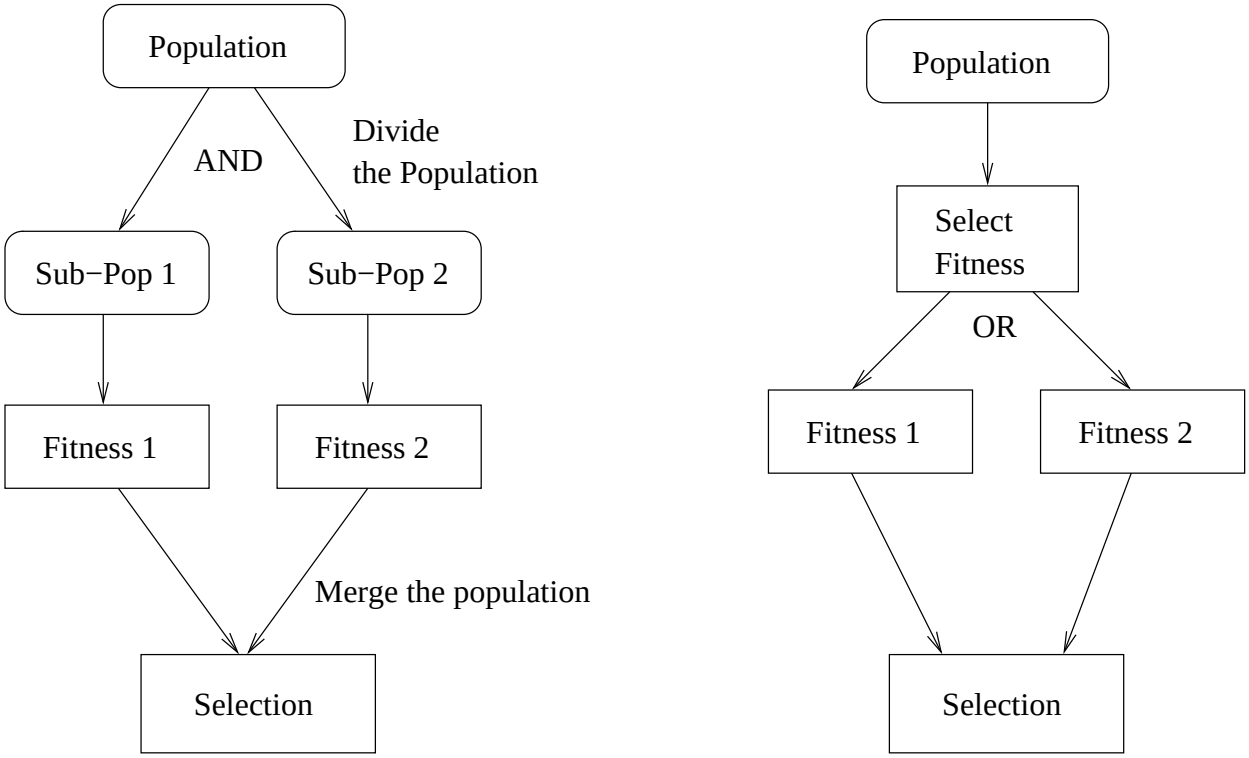

Figure 3.4: Differences between Objective Sharing approaches

a different fitness function, and then all groups are merged for the crossover.

Lexical ordering (Coello Coello, 2001) approaches this differently. Each goal has its fitness function, but instead of using them all at the same time, each generation a different fitness function is used. The way to choose which fitness function to use every generation is an important design decision for this method.

The differences between the two methods can be visualized in figure 3.4.

Objective sharing methods do not show the same problems regarding concave Pareto sets as agregate methods do. However, it has some issues with speciation. Speciation means that, due to the nature of separating the population and evolving each part under a different metric, individuals with extreme values for one of the objectives will appear more often than individuals with average values for all goals. This can be a disadvantage when balance between the objectives is critical.

3.2.3 Pareto-based Methods

Pareto-based methods use the concept of *dominance* between solutions in order to determine the fitness of individuals. We say that a individual a dominates an individual b, if for at least one goal a is better than b, and for every other goal, a is at least as good as b. Two individuals are *non-dominant* between each other if each has at least one objective metric that is higher than the other. For example, the Pareto Set described earlier in this section is composed of individuals that are non-dominated by any other solution in the population.

Most MOGA methods use the concept of dominance to establish a rank among the individuals, and generate the fitness based on this rank. Thus, the values for the goal metrics are not used directly to assign fitness values to the individuals.

Pareto-based techniques have the advantage of being very efficient, and are reported to generate solutions in all positions along the Pareto frontier. However, the fitness assigning step is difficult to estimate correctly, and is known to heavily influence the efficiency of the method. Also, Pareto-based methods require an extra step called “fitness sharing”. This step is to prevent the populating from concentrating in a single spot in the Pareto front. The effective implementation of fitness sharing is still a topic of study (Horn, 1997).

Chapter 4

Related Work

Although the Modern Portfolio Theory was first proposed in the 50ies, and the CAPM in the 80ies by Markowitz, the first uses of Genetic Algorithms for portfolio optimization are much more recent.

We trace the earlier works in this subject to 2002 (Jian and K.Y.Szeto, 2002). These first works approach Portfolio Optimization as simply the problem of translating to the Genetic Algorithm paradigm the model described to Markowitz.

Later works build upon these models, by trying to add extra constraints that make the model closer to real life applications, or by making changes to the Evolutionary Algorithmic heuristic that yield better results in this particular problem.

Most interesting recent works are the recent articles by Wei and Clark (Yan and Clack (2006), Yan and Clack (2007)), where they research the characteristics of a specific subproblem of the portfolio optimization problem: Hedge Trading. We believe that working with application niches might indeed turn up interesting improvements to the heuristics.

So, we consider this to be a research topic still in an early stage of development. Evidence of this is that almost no works approach the multiscenario portfolio problem, limiting themselves to the single scenario problem initially proposed by Markowitz. This encourages us to follow up this line in this work.

4.1 Review of GA based methods

The implementation of the Markowitz model for portfolio optimization can be neatly divided in several independent blocks. This was demonstrated earlier, when we described the problem, and can be observed in all of the previous works, where

the same division takes place.

The basis of the algorithm is the optimization of the portfolio weights in order to maximize some utility function. While the utility function is most often the Sharpe Ration, other functions have been proposed in the literature.

Regardless of the utility function chosen, a number of estimation techniques to analyze the data used in the function are necessary. Return estimation and Risk estimation are the two obvious functions in this regard, but if other data (like volume) are considered in the model, they will need their own estimation methods.

Besides the model for the portfolio optimization, a lot of research has been focused in the Genetic Operators. The use of MOGA instead of a single fitness function for risk and return is the most obvious of the changes, but studies on the most appropriate encoding have also showed interesting results.

We will detail below each of these building blocks for the system, describing the most used approach, interesting diversions from this consensus, and our opinions on the research done so far.

4.1.1 Portfolio representation

The most common way to represent a portfolio in a Genetic Algorithm is by using a vector of real valued variables, one for each weight in the portfolio. Some examples of this representation technique are Lipinski *et al.* (2007), Lin and Gen (2007) and Hochreiter (2007).

The real-valued representation has the advantage of directly representing the portfolio. So the transformation from the genome to the problem solution is very straightforward.

There have been works on more indirect representations as well. Werner and Fogarti (2002) suggests the use of Genetic programming to generate rules that calculate the values for the portfolio weights, instead of directly optimizing the weights themselves. They propose that by generating this rules, it improves the adaptive characteristic of the portfolio. However, the work has not been continued.

Yan and Clack also use GP (Yan and Clack (2006), Yan and Clack (2007)). In

their work, the genetic programming generates rules for buying/ selling each asset, in a similar way to that performed in Automated Trading Strategies research. The weights of the portfolio, in this work, are the same across all assets, and the only difference is whether the particular asset is held in a short or long position.

4.1.2 Asset selection

Past works face a common problem of executing simulations on a small universe of assets. Most works use only up to 20 assets at once to compose the portfolio. Yan and Clack (2006), Yan and Clack (2007), Hochreiter (2007) are examples of works where all assets are used at the same time to compose the portfolio.

Lipinski *et al.* (2007), on the other hand, uses a larger universe for the total assets, but chooses 10 assets at random from this universe to compose the portfolio.

A more interesting idea was proposed by Streichert *et al.* (2003). In this work, an index array is used together with the real-valued array to select the assets that will take part of the portfolio. This index array goes under the evolutive process in the same way that the real-valued array. It is said that selecting the assets like this provides better results than purely using all the assets in the portfolio.

In this dissertation, we use this last technique, which details will be discussed in the next chapter.

4.1.3 Market Model

Most works in this subject use simulations based on historical data in order to test their results against the model. While the use of historical data allows for a precise repetition of real-world events, it also has some limitations. The most important of those is that simulations using past data do not react to the actions of the portfolio system.

An alternative is presented by Subramanian *et al.* (2006), which performs its portfolio selection under an artificial market composed by many agents. When all the agents that influence the market are represented in the simulation, the effect of buying or selling large amounts of desired assets can be analyzed.

However, this approach has the disadvantage of being more artificial. The number of agents that can be simulated is limited, and the accuracy of the simulation is also limited. So it becomes difficult to understand to what degree a system that succeeds in an artificial market can also succeed in a real one.

In this work, we follow the general trend of using historical data simulation as the model for the market.

4.1.4 Return Estimation

The most common method for Return Estimation in the Portfolio Optimization context is the moving averages (Yan and Clack (2006), Yan and Clack (2007), Lipinski *et al.* (2007), Lin and Gen (2007)). The moving averages method can be simply described as the average of the most recent N prices in the available historical data.

Werner and Fogarti (2002) uses a more sophisticated approach, with a least squares optimization method to model the value of the return for each asset in the portfolio.

