MULTI-OBJECTIVE OPTIMISATION WITH FINANCIAL APPLICATIONS

NOEL-ANN BRADSHAW

A thesis submitted in partial fulfilment of the
requirements of the University of Greenwich
for the Degree of Master of Philosophy

AUGUST 2015

DECLARATION

I certify that this work has not been accepted in substance for any degree, and is not concurrently being submitted for any degree other than that of Master of Philosophy being studied at the University of Greenwich. I also declare that this work is the result of my own investigations except where otherwise identified by references and that I have not plagiarised the work of others.

ACKNOWLEDGMENTS

I would like to thank my supervisor Dr Chris Walshaw for his encouragement, guidance and support during the undertaking of this research. I would also like to thank my other supervisors Dr Cos Ieretheou and Professor Kevin Parrott for their input, help and support.

ABSTRACT

Portfolio Optimisation is a multi-objective problem which involves finding the allocation of shares in a portfolio that optimises the likely return for a level of risk which an investor is prepared to tolerate. There have been several multi-objective evolutionary algorithms that have been used to solve the portfolio optimisation problem in recent years.

This thesis recounts the development of a new multi-objective evolutionary algorithm, Adaptive Cell Resolution Evolutionary Algorithm (ACREA), which has been shown to perform well in portfolio optimisation. ACREA uses a novel grid-based dynamic reduction mechanism which allows the solution population to grow and reduce whilst maintaining diversity across the whole solution space. The algorithm is tested on data from the OR Library and uses the difference in area between the analytic solution and found solution to measure the new algorithm's success.

Before this algorithm was developed other heuristics were written in order to understand the basics of coding evolutionary algorithms in Java. After covering background information on financial applications and optimisation techniques, this thesis goes on to describe a simulated annealing algorithm to find the parameters for GARCH (1,1) and an evolutionary algorithm to determine trading rules using various trading indicators. These algorithms are described briefly in order to give an account of the journey and development.

CONTENTS

TABLES

FIGURES

1. Introduction

The academic field of financial mathematics covers a wide range of topics which can broadly be described as mathematical techniques and procedures that can be applied in the financial world. Examples of these applications include asset pricing, volatility forecasting, hedging and risk management, and portfolio management. This thesis is mainly concerned with portfolio management and more specifically optimisation but it does touch on other aspects of financial mathematics such as estimating volatility and technical analysis.

Over the last ten to twenty years there has been a proliferation of financial mathematics courses ranging from those at undergraduate level to postgraduate courses either for those already active in the financial sector and postgraduate courses or for those with a numerate background who want to obtain the technical skills to enable them to get into the financial world. There has also been an increase in financial mathematics conferences and streams at conferences. For example the Institute of Mathematics and its Applications (IMA) ran a conference in computational finance in 2007 followed by a conference on the mathematics of finance in 2013 and 2015, The British Applied Mathematics Colloquium/British Mathematics Colloquium (BAMC/BMC) had a financial mathematics stream in 2010 and numerous other financial mathematics conferences have sprung up in the few years since this research was started.

There could be many reasons for this increased interest in the world of econometrics. Maybe this an acknowledgement of the need for more understanding of the mathematics behind the financial markets or quite simply universities have been able to attract financial mathematicians who previously worked for investment companies and banks and are therefore able to address the growing demand for training for the next generation of financial experts.

At the time this research began few people would have predicted that financial mathematics would become as high-profile as it has since the financial crash. There has been controversy over the role of mathematics in finance (Hand, 2009; Steinsaltz, 2009). This is picked up further by Harris in his discussion concerning the context of this debate (2015, pp. 79-108). Nevertheless the application of advanced mathematics to the financial world is now largely established as part of the UK mathematics curriculum.

Chapter two provides an introduction to some of the theory behind key measurements in financial mathematics namely volatility, risk and asset pricing and then goes on to explain the mathematics of portfolio management and the basics of optimisation. Chapter three examines various heuristics that can be employed in complex optimisation situations where the analytical solution is either too complex or takes too long to compute. It starts with optimisation for single-objective problems but then progresses to look at multi-objective optimisation. This chapter culminates with an overview of various evolutionary algorithms that are currently used in portfolio optimisation and provides the background theory for the algorithm that has been produced by this research.