Neural networks are another source of prediction for the return values. Kwon et al. (Kwon and Moon (2004), Kwon *et al.* (2004)) uses a hybrid approach between Evolutionary computing and Neural Networks to calculate the return values of different assets. Evolutionary Computation is used to optimize the weight of the Neural Network. The inputs are the previous values of the stock price time series, and the output is the future value.

Azzini and Tettamanzi (2006) also uses neural networks with weights calibrated by evolutionary computing to calculate the future price of stock, and uses this values in a portfolio optimization system.

It is not known by how much the use of advanced methods for calculating the estimated return can improve the results of MPT optimization. Since the goal is not to reach an specific value, or trade the portfolio in specific trade points, predicting the exact value of the asset is not as important as predicting the general trend of the market.

Using elaborate methods for Return Estimation adds a complexity to the system that makes it more difficult to evaluate the effect of adding cost to the model by itself. So for this work, we will use the more common and accepted method of moving averages.

4.1.5 Risk Estimation

The standard deviation, as suggested by Markowitz, is the most used measure of risk (Lin and Gen (2007), Hochreiter (2007)). Like for the expected return, many works suggest other ways to calculate the risk for a given asset. However, unlike the expected return, the alternate methods of risk calculation do not aim at achieving more precise values.

Instead, alternate methods for calculating risk are actually changes in the MPT model itself, to put extra effort in certain aspects of the assets that compose the portfolio. For instance, Lipinski *et al.* (2007) and Subramanian *et al.* (2006) use the semi-variance, which reduces the influence of positive variance, and increases the influence of negative variance. By using this method for risk estimation, these works reinforce the value of positive risk (the risk that the actual value of an asset is above the value predicted). Whether this assumption is correct or not is beyond the scope of our work, and requires a deeper knowledge of economics to correctly access the answer.

Chen *et al.* (2002) also suggests the use of other models for risk. In this particular paper, they introduce the l_{inf} function, which is similar to the risk function used by Markowitz for the portfolio risk, but does not uses the covariance value between assets. This results in an average risk rate which is lower than the one reported by works which use the Markowitz model.

4.1.6 Cost modeling

Cost measures are often cited in financial engineering works as necessary, but rarely they are actually used in the simulations of those works. This is not different for Portfolio Optimization.

Streichert *et al.* (2003) and Fyfe *et al.* (2005) choose two arbitrary, fixed values for the cost, and arrive at quite different results due to this. Chen *et al.* (2002) uses a function that varies the cost according to the volume traded.

It is difficult to correctly insert a measure of cost into the portfolio model. Stock dealers may have different policies regarding the costs of financial transactions. Often, an arbitrary value for cost is chosen, like in Streichert *et al.* (2003); Fyfe *et al.* (2005). Sometimes, a more abstract approach is taken, and the cost value is let unfixed Chen *et al.* (2002). Lin *et al.* (2005) build the cost directly into the function that generates the return estimate of the portfolio. The cost values are different for buying and selling.

In all these works, the costs are added to the final results, but not taken directly into account by the evolutionary process. Also, all the above works are single scenario, which means that the costs are considered against an initial position where no assets are held.

In this work, we propose a more complete cost model, where an explicit objective function for cost is added to the evolutionary model, and the position held in the previous scenario is taken into account.

4.1.7 Fitness Functions

The most common fitness function for Evolutionary solutions of the Portfolio Optimization problem is the Sharpe Ratio (Yan and Clack (2006), Yan and Clack (2007), Subramanian *et al.* (2006), Lin and Gen (2007), among others). The Sharpe Ratio is suggested in basic books on finance as a way to compare a portfolio to the Market Portfolio – the theoretical optimal portfolio. The Sharpe Ratio also has the advantage of cleanly putting together risk and measure in a single function, but it is by no means the only fitness function used in the literature.

Hochreiter (2007) and Lin *et al.* (2005) use a weighted sum between the Risk measure, and the Return, as the fitness function. If the risk of the portfolio is σ_P, and the return is r_P, their fitness function would be something like $P(x) = \alpha\sigma_P + (1-\alpha)r_P$. Alpha is a balancing parameter between the two goals.

This approach has the obvious disadvantage that a new parameter is added

to the system. But another serious problem with this function is that Risk and Estimated return are not two directly comparable values. Adding both directly is similar to adding the speed of an object with its acceleration.

Streichert *et al.* (2003) suggests that risk and return should be evaluated separately, in a Multi Objective GA architecture. In their work, they use the Pareto front as an elite strategy to keep solutions with both high return and low risk in the population during the evolution. Yan and Clack (2007) disagrees with this approach, and says that Pareto-based MOGA is not able to solve the risk-return optimization, but do not provide evidence to this assertion.

4.1.8 Evolutionary Strategies

Most GA-based Portfolio Optimization systems follow a formula similar to the one described by Lin and Gen (2007). However, as we stated before, in some works different evolutionary strategies are pursued.

Yan and Clack (2006) and Yan and Clack (2007) study how to increase the robustness of the genome during the search. In the first study, the concept of genetic diversity is used to improve robustness. They use the correlation values between different solutions in order to increase the difference between the individuals of the same generation.

In the second study, they change the selection step by adding different scenarios for evaluation for each generation. This is done with the goal of reducing overfitting and increasing robustness of the achieved solutions.

Chapter 5

Multi-Objective Portfolio Management

In this work, we expand over well-known Genetic Algorithm-based Portfolio Optimization methods. Our approach is to pick the best ideas among the published work to build a scenario-based portfolio optimization system. Over this system, we implement two ideas researched over this work to achieve multi-scenario Portfolio Management.

The first one is Seeding, based on the idea of migration. Individuals from previous scenarios are brought in to the current scenario, to breed with the new population, and tilt the search towards regions in the search space that were successful in the past.

The second one is Objective Sharing, based on Multi-Objective GA techniques. We define a new fitness function to measure the distance between the individuals in the population and the portfolio position taken in the previous scenario. At each generation, there is a chance that the standard optimization fitness function will be replaced by this new fitness, directing the population towards solutions that have smaller distances from the position.

Our goal is to use the self-adapting characteristics of Genetic Algorithms to develop a portfolio selection strategy that is resistant to changes in the market.

In this chapter, we describe how we intend to achieve such a goal. The model used to represent our problem is described, along with its assumptions and restrictions. Then, based on this model, we describe an optimization and management system that generates effective portfolio strategies.

5.1 Portfolio Model

Assume a market with N assets. For each asset $n \in N$, r_n^t is the given historical return at time t for that asset. Also, for each asset, σ_n^t is the given risk measure at time t for that asset.

Based on this data, the system must generate investment portfolios that are optimized in three different measures: to minimize the portfolio's risk, to maximize the portfolio's return, and to minimize the cost between the current portfolio (time t), and the previously generated portfolio (time $t-1$).

5.1.1 Portfolio Representation

The output for the system is a portfolio configuration for each time t. These portfolios are defined as sets, W^t, of N weights, in which $w_n \in W^t$ is the weight that refers to the proportion of investment that will be allocated to asset n.

We impose two restrictions to the value of the weights w: $\sum_n w_n = 1$ and $0 \leq w_n \leq 1$. The first restriction indicates that the total of the weights cannot exceed the total of resources, and the second restriction imposes that the values of the weights must be non-negative.

5.1.2 Return Measure

The return for one asset in the model is given as its logarithmic return value. To calculate this, we use the closing price of the asset at times t and $t-1$ (indicated by P_n^t and P_n^{t-1}):

$$r_n^t = log\left(\frac{P_n^t}{P_n^{t-1}}\right) \tag{5.1.1}$$

The return of a portfolio is given as the weighted sum of the returns of its composing assets, and calculated as follows:

$$r_W^t = \sum_{n=1}^{N} w_n r_n^t \tag{5.1.2}$$

5.1.3 Risk Measure

The risk for one asset in the model is given as the variance of the return of that asset. The risk up to a given time t is calculated as the variance of all the data available up to that time. Because of this, it is needed in practice to have some historical data available prior to the time t from which we want to calculate the portfolio.

The risk of the portfolio is given as the weighted sum of the risk of its composing assets:

$$\sigma_W^t = \sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j \sigma_{ij} \tag{5.1.3}$$

5.1.4 Cost Measure

One of the main novelties we introduce is the addition of transaction costs into the portfolio model. Transaction costs are added when selling or buying an asset, so that if an investor changes his position too much to follow the market, the transaction costs may become higher than the difference in return of the new position.

In practice, transaction costs depend on the policies of the broker. A flat rate can be charged per transaction, or it can be proportional to the trade volume. Different rates can be charged of different classes of assets.

We use, instead, an indirect approach to cost. We assume that the transaction costs are based on the amount of an asset sold or bought, but do not set a fixed value to this cost. Instead, the *distance* between the current portfolio and the desired portfolio is used as the cost measure. This concept has not yet been used in GA optimized portfolio approaches.

Following this definition, distance is the amount by which the weights of two different portfolios differ. We quantify the distance as the Euclidean distance of the weight vectors:

$$d(W_a, W_b) = \sqrt{\sum_{n=1}^{N} (w_n^a - w_n^b)^2} \tag{5.1.4}$$

With higher distance values meaning higher costs to move from one portfolio position to another.

5.1.5 Assumptions of the model

The model, as presented, makes a number of assumptions that simplify the problem, and do not hold in practice. We make explicit those assumptions here. Note that these assumptions are also made by most of the current work in the field. The removal of each restriction must me progressively addressed as the model evolves through time. But an understanding of the effects of these restrictions and their consequences allow us to use the results obtained by the system in the real world.

The effects of Volume in trading are not included. This means that there are no limits on the minimal or maximal amount of any asset that can be held by the investor. In practice, the minimal amount of an asset that can be held is usually limited by the price of a single share of that asset, and there are often limits to the maximum amount of an asset that can be held by a single individual. Also, the price for buying or selling assets can be affected by the volume dealt. The effects of volume trading have not been satisfactorily addressed in other works in the same field.

By restricting the weights to non-negative values, we remove the possibility of holding short positions in the portfolio. However, this is not unrealistic, as it can be shown that models for the MPT with and without short-selling are equivalent (Yuh-Dauh-Lyu (2002)). We add this restriction to the model for the sake of simplicity.

The choice of the Euclidean Distance as a measure of cost also carries with it a few assumptions about the model. According to the definition we use for the function, many small changes in different weights result in a smaller distance (cost) than a large change in only one weight.

However, the choice of the Euclidean Distance as a measure of cost has some drawbacks. The major one is that it introduces the assumption that large transactions of a single asset are less desirable than small transactions of many assets. This often does not hold in practice. So for future works, a better general function

for defining the distance between two portfolios needs to be found.

5.2 Evolutionary Optimization

Our Portfolio Optimization system is not very different from the single-scenario systems that were developed in previous works. We built the system by making the choices that seemed to meet with most success in the literature, and added the methods for reducing distance.

We use a simple GA, which generates many different portfolio combinations, and test them for their performance on scenario based on historical data. The risk/return ratio of a portfolio, known as "Sharpe Ratio", is used as the fitness measure. To calculate it, we estimate the return of the portfolio by using the "moving averages" method.

In this section we will explain in details each component of the optimization system that is used to resolve one scenario. The techniques introduced to improve the Multi-Scenario portfolio optimization are described in the next section.

One of the known difficulties with GA is the large number of parameters that are used in the system. Table 5.1 lists all the parameters used in our system. These parameters are further described in this chapter, and the methods for choosing their values are explained in the next chapter, together with the experiments performed.

5.2.1 Genome Representation

Each individual's genome is represented by two arrays. The Index array, I, and the weights array, W.

The Index array is composed of N boolean elements, where N is the number of available assets. Each element I_n represents whether or not the asset n belongs to the portfolio.

The Weight array is composed of N non-negative, real-valued elements, where N is the number of available assets. Each element, W_n represents the non-normalized weight of that array in the portfolio.

Table 5.1: Parameters Used

Parameter Name	Type	Meaning
Number of Generations	Integer	Number of GA interactions
Number of Individuals	Integer	Number of candidate solutions
Time Length	Integer	Number of samples for Moving Average
Legacy Size	Integer	Population Generation parameter
Fill Ratio	[0,1]	Individual Generation parameter
Elite Size	Integer	Selection Parameter
Tournament K	Integer	Selection Parameter
Crossover Ratio	[0,1]	Probability of crossover
Mutation Ratio Flip	[0,1]	Probability of mutation in Index Array
Mutation Ratio Perturb	[0,1]	Probability of mutation in Weight Array
P_{os}	[0,1]	Fitness selection probability

To generate a valid portfolio (phenotype) from this representation we need to normalize the genome, since there is no limit to the values of W. First, we set to 0 the value of each weight W_n whose corresponding element in the Index array has a value of "False". Then we normalize the remaining values, and that will be the represented portfolio. The procedure is shown in figure 5.1.

A random individual is created by generating random values for both arrays. For the weight array, we generate a random number between 0 and 1 (inclusive) for each element.

For the index array, each element has a probability of being "true" equal to the parameter *fill_ratio*. Else, the element is set to false. This parameter allow us to limit the size of portfolios for very large datasets.

The two-array setup for the representation isn't very common in current literature. More often, a small number of assets (10 or 20) is chosen arbitrarily, and optimized in a single weight array. In this system, we aim to solve portfolio problems with many assets available. Since the search space increases exponentially with the number of assets, we introduced the index array to allow the Evolutionary Algorithm to quickly test for the addition or removal of certain assets from the portfolio.

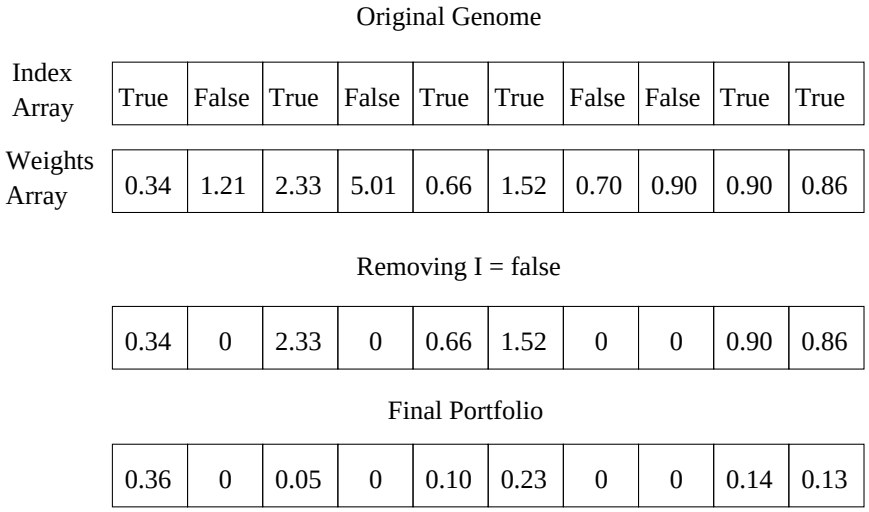

Figure 5.1: Generating a portfolio from its genomic representation.

5.2.2 Return Estimation

We use the traditional technique of Moving Averages to calculate the expected return for the available assets and the portfolio.

For a given asset n, its expected return value is calculated as:

$$r'_n = \frac{\sum_{i=1}^{T} r_n^{t-i}}{T} \tag{5.2.1}$$

Where r_n^x is the return value of asset n at time x, t is the current time, and T is a parameter that determines the size of the moving average.

The expected return for a portfolio is given by the expected return of its composing assets:

$$r'_W = \sum_{i=1}^{N} w_i r'_i \tag{5.2.2}$$

Where N is the total number of assets.

The accuracy of this calculation depends on properly choosing the parameter T, which determines how many time periods (months, days, etc) to be considered when calculating the expected return. If T is too large, a global rising or falling trend will fool the moving average into a value that goes against the current

moment of the trend. On the other hand, if T is too short, than the result of the moving average will be too local.

So this parameter, which we call the *time_length* is heavily dependent on the problem. To decide a good value for it, we estimate the value which brings the smallest difference between expected return and actual return for $t-1$, and use that value.

Also, instead of the moving average, more complex methods of return estimation could be used. For instance, Geometric, or weighted moving averages, or some sort of time-series analysis system like the one described in (Nikolaev and Iba (2001)). In this work, we use a more common method in order to put the novelties we introduce in the model in the spotlight.

5.2.3 Fitness Measures

Two different fitness measures to optimize the candidate portfolios in the system. One is the Sharpe Ratio, and the other is the Euclidean distance.

As explained in chapter 2, the Sharpe Ratio indicates the "market price of risk" of a certain investment. In other words, it is the ratio of by how much the expected return should increase, if we increase the risk of the target investment, by leveraging it with the riskless asset.

The Sharpe ratio of a portfolio W is calculated as:

$$Sr_W = \frac{r'_W - R}{\sigma_W} \tag{5.2.3}$$

Where R is the Risk Free Asset. While a perfectly risk free asset only exists in theory, in practice government bonds with very small variance are often used in its place (Yan and Clack (2007); Yuh-Dauh-Lyu (2002)).

The higher the value of the Sharpe Ratio, the better the portfolio being evaluated. A good target value is the Sharpe Ratio given for the market index, since it is the theoretical optimal portfolio.

The Euclidean Distance, as described earlier, in equation 5.1.4, is used as a measure of difference, and consequently cost, between two portfolios. In this

system, it is only used concurrently with the first fitness measure, since it cannot lead to an optimized portfolio by itself.

When used as a fitness measure in this system, each candidate portfolio for the current scenario is compared to the best individual from the previous scenario (not the previous generation, beware of not confusing the concepts). A lower Euclidean distance value represents higher fitness.

5.2.4 Evolutionary Strategy

Selection proceeds by a mix of Elite strategy with tournament selection.

First, a number of individuals defined by the parameter *Elite_Size* are copied to the next generation, without mutation or crossover. The remaining individuals of the next generation are created by crossover. For each pair of new individuals, two parents are chosen by tournament selection.

The tournament selection used in this work is deterministic tournament selection. A parametric number of *Tournament_K* individuals is randomly chosen with equal probability, and the individual with the highest fitness wins the tournament. This technique is very fast to implement, and offer a lot of control to the selective pressure.

After the two parents are chosen, crossover happens with probability equal to the parameter *xover_chance*. If the crossover does not happens, both parents are copied to the next generation, but may be subject to mutation (unlike the individuals chosen by the elite rule).

The crossover technique is a simple linear crossover applied for both arrays. The offspring receives the index and weight values from either one of the parents with equal probability, for each asset.

The choice of the linear crossover instead of a k-point crossover follows from the fact that the position of the assets in the genomic representation of the portfolio has no intrinsic meaning. The fact that two arbitrary assets are located in contiguous positions in the array does not mean anything. So a k-point crossover will create a spatial relationship between the array elements where there is none.

Mutation is applied in all individuals generated by crossover. Each element

in each generated individual has a chance of being mutated. Each element in the index vector of the individual has a fixed chance to be flipped (switched from 1 to 0 and vice-versa). This chance is determined by the mutation parameter *mutation_flip*. Each element in the weight vector has a fixed chance of being perturbed by +/- 10%, within a normal distribution. This chance is determined by the mutation parameter *mutation_perturb*.

5.3 Portfolio Management

Portfolio Management is the name we give to the problem of optimizing the portfolio structure across multiple scenarios. Besides the problems of Single-scenario optimization, in Portfolio Management we also have to deal with transaction costs and trading restrictions, which may limit the portfolios we can choose in the current scenario based on the position taken in the previous scenario.

A simple portfolio management technique is the *recalibration* of portfolios. By recalibration we mean returning to its original composition a portfolio which had been modified due to fluctuations in the values of assets.