Chapter four begins with the description of a new simulated annealing algorithm for finding GARCH(1,1) parameters and an evolutionary algorithm for determining trading rules based on various technical indicators. After this it goes on to introduce the main focus of this research which is a new multi-objective evolutionary algorithm (MOEA) that has been developed for portfolio optimisation. It details the differences between this and the best known MOEAs and looks at some of the initial results. Chapter five outlines the possibilities for further research in this area and presents the conclusions.

2. Background

This chapter discusses some of the background terminology used in the dissertation. It begins with the idea of using share price returns to analyse performance and examines various ways that the volatility of these returns can be measured including Historic, Exponentially Weighted Moving Average (EWMA), Autoregressive Regressive Conditional Heteroskedacity (ARCH) and Generalised Autoregressive Regressive Conditional Heteroskedacity (GARCH). GARCH is referred to again in chapter 4 when a simulated annealing program for calculating the parameters is discussed.

The next section describes the Efficient Market Hypothesis and briefly discusses some of the issues that financial practitioners and academics disagree over. This is followed by a section on Trading Indicators. These trading indicators: Moving Averages, Price Channel Breakouts, Relative Strength Index, Moving Average Convergence Divergence and Oscillators are referred to again in Chapter 4 when an evolutionary algorithm for determining these is discussed.

This chapter goes on to look at risk; discussing both value at risk and conditional value at risk. It concludes with a section on portfolio management which focuses on the Modern Portfolio Theory as recounted by Markowitz (1952). The theory of Portfolio Optimisation is returned to later in Chapter 4 is conjunction with the description of a new multi-objective evolutionary algorithm for portfolio optimisation: Adaptive Cell Resolution Evolutionary Algorithm (ACREA).

2.1. Returns

In order to analyse share price performance it is necessary to look at returns. The return of a share can be described as 'the percentage growth in the value of an asset' (Wilmott, 2001). This is a much better way of describing the changing value of an asset rather than just looking at the share prices as it takes into account which direction and by how much the price is moving.

The simple or arithmetic return (*u*) from day (*i*-1) to day (*i*) for price (*S*) is calculated as:

$$u_i = \frac{S_i}{S_{i-1}}$$

This gives a very similar result to

$$u_i = \ln\left(\frac{S_i}{S_{i-1}}\right)$$

These ‘log-returns’ are often preferred (Maringer, 2005) as they are very similar in behaviour to simple returns and yet the continuously compounded returns are time additive and more easily manipulated in mathematical calculations.

Brookes and Tsolacos (2010) explain this by using the following example. Suppose the weekly return is required and daily log returns have been calculated. These are represented by $u_2 \ldots . u_6$ below. The weekly return can be computed by summing the daily returns for the week.

$$u_2 = \ln\left(\frac{S_2}{S_1}\right) = lnS_2 - lnS_1$$
$$u_3 = \ln\left(\frac{S_3}{S_2}\right) = lnS_3 - lnS_2$$
$$u_4 = \ln\left(\frac{S_4}{S_3}\right) = lnS_4 - lnS_3$$
$$u_5 = \ln\left(\frac{S_5}{S_4}\right) = lnS_5 - lnS_4$$
$$u_6 = \ln\left(\frac{S_6}{S_5}\right) = lnS_6 - lnS_5$$

Thus the return over the week is:

$$\sum_{i=1}^{6} u_i = lnS_6 - lnS_1 = ln\left(\frac{S_6}{S_1}\right)$$

However this property is not so useful in portfolio management where returns are aggregated across different assets rather than within one asset across time. Consequently simple or arithmetic returns are used when evaluating a portfolio’s periodic return and log returns are used when evaluating price behaviour over time (Meucci, 2010). Lists of market prices tend not to quote returns but rather periodic (often daily) prices.

2.2. Volatility

‘The volatility σ of a stock is a measure of uncertainty about the returns ... and can be defined as the standard deviation of the return provided by the stock in one year.’ (Hull, 2006). Hull goes on to say that stocks typically have a volatility measure of between 15% and 60%. A stock with a low volatility might be considered a safe investment but there is as little chance of a good return as a high loss. Conversely stocks with high volatility are considered risky when in fact there is as much chance of making a profit as a large loss. There is limited value in calculating past volatility; what is more important and valuable is having some sort of estimation of future volatility. A number of methods for forecasting volatility have been proposed and those used in this report are outlined below.