For instance, imagine a portfolio that, at time t, is composed by two assets, A and B, with weights 0.2 and 0.8, respectively. If A shows a return of 2.0 and B shows a return of 1.5, the portfolio weights will become 0.25 and 0.75 the next day, because of the different valorizations. To recalibrate the portfolio, the investor needs to either sell A or buy B. This simple strategy is not a bad one, because it can be translated into the "buy low, sell high" common sense rule.

In our method, we approach this problem by giving to the current scenario genetic information from the previous scenario. This is done by two techniques: Seeding and Objective Sharing.

5.3.1 Seeding

The first technique we introduce to generate a portfolio strategy consistent over time is *seeding*. When creating the new population to optimize the portfolio for a time t, we will introduce some individuals from the final population from the

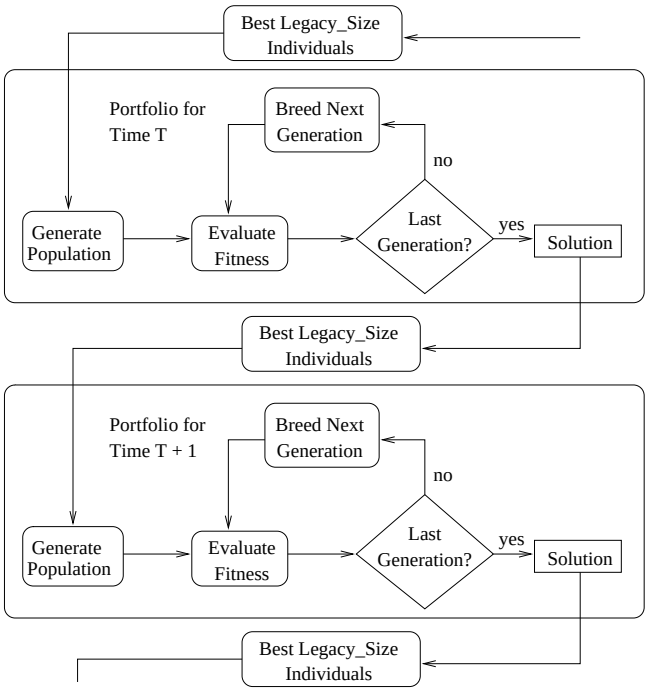

Figure 5.2: Description of the Seeding method

previous scenario (time period $t-1$). These new members are copied into the first generation of the new scenario, and from then on act as if they were normal, randomly generated individuals (see Figure 5.2).

In practice, this technique generates a bias in the evolutionary search towards the region of the search space that contained the solution in the previous time period. Unless the return values changed very drastically, the seeded individuals will have a fitness value slightly higher than most randomly-generated individuals, and will reproduce more, leading the population to focus its search on the area of the previous winner.

Seeding involves the addition of the parameter *legacy_size* to the system. Its value indicates the number of individuals from the previous scenario that are copied to the new scenario. As will be shown in the next chapter, we found that the number of seeded individuals is less important than the existence/lack of seeders.

5.3.2 Objective Sharing

The second technique we introduce in the system to realize Portfolio Management is *Objective Sharing*. This is a modified version of the lexicographic ordering (Described in Coello Coello (2001)).

With Objective Sharing, a second objective is introduced to the system: the Euclidean distance between an individual (current solution) and the best solution from the previous scenario. The Euclidean distance between two solutions here is calculated as described in equation 5.1.4.

With this new objective measurement, we have to change the evolutionary process to take both into account. Every generation, one of two fitness measures is used to evaluate the population. The probability that one of the two will be chosen is given by the parameter p_{os} ($p_{os} > 1$). The main fitness measure, Sharpe Ratio, is chosen with probability $P_{sr} = 1 - (1/p_{os})$, and the secondary fitness measure, Euclidean Distance, is chosen with probability $P_d = 1/p_{os}$.

Once one of the two objectives is chosen, selection and breeding happens the same way as described before, and another objective is then chosen for the next generation. In this way, the population is directed towards both objectives. The value of p_{os} can be adjusted to determine the relative priority of both objectives.

Chapter 6

Experiments and Results

We performed a series of experiments with market simulations to further understand and validate our proposal. In this chapter we'll discuss these experiments and their results.

We report on two kind of experiments. The first are sensibility analysis for the parameters introduced in our system. Our objective is to understand how the behavior of the system changes when we apply different values to its parameters, and if possible to find a strategy for optimally choosing the parameter values.

The results of the sensibility analysis allowed us to better understand the effects of objective sharing, and showed us that in practice, legacy worked a little differently than what we expected.

The second are full simulations of the market under our model. We use 10 years of data from two different markets, and compare the performance of our method with the market index, and a representative of previous approaches to GA-based portfolio optimization.

The results of the second experiment shows us that our method achieves the objectives for which is was set. In other words, it is able to generate a good portfolio strategy over a period of time, while reducing the distance (cost) between portfolios.

In this chapter, we describe the details of the experiments made, and the results attained.

6.1 Data Used

We use publicly available real world historical data from stock markets. For each market, we pick its representative index and the assets composing that index. These representative indexes are determined by a capitalization weighted average of all the member assets, in other words, "market portfolios" by definition.

The data markets were chosen for being well known market indexes, and for containing a large number of composing assets. The large number of assets results in a more difficult and interesting problem for the portfolio optimization. As seen in the previous chapter, most works in this field use a much smaller set of assets for their portfolios.

The return values is taken from the monthly closing price of the index and composing assets. The closing value is the last value for a specific period. It is an arbitrary choice, which is often taken in other works.

6.1.1 NASDAQ 100 Index

NASDAQ was the world's first electronic stock market. Founded in 1971, in the United States, it has more than 3.000 companies. In number of companies, and shares traded daily, it is the biggest U.S. stock market.

The NASDAQ-100 is an size-weighted index composed of the 100 largest companies listed on the NASDAQ stock exchange. It includes companies from different countries, but does not includes financial companies.

For the dataset, we choose a subset of the NASDAQ-100, composed of 89 assets which have been part of the index continuously for the 6 year period from November 2000 to October 2006. This 72 month period composes our data.

A brief overview of the market behavior in this period can be seen in figure 6.1. It shows the value of the market index from 1998 to 2007. The main characteristic in this period is the "dot-com" stock bubble and crash, from late 1998s to early 2001. During this period, the stock quickly raised 3 times its value, and then fell back all at once.

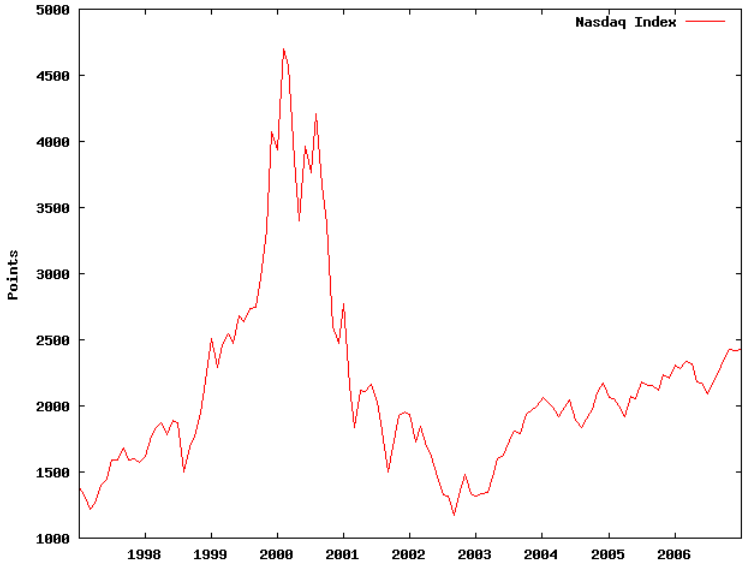

Figure 6.1: Performance of the NASDAQ composite Index.

6.1.2 NIKKEI 255 Index

The NIKKEI is the stock market index for the Tokyo Exchange. It has been calculated since 1950, and is the most watched index on the Asian market. It is composed of 255 assets from the Japanese economy.

Unlike the NASDAQ, which concentrates on technology companies, the NIKKEI index is designed to reflect the overall market. So, there is no weighting of particular industries. Instead, the composing assets are generally price weighted. Following this, the assets composing this index are much less correlated than those in the NASDAQ index.

In the dataset, 205 out of the total 225 companies are included. The choice was based on their continued presence in the index for the period from January 1998 to December 2006 (106 months).

This period characterizes the end of the Japanese bubble, and the corresponding bubble burst, which begins in 1999. By 2003, the Nikkei index reaches a record low, and then starts recovering. The behavior of the index can be seen in figure 6.2.

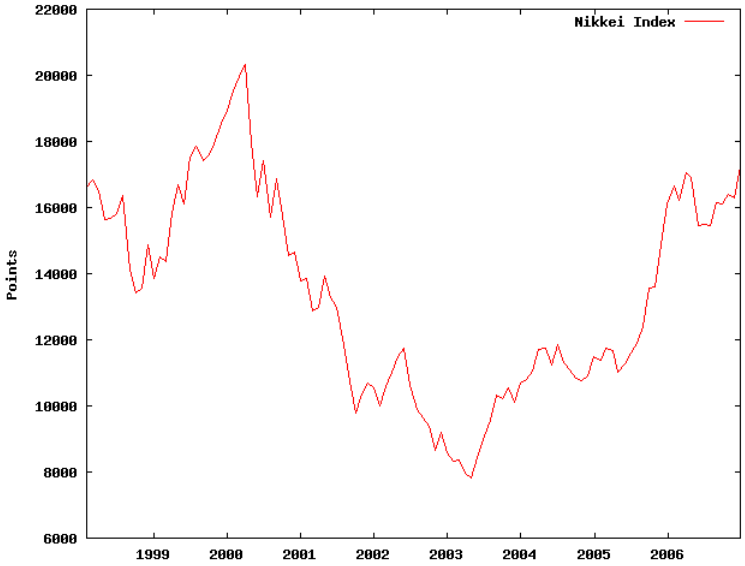

Figure 6.2: Performance of the NIKKEI Index.

6.2 Methodology

6.2.1 Parameter Value

We can separate the parameters used in our system in two types. Evolutionary Algorithm-related parameters, and Technique parameters. The first class describes parameters that are not unique to the technique we have developed, but are used frequently in Genetic Algorithm applications. The second class denotes the parameters that have been added to the methods described in this work. Table 6.1 divides the parameters in these two groups.

The values for the Specific Parameters have been decided based on sensibility experiments. We executed simple experiments, which will be discussed later in this chapter, with scenario-based optimization to understand the best values for these parameters.

The values for the Generic Parameters control the convergence speed and the influence of randomness in the result, by changing the selective pressure. They were determined by starting with conservative values and increasing them by hand

Table 6.1: Generic Parameters and Specific Parameters

Generic Parameters	Specific Parameters
Number of Generations	Objective Sharing Probability
Number of Individuals	Time Length
Elite Size	Legacy Size
Tournament K	
Crossover Ratio	
Mutation Ratio Flip	
Mutation Ratio Perturb	
Fill Ratio	

Table 6.2: Values for Generic Parameters

Parameter Name	NASDAQ dataset	NIKKEI dataset
Number of Generations	60	100
Number of Individuals	200	400
Elite Size	11	6
Tournament K	10	10
Crossover Ratio	0.8	0.8
Mutation Ratio Flip	0.05	0.05
Mutation Ratio Perturb	0.05	0.05
Fill Ratio	20%	20%

until an acceptable convergence rate was found.

The values used in the experiments reported here can be seen in table 6.2. The number of Generations and Number of Individuals are different for each dataset because of the difference in size between the data (NASDAQ has 100 assets, NIKKEI has 225).

6.2.2 Effect of randomness in the results

As a heuristic based random search, Genetic Algorithms relies on some non-deterministic steps. Nominally, the selection, mutation and crossover operators in this system all operate in a probabilistic fashion. The same thing is valid for

the Objective sharing method.

So, while a good choice and implementation of the the heuristic will use the randomness to improve the quality of the results, a poor implementation may "luck out" some times, and present very bad results in the average. Because of that, we added to the experiments an analysis of how much random chance affected the results.

Each of the following listed experiment was repeated 30 times, with different random number generator seeds. The results in the tables and figures in this chapter are thus all averages of the best results in each run, and numeric values are accompanied of the standard deviation of this average. So it's possible to see how much randomness affects the validity of the results.

6.2.3 Comparison and validation

To validate the results of our methodology, we have chosen two benchmarks to compare them against. The first one is the Market Portfolio, and the second one is a scenario-based Genetic Algorithm.

The Market Portfolio

One consequence from the Modern Portfolio Theory is that the optimal strategy for the portfolio investor would be to hold the *Market Portfolio* (i.e., the portfolio composed of all risky assets, proportional to their value. See section 2.1.3).

In the real world, this means the many Value-weighted Market indexes that we find. These indexes are composed by all (or the highest weighted) components of each market, and thus they approximate an efficient portfolio, according to the Modern Portfolio Theory.

Some critics say that using these indexes is not necessarily the optimal method, and may lead to trend-following decisions that result in a sub-optimal risk-return trade off. Some reasons for this are that the indexes do not contain all assets, and that the Modern Portfolio Theory does not hold completely.

However, in practice there are companies that hold funds based on market indexes, and people who trade them. So a direct comparison of the value of the

portfolio generated by our method and the value of the Market Portfolio is a useful tool to determine whether this method is able to "beat the market".

In this work, since the needed formulas for the calculations of the exact weights of the different companies into the Market Portfolio for each index are not available, the value of the Index will only be used for comparison of Risk and Returns. The Risk of the Market Portfolio is estimated as the standard deviation of its returns. For comparison of Trading Costs, only the other methods described in this section are considered.

Simple GA-Optimization

Besides comparing the results with the market portfolio, we also performed the experiments with a simple version of a GA Portfolio Optimization system without the techniques described in this text.

This competing technique, which we will call "simple GA", was designed to help us understand how the previous works would act in a multi-scenario situation. We suspected that since they were searching for the optimal portfolio independently of the previous or following scenarios, the distance between each optimized portfolio would be rather large.

The design of Simple GA follows the description of section 5.2. In the experiments the same parameter values as the proposed method are used. The values returned by this method are used in the comparison of the full length experiment.

6.3 Parameter Sensitivity Analysis

When designing new techniques, it is important to understand the effects of these techniques under many different conditions. If the techniques introduce new parameters, it is necessary to extensively experiment with the possible values of these parameters in order to understand how their changes affect the system.

The first experiments we describe in this text, then, are the sensitivity analysis for the parameters we have introduced. We perform experiments for P_{os}, *time_length* and *legacy_size* parameters. For each parameter, we ran a series of

experiments to compare the results for varied values.

The best values found in each experiment in this session were used for the "performance" experiments in next session. They were, specifically, 2 and 5 for P_{os}, 1 for *legacy_size*, 11 for *time_length* in the NASDAQ dataset and 6 for the NIKKEI dataset.

More detailed information on the methodology and results of the experiment follows.

6.3.1 Objective Sharing Parameter analysis

With Objective Sharing technique, we introduce the parameter p_{os} to the System. Its value determines the rate at which one of the two goals will be chosen for every generation.

Intuitively, we expect that a lower p_{os} value will result in a portfolio strategy with both smaller distances and returns, because of the increased importance of the distance fitness measure in the evolutionary process. As the value of this parameter is raised, we should find different degrees of compromise between the two measures.

Ideally, the investor should experiment with raising and lowering this value until an acceptable solution is found.

The actual results of experiment with this parameter can be seen on Table 6.3. The results reflect our intuitive expectations. The values in the table illustrate the results when applying different values for P_{os} on the NASDAQ dataset. Results for the NIKKEI dataset were similar.

When P_{os} is 2, we have the largest loss and the smallest distance as compared with the results of not using Objective Sharing (denoted as "none" in the table.

As we raise the value of P_{os}, we get smaller gains of Sharpe Ratio for larger losses in distance. We can see the trade-off between Sharpe Ratio and average Distance, as P_{os} increases.

From these results, we can see that changing from not using Objective Sharing to a value of 2 will result in a rather great increase of returns for a small increase in the distance measure. After that, increasing the value of the parameter p_{os} will

Table 6.3: Sharpe Ratio and Distance as a function of P_{os} parameter.

P_{os}	Sharpe Ratio	Distance
None	0.55	0.909
2	0.397	0.294
3	0.450	0.309
4	0.497	0.374
5	0.521	0.427
6	0.535	0.661
7	0.542	0.792

increase the distance for smaller increases in returns. This kind of information allows the analyst to tune the algorithm according to his particular cost structure, by balancing the two goals.

6.3.2 Time Length parameter analysis

The Time Length parameter indicates how many months prior to the current scenario will be taken into consideration in the moving average formula for calculating the expected return.

We tested a range from 3 to 30 months in both the NASDAQ and NIKKEI datasets, running the Simple GA algorithm to decide the best length for this parameter.

The results can be seen at figures 6.3, for the NASDAQ dataset, and 6.4 for the NIKKEI dataset. The values in the figures are the average total profit out of 30 runs of the system (with different random seeds).

In the NASDAQ dataset, we have two peaks, at the points of 11 and 18 months. In the NIKKEI dataset, we have a peak at 6 months, and then the results quickly lower. We have observed that the time length value is a problem-dependent parameter. From these results, we decided on the parameter values of 11 and 6 for the NASDAQ and NIKKEI datasets for the experiments in this work.

The proper value for the time_length parameter needs to be chosen carefully. There are local optima and the result's quality drops sharply for badly chosen

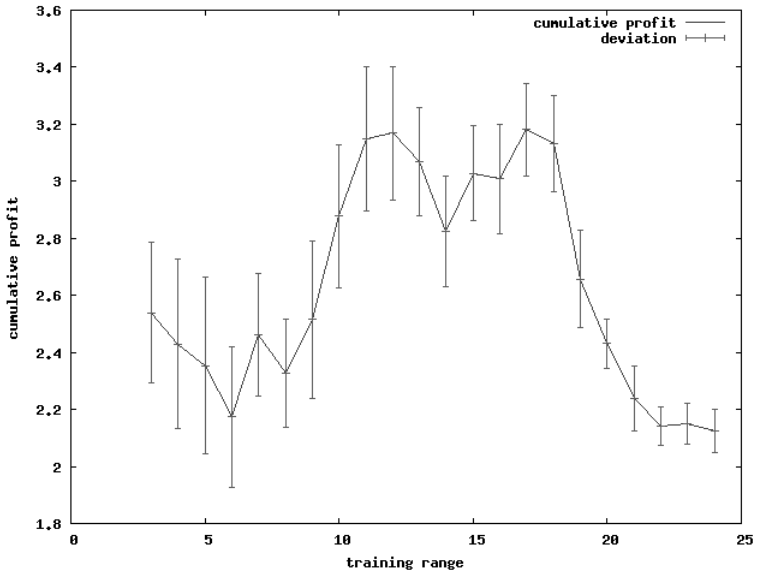

Figure 6.3: Total return as a function of time_length for NASDAQ data.

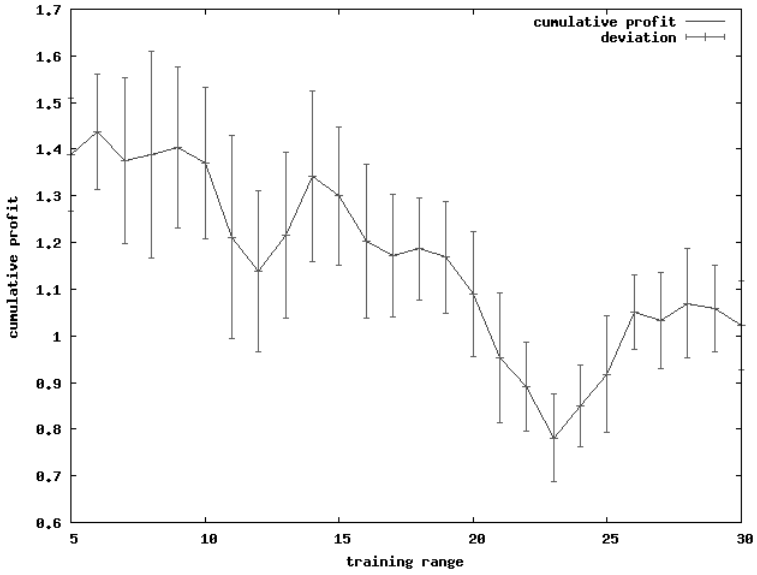

Figure 6.4: Total return as a function of time_length for NIKKEI data.

parameters. Based on our experiments, the method we found best for choosing the time_length parameter is to first find the best time length of a subset of the known data, and then use this value to manage the portfolio for a time period.