2.2.1. Historic

Historic volatility is often used as the benchmark; this is the measure to which other volatility estimates are compared. It can be worked out using daily, hourly, weekly etc prices. Hull (2006) gives the following method of calculation:

Defining:

n : Number of observations
S_i: Stock price at the end of the i^{th} interval with $i = 0, 1, \ldots, n$
u_i : Log return
$\bar{u}_i$: The mean logreturn

So

$$u_i = \log\left(\frac{S_i}{S_{i-1}}\right)$$

for $i = 1, 2, \ldots, n$ Then the usual measure of historic volatility or standard deviation (σ) of the returns (u_i) is given by:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(u_i - \bar{u})^2}$$

Or

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}{u_i}^2 - \frac{1}{n(n-1)}\left(\sum_{i=1}^{n}u_i\right)^2}$$

2.2.2. Exponentially Weighted Moving Average

In 1994 JP Morgan published the first edition of Riskmetrics™, a technical document which proposed using the Exponentially Weighted Moving Average (EWMA) for forecasting volatility (JP Morgan, 1996).

$$\sigma = \sqrt{(1-\lambda)\sum_{i=1}^{n}\lambda^{i-1}(u_i - \bar{u})^2},$$

where the parameter λ (0 < λ <1) is referred to as the decay factor. This parameter determines the relative weights that are applied to the returns and thus how much they are used in determining volatility. JP Morgan (1996) state two main advantages for this measure. Firstly it gives more weight to most recent changes in the market but secondly, after any large jump in price, the volatility will decline exponentially as the weight associated with the jump in price falls. Whereas, the use of a simple moving average would lead to relatively abrupt changes in the standard deviation once the volatility, caused by the jump in price, reduces.

Riskmetrics™ outlines many ways of determining the best value for λ (the decay factor) (JP Morgan, 1996). Their results show that for daily volatility it is best to use $\lambda = 0.94$ and for monthly $\lambda = 0.97$.

2.2.3. Autoregressive Conditional Heteroskedasticity (ARCH)

Moving average models of volatility such as EMWA assume that returns are independent and identically distributed (Alexander, 2001). However this is often not the case and at high frequencies returns may show signs of autocorrelation. In the late 1970's and early 1980's Engle and others came up with the ARCH model for forecasting volatility. This is documented in (Engle, 1982) and many other publications. ARCH takes into account volatility clustering and assumes that today's conditional variance is a weighted average of past returns (Alexander, 2001). The formula for ARCH(*n*) over *n* days using the *m* most recent observations is given as:

$$\sigma_n^2 = \alpha_0 + \sum_{i=1}^{m} \alpha_i u_{n-i}^2$$

According to Alexander (2001), ARCH models are not often used in financial markets as GARCH models outperform them.

2.2.4. Generalized Autoregressive Conditional Heteroskedasticity (GARCH)

In 1986 Tim Bollerslev adapted the ARCH model to GARCH (1,1) the simplest of the ARCH/GARCH models. This is the only GARCH model that is used in this report. Here returns are assumed to be generated by a stochastic process with time-varying volatility (Alexander, 2001). Bollerslev (1986) describes the model as allowing for a much more flexible lag structure than ARCH. The "(1,1)" in GARCH(1,1) indicates that σ_n^2 is based on the most recent observations of $u^2$and the most recent estimate of the variance rate. The more general GARCH(p,q) model calculates σ_n^2 from the most recent p observations on u^2 and the most recent *q* estimates of the variance rate.

A general equation for a weighted average for variance over *n* days using the most recent *m* observations could be expressed as:

$$\sigma_n^2 = \sum_{i=1}^{m} \alpha_i u_{n-i}^2$$

where u_i is the return on day i and α_i represents the amount of weight given to observation i days ago. The α_i's are positive. If they are chosen such that $\alpha_i < \alpha_j$ when $i > j$, then less weight is given to older observations. The weights must sum to one (Hull, 2006).

GARCH extends this by assuming that there is a long-running average variance rate and that this should be given some weighting (Hull, 2006). This leads to:

$$\sigma_n^2 = \gamma V_L + \sum_{i=1}^{m} \alpha_i u_{n-i}^2$$

where V_L is the long-run variance rate and γ the assigned weight. Because the weights sum to one we have:

$$\gamma + \sum_{i=1}^{m} \alpha_i = 1$$

Defining $\omega = \gamma V_L$ we obtain

$$\sigma_n^2 = \omega + \sum_{i=1}^{m} \alpha_i u_{n-i}^2$$

Eventually we get to the equation for GARCH(1,1)

$$\sigma_n^2 = \omega + \alpha u_{n-1}^2 + \beta \sigma_{n-1}^2$$

where ω is a constant, α is the GARCH error coefficient and β the GARCH lag coefficient.

The value of the estimated parameters determines the short-run movement of a stock's volatility. Large β coefficients suggest that very high or low volatility will take a long time to die out whereas large α coefficients indicate that the volatility of a particular stock will react intensely to market movement (Alexander, 2001). So stocks which have a high value of α and a low value of β will show a spiky volatility time-series (Alexander, 2001). Chapter 4 describes an evolutionary algorithm for determining weights ω, α, β, for a set of given returns.

2.3. Risk

There is always risk when investing in stocks and shares. What is the risk of losing money? The study of risk is complicated and much has been written on this. Risk is touched on here in order to mention the two measures of risk described below.

2.3.1. Value at Risk

Value at Risk (VaR) answers the question ‘how bad can things get?’ (Hull, 2006). In 1996 the Basel Committee on Bank Supervision concluded that banks should calculate their market risk using VaR. A degree of confidence in the measure is set and a period of time agreed. For example it might be calculated that over the next month there is a 95% probability that a portfolio of assets will lose no more than £5m (Wilmott, 2001). This would be written as:

$$\text{Prob}\{\delta V \leq -\text{£5m}\} = 0.05$$

Where δV is the change in the portfolio’s value. Generally we have:

$$\text{Prob}\{\delta V \leq -\text{VaR}\} = 1 - c$$

where c is the degree of confidence, 95% in the above example.

Wilmott goes on to say that VaR is calculated using the current prices of all assets in the portfolio, their volatilities and the correlations between them. Hull (2006) shows this as being $z\sigma_P\sqrt{n}$ where n is the number of days in the time period, σ_P the standard deviation of the change in the portfolio and z the number of standard deviations away from the mean that a normally distributed variable will decrease by for a given probability. This standard deviation of the portfolio can be calculated as follows:

$$\sigma_P^2 = \sum_{i=1}^{n} \alpha_i^2\, \sigma_i^2 + 2\sum_{i=1}^{n}\sum_{j<i} \rho_{ij}\, \alpha_i\alpha_j\sigma_i\sigma_j$$

Where α_i is the amount invested in asset i $(1 \leq i \leq n)$ and ρ_{ij} is the coefficient of correlation of the returns between assets i and j.

2.3.2. Conditional Value at Risk

Conditional Value at Risk (CVaR) answers the question ‘If things do get bad how much can one expect to lose?’ (Hull, 2006). Thus this is considered to be a better measure of risk but, until recently, was not so well used. Alexander (2001) argues that VaR is not a coherent risk measure as it does not necessarily satisfy the following inequality VaR $(X+Y) \leq \text{VaR}(X)+\text{VaR}(Y)$ where X and Y are the losses incurred by two different assets. To overcome this Artzner, et al. (1999) introduced Conditional Value at Risk as a new measure of risk and describe it as the expected loss given that the loss has exceeded the VaR threshold. That is:

$$\text{Conditional VaR} = E(X|X > \text{VaR})$$

Both VaR and CVaR were originally used as the measures of risk in the prototype of the new algorithm ACREA for Portfolio Optimisation proposed in chapter 4. However in order to be able to make comparisons between this algorithm and others in the literature VaR and CVaR are no longer used.

2.4. Efficient Market Hypothesis

This is the assumption that states that share prices are random and that successive price changes are independent of each other. It is still the subject of much research, discussion and disagreement between leading academics and practitioners with academics promoting the hypothesis and practitioners dismissing it (Adams, 2003).