6.3.3 Legacy_Size parameter analysis

The Population Seeding technique introduces a new parameter to the Genetic Algorithm, the *Legacy_Size*, which determines how many individuals from the previous genetic search should be included in the new scenario.

As the number of assets available to a portfolio increases, the search space for the optimal portfolio also increases exponentially, making the problem harder. The Population Seeding technique is a heuristic which says that "the best *legacy_size* solutions of the last scenario are probably good enough to be used again". In other words, it supposes that the environment from one scenario to the next does not change that much. The first population in the new scenario will begin its search where the last population of the previous scenario was found to be successful.

To verify this hypothesis, we tested a Legacy Size from 0 to 30% of the population size for both NIKKEI and NASDAQ datasets, and the results can be seen in figures 6.5 and 6.6.

For the NASDAQ dataset, we can see that the lowest value for the Sharpe ratio happens when the legacy parameter is 0. The results rise sharply for small values of this parameter, then we see a small drop for values above 5.

For the NIKKEI dataset, we also observe the lowest Sharpe Ratio when the value of the legacy parameter is zero. The Sharpe Ratio drop for higher values however is smaller when compared to the NASDAQ dataset.

These results show us that our initial hypothesis isn't entirely correct. For both the NIKKEI data and the NASDAQ data, there is a significant difference between Legacy Size $= 0$ and Legacy Size > 0. The average cumulative profit and Sharpe ratio is much lower when Seeding is not used. However, when seeding is used, the effect of the legacy parameter value over the results quickly stabilizes.

These results suggest that Seeding has the following effect in the algorithm: the seeded individual will have a fitness value higher than that of the randomly

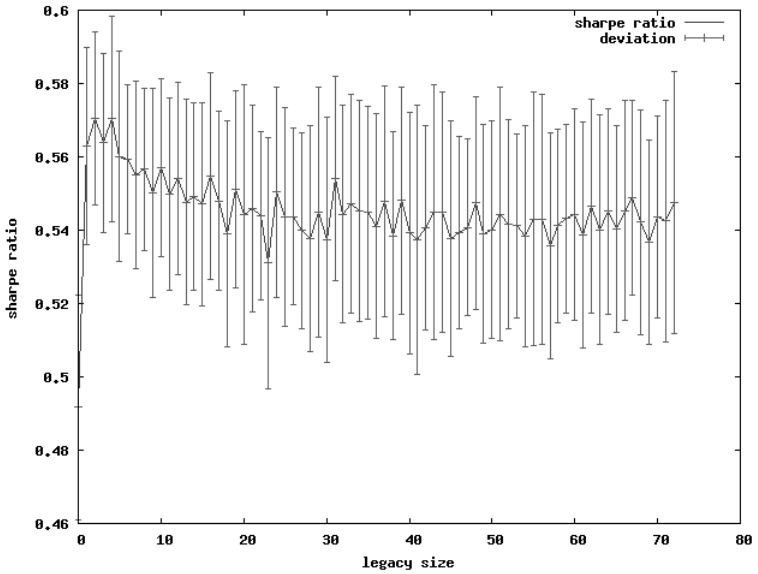

Figure 6.5: Effect of Legacy Size on the NASDAQ dataset.

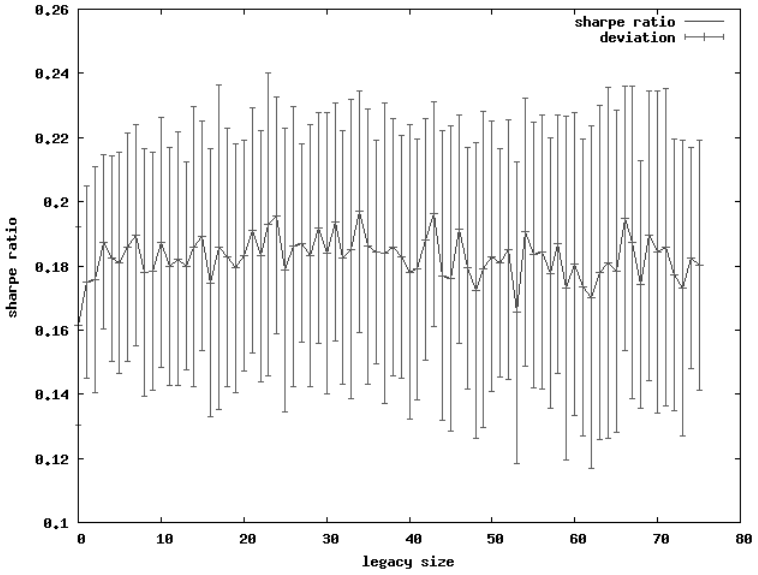

Figure 6.6: Effect of Legacy Size on the NIKKEI dataset.

generated individuals, and then soon dominate the population, directing it towards its region in search space. Since the seeded individual quickly dominates all the randomly generated ones, additional seeded individuals will not have as much of an effect as the first one.

6.4 Full Length Returns experiment

To validate and further investigate our methodology, we have simulated the results of the portfolio strategies generated by it. In this experiment, we let the genetic algorithm generate portfolio strategies for the whole 6-year period available in our dataset, and compared the obtained results with the market index and the standard GA Portfolio optimization technique.

We choose 5 different combinations of the specific parameters proposed in this research, decided according to the results described in the previous section.

Seeded GA means that we only use seeding, and not Objective Sharing. The *legacy_size* parameter is set to 1.

SOx GA adds the Objective Sharing technique, with parameter p_{os} equal to x. Seeding is not used. Like described in the previous section, the values of 2 and 5 were the smallest and highest values that showed a significant difference in results from other systems.

Seeded SOx GA includes both modifications to the system at the same time.

The results of the experiments are summed up on tables 6.4 and 6.5, for the NASDAQ and NIKKEI datasets, respectively.

"INDEX" is the value for the Financial Market Index corresponding to that dataset. "SO2" and "SO5" refer to the Objective Sharing heuristic with the value of the p_{os} parameter set to 2 and 5, respectively. "Seeded" refers to the Population Seeding heuristic.

Cumulative return, and Sharpe ratio are measures of the "efficiency" of the portfolio strategy. They should be compared with the values given for the index. Cumulative return is how much the portfolio's value increased with relation to its initial value. Sharpe Ratio indicates the risk - for similar cumulative returns, a

Table 6.4: Results of Full Length Experiment - NASDAQ Dataset

Dataset	NASDAQ		
Method	Sharpe Ratio	Distance	Cum. Return
Index	0.239	n/a	1.721
Simple GA	0.486 (0.0006)	1.477 (0.004)	2.82 (0.02)
Seeded GA	0.55 (0.0009)	0.909 (0.012)	3.14 (0.01)
SO2 GA	0.425 (0.0011)	0.429 (0.002)	2.85 (0.023)
Seeded SO2 GA	0.397 (0.002)	0.294 (0.003)	2.59 (0.01)
SO5 GA	0.492 (0.0007)	0.818 (0.0009)	2.89 (0.001)
Seeded SO5 GA	0.521 (0.0008)	0.42 (0.0039)	3.12 (0.001)

Table 6.5: Results of Full Length Experiment - NIKKEI Dataset

Dataset	NIKKEI		
Method	Sharpe Ratio	Distance	Cum. Return
Index	0.216	n/a	1.562
Simple GA	0.168 (0.001)	8.51 (0.5)	1.439 (0.02)
Seeded GA	0.181 (0.001)	8.21 (0.49)	1.49 (0.01)
SO2 GA	0.154 (0.0017)	1.32 (0.06)	1.376 (0.002)
Seeded SO2 GA	0.125 (0.002)	1.19 (0.07)	1.28 (0.015)
SO5 GA	0.174 (0.0009)	2.55 (0.09)	1.446 (0.006)
Seeded SO5 GA	0.156 (0.0015)	2.48 (0.11)	1.38 (0.01)

higher Sharpe Ratio indicates a smaller amount of risk.

Distance is the average distance between portfolio positions through the strategy. It is a linear measure of the cost associated with that strategy. Higher value means that the portfolio position changes more during the period, and thus there will be a higher associated cost. The final cost will depend on the volume of trade, and policies of the trading company.

Comparing the GA methods, we notice that their performances relative with one another were consistent with our expectations in both datasets. However, their performance relative to the Index were quite different in each dataset. We'll first examine the independent performance, and then address the index performance.

Seeded GA dominated Simple GA in both datasets, with higher Sharpe Ratio and Cumulative Returns, and lower average distance. This shows that while we introduced the seeding technique to reduce distance, it has also managed to bias the search towards a better region of the search space.

The results for Objective Sharing were as expected. SO2 had a smaller Sharpe Ratio when compared with Simple GA, in exchange for a much smaller Average Distance. Interestingly, the Cumulative Return in the NASDAQ dataset remained near the same, which means that sometimes the SO algorithm may trade only risk for distance, instead of return.

When we combined the two methods, mixed results were obtained. In the NASDAQ dataset, Seeded SOx dominated the SOx methods, while the opposite happened in the NIKKEI dataset.

We suppose that the difference between the two behaviors can be explained because of the crash behavior of the NIKKEI assets. With the sudden changes on the direction of the market, information from previous successful portfolios might cause more harm than good. More study is needed to fully understand this phenomenon.