An efficient stock market is one where the share prices fully reflect all available information such as the economic prospects of the companies concerned, stock splits, dividend increases, and merger announcements etc. If a market is efficient, this implies that it is not possible to use technical analysis of past prices to predict future patterns in the behavior of stock prices. Burton Malkiel (author of *A Random Walk Down Wall Street*) discusses this at some length in *Is the Stock Market Efficient?* (1989) concluding that ‘Pricing irregularities may well exist and even persist for periods of time, and markets can at times be influenced by fads and fashions. Eventually, however, any excesses in market valuations will be corrected.’ However he goes on to say that as technology advances we may see further departures from efficiency and be able to understand their causes more fully. This is echoed in his survey of the literature in The Efficient Market Hypothesis and Its Critics (Malkiel, 2003).

2.5. Trading Indicators

If you search online for ‘Trading Indicator’ numerous websites come up all claiming to offer great financial rewards for those who are prepared to study patterns in the distribution of stock prices. The use of trading indicators (also called technical analysis) is a way of predicting future price movements based only on observing the past history of prices (Wilmott, 2001). The idea is that by looking for patterns in past prices traders can decide whether they should buy, sell or hold their shares for a particular asset. Depending on which indicators seem to work for a particular asset trading rules can be determined.

According to Lo, et al. (2000) one of the greatest gulfs between academic finance and industry practice is the separation that exists between technical analysts and their academic critics. The idea of being able to predict what a share in a company might do by looking at these indicators is an anathema to the academics who believe fully in the Efficient Market Hypotheses. Indeed, Malkiel (1996) argues that "under scientific scrutiny, chart-reading must share a pedestal with alchemy". Lo, et al. (2000) attempt to bridge the gap between technical analysis and quantitative finance by developing a systematic and scientific approach to the practice to gauge the efficiency of technical indicators over time and across securities.

There are mixed results for those using trading indicators to determine a set of rules which investors can use to determine when to buy and sell assets. Ready (2002) compares the work of Allen & Karjalainen (1999) and Brock, et al., (1992) who have used different methods to determine trading rules with mixed results. Following on from Allen & Karjalainen (1999) an evolutionary algorithm for finding these rules is discussed in chapter 5. The indicators used in this program are discussed below.

2.5.1. Moving Averages

This is a technical indicator that is used to measure the momentum or trend of a particular stock (Anon., 2015). The moving average of closing prices over *n* days is calculated and compared with the current price. If the current price is higher, then a 'buy' indicator is returned whereas, if lower, a 'sell' indicator is returned. Moving averages over different periods of time can be compared and the point where they cross can signify a change in the underlying trend and indicate a time to buy or sell (Wilmott, 2001).

Both Allen & Karjalainen (1999) and Brock, et al., (1992) use moving averages as indicators extensively. Murphy (2015) says that while some technical indicators are more popular than others, few have proved to be as objective, reliable and useful as the moving average.

Figure 1 shows an example of a moving average where *n*=10. Point A shows the price rising above the average and thus triggering a 'buy' signal whereas point B shows the price falling below the average and thus triggering a 'sell' signal.