Figures 6.7, 6.8, 6.9 and 6.10 give us a better image of metrics in tables 6.4 and 6.5. We compare the methods with best overall results with simple GA and the index values.

Figures 6.7 and 6.8 show the results for the NASDAQ dataset. The NASDAQ

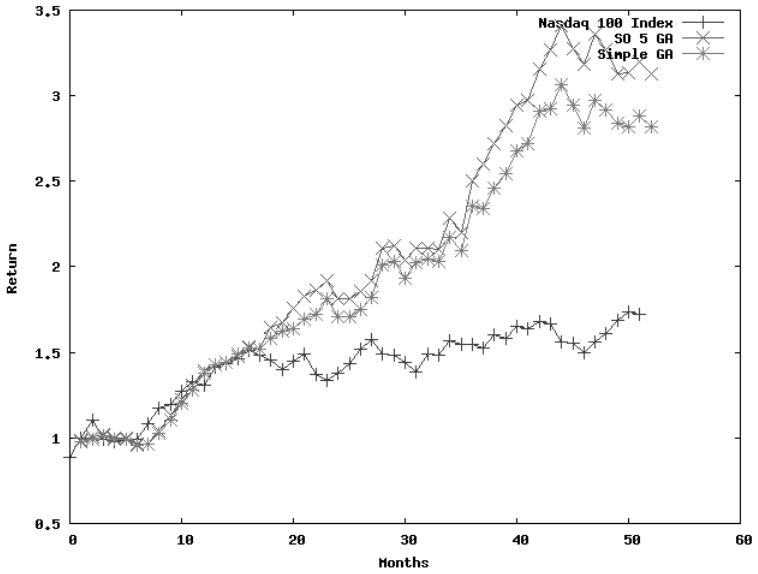

Figure 6.7: Cumulative returns for the NASDAQ dataset

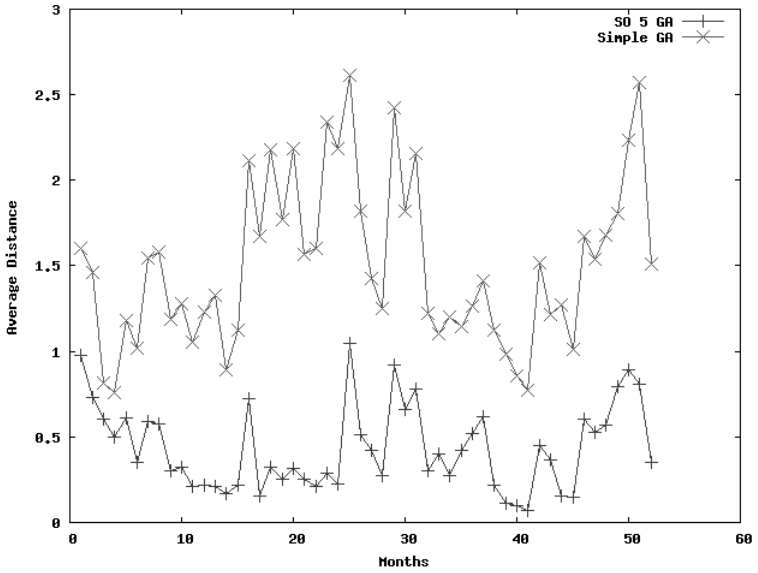

Figure 6.8: Distance values for the NASDAQ dataset

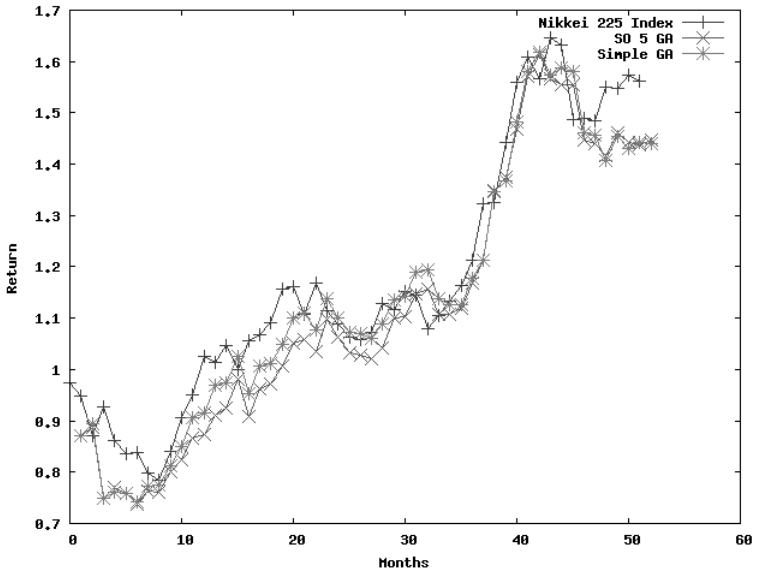

Figure 6.9: Cumulative returns for the NIKKEI dataset

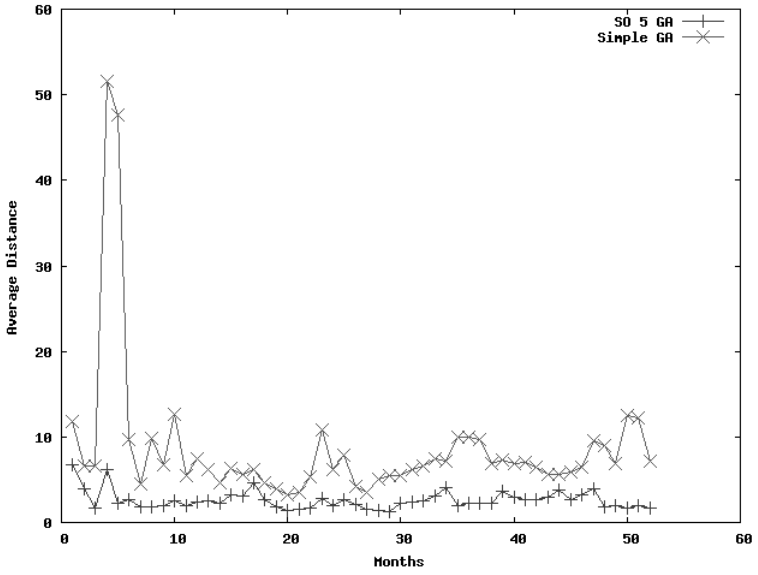

Figure 6.10: Distance values for the NIKKEI dataset

index was stable during most of the period, which was soon after the dot-com bubble. The market showed an overall raising trend.

The first image shows the cumulative return for the GA methods and the Index. The Evolutionary techniques are able to pick good assets and build portfolios with larger returns, while maintaining similar risk ratios as the Market Index.

The second figure shows the difference between the distances in the simple GA method, and the method we propose. While the peaks are in similar places, our method shows consistently lower portfolio distances between the scenarios.

Figures 6.9 and 6.10 show the results for the NIKKEI dataset. The Nikkei index was going through a crash behavior during the period. The market was quickly losing value, until it reached a critical low for a while, then quickly regained its normal value.

Like for the NASDAQ images, the first image shows the cumulative return for the GA methods and the Index. Here, however, the GA methods are not able to consistently beat the index, instead showing a similar/worse performance. In the sharp drop in month 2 both methods make a wrong decision, and have to play catch up with the index. Looking at the second figure, we see that Simple GA show extreme changes during the first drop, while SO5 manages to recover using a much simpler change in the portfolio.

6.5 Experiment Result Analysis

The results we obtained in the experiments described in this chapter are very encouraging. Genetic Algorithm based Portfolio Optimization managed to beat the market index consistently.

In the NASDAQ dataset, where the best quantitative results were obtained, we see that while the normal GA method beats the index by a certain margin, the margin for the improved method is much larger. However, in the NIKKEI Dataset, where the GA methods can't do more than follow the index, we find that the normal and improved GA methods show similar returns. This indicates to us that the Nikkei dataset is particularly difficult for GA methods in general,

because in all other tests the Improved GA beat the non-improved method by a large margin.

Another analysis that we can perform between the normal and improved GA methods is by looking at the distance figures. In figures 6.10 and 6.8 we can notice a characteristic of this reduced distance. The portfolios management strategy shows much larger distances when the market changes its direction, in comparison to the changes that happen when only the intensity of the direction changes.

In other words, when the market starts falling after rising, or rising after falling, we observe the largest changes in the evolved portfolios, while even if the rate of falling or rising changes dramatically, the change in the portfolio will be much less. We can attribute this behavior to the adaptive characteristics of evolutionary algorithms.

Finally, as a last observation, we have observed no significant change in processing time for any of the combinations during the experiment. As can be seen from the heuristics descriptions, neither of the two are computationally intensive. So the genetic algorithm system afford a higher degree of complexity for a better result.

Chapter 7

Conclusion

This work discusses the problem of Portfolio Management. The Portfolio Management Problem consists of optimizing a distribution of financial resources over many available assets (like bonds or stocks). The goals of this distribution is twofold: One, to maximize the return realized by the portfolio, and two, to reduce the individual risk of the composing assets.

We explained the basic portfolio theory, as described by Markowitz. According to this theory, as the number of assets in the portfolio rises, the contribution of each asset towards the portfolio risk diminishes. Following this rationale, the most efficient portfolio is the Market Portfolio, which contains all the available assets, weighted by their capitalization. When real-life constraints are added, the problem of calculating this efficient portfolio becomes intractable by numerical methods.

We review the use of Genetic Algorithms, a heuristic for random search, to solve this problem. This has been a popular approach in recent years, with a large number of works published in the topic. From this review we can understand that a common point among many current works is the lack of cost measuring. Many works in GA-based portfolio optimization are *single-scenario*, which means that they do not take into account what happens before or after the time window of the optimization. Also, costs are not taken into account when choosing the portfolio weights. Since stock trading can be seen as a positive-sum game before costs are added, these results that do not take cost into account will be unrealisticly good.

To address this problem, two techniques are described to draw a connection between the portfolio being optimized with the position in previous scenarios. We call these techniques Seeding and Objective Sharing. We execute a series of

simulated experiments with a common GA modified by these techniques, and we conclude that adding a cost model as a objective function to the GA allows for lower differences among the portfolios in different scenarios (lower costs) with the same return.

Having validated our technique by means of experiments, we hold that the use of a distance metric as fitness measure, in addition to a possible direct cost measure, is essential to allow the genetic algorithm to correctly and efficiently take cost into account. In order to effectively realize portfolio management over time, some sort of implementation of a distance metric similar to what we propose in this dissertation is needed.

The results of this work were presented in two published papers to international conferences (JCIS07, and CEC2007), and the feedback provided was very encouraging. We believe that this work will be seen as a reference for approaching cost modeling as an evolutionary objective.

7.1 Future Directions

Our results are encouraging, and suggests that Evolutionary Computation is an approach that deserves further attention for the Portfolio Management problem.

7.1.1 Cost Model

Regarding the distance measure, and its use as a fitness function, in this work we developed the Euclidean Distance and realized that this function, although succeeds in reducing the average distance to the management solution, also adds some restrictions to the model that do not correspond to real-world applications. So a very important follow up to this work is the research of more appropriate distance functions. One possible example would be the use of a modified Manhattan distance function.

Also regarding the distance measure, in this work we use only the fitness as and abstract concept of distance. This has earned us good quantitative results, since we are able to measuring the diminishing distances, but the results we have achieved

are not directly comparable with traditional methods. In order to effect this important comparison, the addition of traditional cost models to this evolutionary cost model is required.

7.1.2 Representation

One well known way to improve the efficiency of a Evolutionary Algorithm is to change the representation. A good representation will result in a smoother search space, which increases convergence and reduces the risk of falling into local optima.

In this work we used Real Valued Array as the genomic representation of a portfolio. This is the representation which is traditionally used in similar works.

We intend now to test a new tree-like representation. This tree representation would be similar to the Genetic Programming tree, where the leafs represent weight values for assets, and the nodes are connectors.

Due to the subtree-is-a-tree property of trees, we can use this representation to combine two successful portfolio in a way that is not possible in the current representation. We can also give fitness values for subtrees, and use these sub-fitness values to guide the crossover process.

Due to these characteristics, we believe that a tree representation is an interesting follow up in the Portfolio Optimization research.

Bibliography

Aranha, C., Kasai, O., Uchide, O., and Iba, H. (2007). Day-trading rules development by genetic programming. In *Proceedings of the Joint Conference on Information Sciences (JCIS)*, Salt Lake City.

Aydemir, D., Aranha, C., and Iba, H. (2006). Evolutionary behavior acquisition for humanoid robots. In *Proceedings of the 24th Annual Conferrence of the robotics society of Japan*, Okayama, Japan.

Azzini, A. and Tettamanzi, A. G. (2006). A neural evolutionary approach to financial modeling. In *GECCO 2006 - Genetic and Evolutionary Computation Conference*, pages 1605–1612, Seattle, Washington. ACM Press.

Chen, S.-P., Li, C., Li, S.-H., and Wu, X.-W. (2002). Portfolio optimization model with transaction costs. *Acta Mathematicae Applicatae Sinica*, **18**(2), 231–248.

Coello Coello, C. A. (2001). A Short Tutorial on Evolutionary Multiobjective Optimization. In E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 21–40. Springer-Verlag. Lecture Notes in Computer Science No. 1993.

Das, I. and Dennis, J. (1997). A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, **14**(1), 63–69.

Fyfe, C., Marney, J. P., and Tarbet, H. (2005). Risk adjusted returns from technical trading: a genetic programming approach. *Applied Financial Economics*, **15**, 1073–1077.

Hochreiter, R. (2007). An evolutionary computation approach to scenario-based risk-return portfolio optimization for general risk measures. In M. G. et al.,

editor, *EvoWorkshops 2007*, number 4448 in LNCS, pages 199–207. Springer-Verlag.

Horn, J. (1997). Multicriterion decision making. In D. F. Thomas Back and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, volume 1, pages F1.9:1–F1.9:15. IOP Publishing Ltd. and Oxford Press.

J.H.Holland (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

Jian, R. and K.Y.Szeto (2002). Discovering investment strategies in portfolio management: A genetic algorithm approach. In *Proceedings of the 9th International Conference on Neural Information Processing*, volume 3, pages 1206–1210.

Jiang, R. and Szeto, K. Y. (2003). Extraction of investment strategies based on moving averages: A genetic algorithm approach. In *Conference on Computational Intelligence for Financial Engineering (CIFEr2003)*, pages 403–410, Hong Kong. IEEE.

Jong, K. A. D. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. Ph.D. thesis, The University of Michigan.

Kwon, Y.-K. and Moon, B.-R. (2004). Evolutionary ensemble for stock prediction. In *GECCO 2004 - Genetic and Evolutionary Computation Conference*, pages 1102–1113. ACM Press.

Kwon, Y.-K., Choi, S.-S., and Moon, B.-R. (2004). Stock prediction based on financial correlation. In *GECCO 2004 - Genetic and Evolutionary Computation Conference*, pages 1102–1113. ACM Press.

Lin, C.-M. and Gen, M. (2007). An effective decision-based genetic algorithm approach to multiobjective portfolio optimization problem. *Applied Mathematical Sciences*, **1**(5), 201–210.

Lin, D., Li, X., and Li, M. (2005). A genetic algorithm for solving portfolio optimization problems with transaction costs and minimum transacton lots. *LNCS*, (3612), 808–811.

Lipinski, P., Winczura, K., and Wojcik, J. (2007). Building risk-optimal portfolio using evolutionary strategies. In M. G. et al., editor, *EvoWorkshops 2007*, number 4448 in LNCS, pages 208–217. Springer-Verlag.

Markowitz, H. (1952). Portfolio selection. *Journal of Finance*, **7**, 77–91.

Markowitz, H. (1987). *Mean-Variance analysis in Portfolio Choice and Capital Market*. Basil Blackwell, New York.

Nikolaev, N. Y. and Iba, H. (2001). Regularization approach to inductive genetic programming. *IEEE Transactions on evolutionary computation*, **5**(4), 359–375.

ping Chen, S., Li, C., hong Li, S., and wei Wu, X. (2002). Portfolio optimization with transaction costs. *Acta Mathematicae Applicatae Sinica*, **18**(2), 231–248.

Rosin, C. D. and Belew, R. K. (1997). New methods for competitive co-evolution. *Evolutionary Computation*, **5**(1), 1–29.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100.

Schwefel, H. P. (1981). *Numerical Optimization of Computer Models*. Chichester: John Wiley
& Sons.

Streichert, F., Ulmer, H., and Zell, A. (2003). Evolutionary algorithms and the cardinality constrained portfolio optimization problem. In D. Ahr, R. Fahrion, M. Oswald, and G. Reinelt, editors, *Operations Research Proceedings*. Springer.

Subramanian, H., Ramamoorthy, S., Stone, P., and Kuipers, B. J. (2006). Designing safe, profitable automated stock trading agents using evolutionary algorithms. In *GECCO 2006 - Genetic and Evolutionary Computation Conference*, pages 1777–1784, Seattle, Washington. ACM Press.

Uludag, G., Uyar, A. S., Senel, K., and Dag, H. (2007). Comparison of evolutionary techniques for value-at-risk calculation. In M. G. et al., editor, *EvoWorkshops 2007*, number 4448 in LNCS, pages 218–227. Springer-Verlag.

Vose, M. (1999). *The Simple Genetic Algorithm*. The MIT Press.

Werner, J. C. and Fogarti, T. C. (2002). Genetic control applied to asset managements. In J. F. et Al., editor, *EuroGP*, LNCS, pages 192–201.

Yan, W. and Clack, C. D. (2006). Behavioural gp diversity for dynamic environments: an application in hedge fund investment. In *GECCO 2006 - Genetic and Evolutionary Computation Conference*, pages 1817–1824, Seattle, Washington. ACM Press.

Yan, W. and Clack, C. D. (2007). Evolving robust gp solutions for hedge fund stock selection in emerging markets. In *GECCO 2007 - Genetic and Evolutionary Computation Conference*, London, England. ACM Press.

Yao, X. (1999). *Evolutionary Computation Theory and Applications*, chapter 1. World Scientific.

Yuh-Dauh-Lyu (2002). *Financial Engineering and Computation*. Cambridge Press.

Publication List

1. Claus Aranha and Hitoshi Iba (2006). Using Genetic Algorithms to Improve Ant Colony Clustering In *Symposium of the Japanese Society of Information Processing on Problem Solving by Mathematical Modelling*, pages 85–92, Nagoya, Japan.

2. Deniz Aydemir, Claus Aranha and Hitoshi Iba (2006). Evolutionary Behavior Acquisition for Humanoid Robots In *The 24th Annual Conferrence of the robotics society of Japan*, Okayama, Japan.

3. Claus Aranha and Hitoshi Iba (2006). The Effect of Using Evolutionary Algorithms on Ant Clustering Techniques In *Proceedings of The Third Asian-Pacific Workshop on Genetic Programming (ASPGP'06)*, pages 24–34, Hanoi, Vietnam.

4. Claus Aranha and Hitoshi Iba (2007) Modeling Cost into a Genetic Algorithm-based Portfolio Optimization System by Seeding and Objective Sharing In *Proceedings of Conference on Evolutionary Computing (CEC)*, Singapore, accepted.

5. Claus Aranha and Hitoshi Iba (2007) Portfolio Management by Genetic Algorithms with Error Modeling In *Proceedings of the Joint Conference on Information Sciences (JCIS)*, Salt Lake City, EUA

6. Claus Aranha, O. Kasai, O. Uchide and Hitoshi Iba (2007) Day-Trading Rules development by Genetic Programming In *Proceedings of the Joint Conference on Information Sciences (JCIS)*, Salt Lake City, EUA