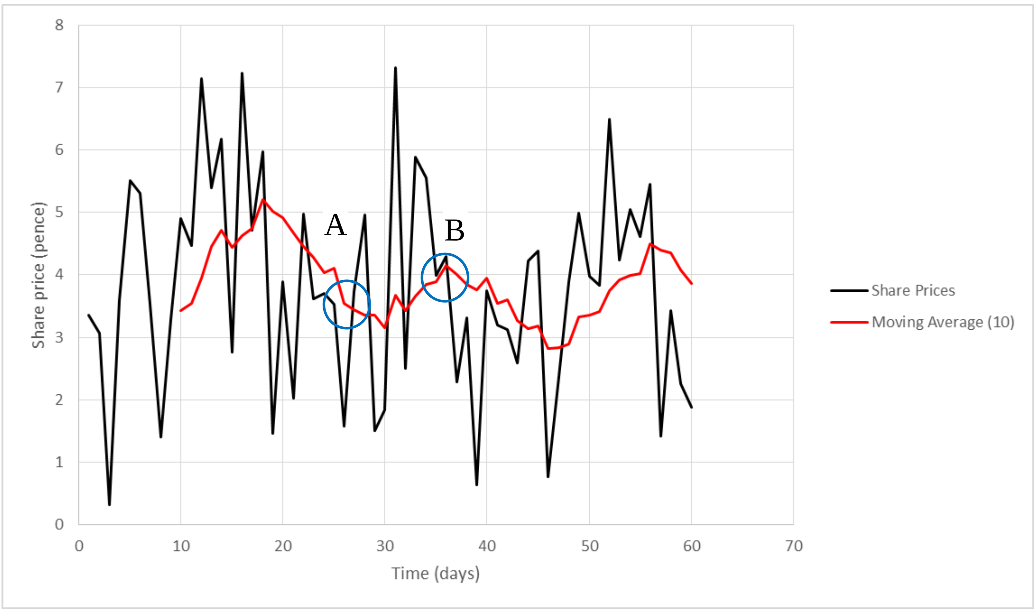


Figure 1: Diagram showing how a moving average can be used to create buy and sell signals

2.5.2. Price Channel Breakouts

A maximum and minimum value for the channel are set. These are usually taken as the highest high and lowest low over the last period time intervals and according to Krivo (2012) are often referred to as Donchian channels after Richard Donchian (Donchian, 1957). The stock price is plotted on the channel and if it exceeds the maximum then the indicator becomes 'sell' whereas if it goes below the minimum then 'buy' is returned. This is shown in Figure 2 where point A shows a 'sell' signal and point B a 'buy' signal.

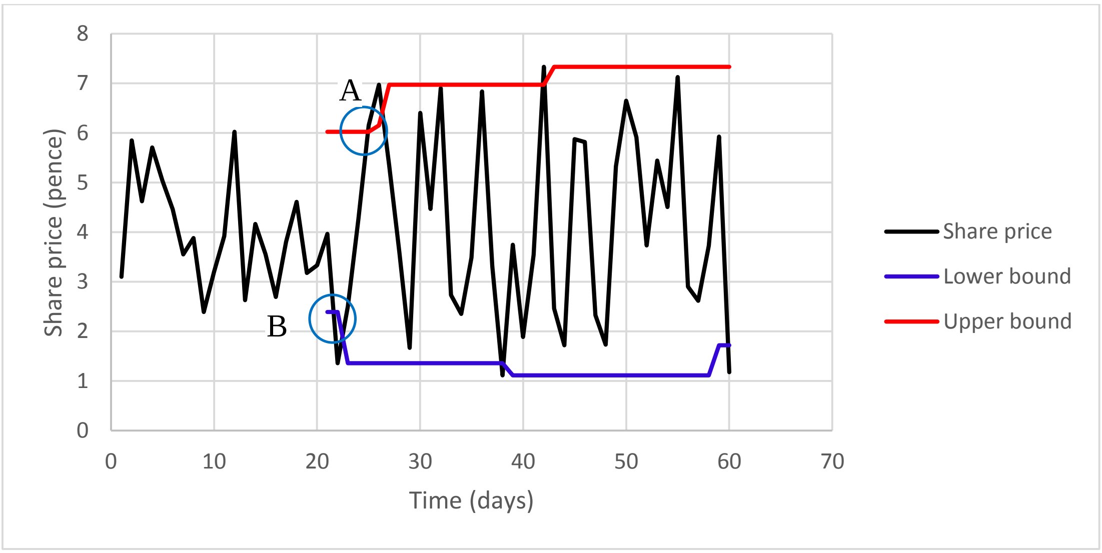


Figure 2: An example of a price channel breakout

2.5.3. Relative Strength Index

Wilmot (2001) describes this as the percentage of up moves in the last *n* days. A number higher than 70% is said to be overbought and likely to fall so returns a 'sell' indicator, whereas a number below 30% is said to be oversold and thus rise and so returns a 'buy' indicator (Wilmott, 2001). It can be calculated using the following formula $\text{RSI} = 100 - \frac{100}{(1-RS)}$ where *RS* is the average of *x* days' up closes divided by the average of x days' down closes.

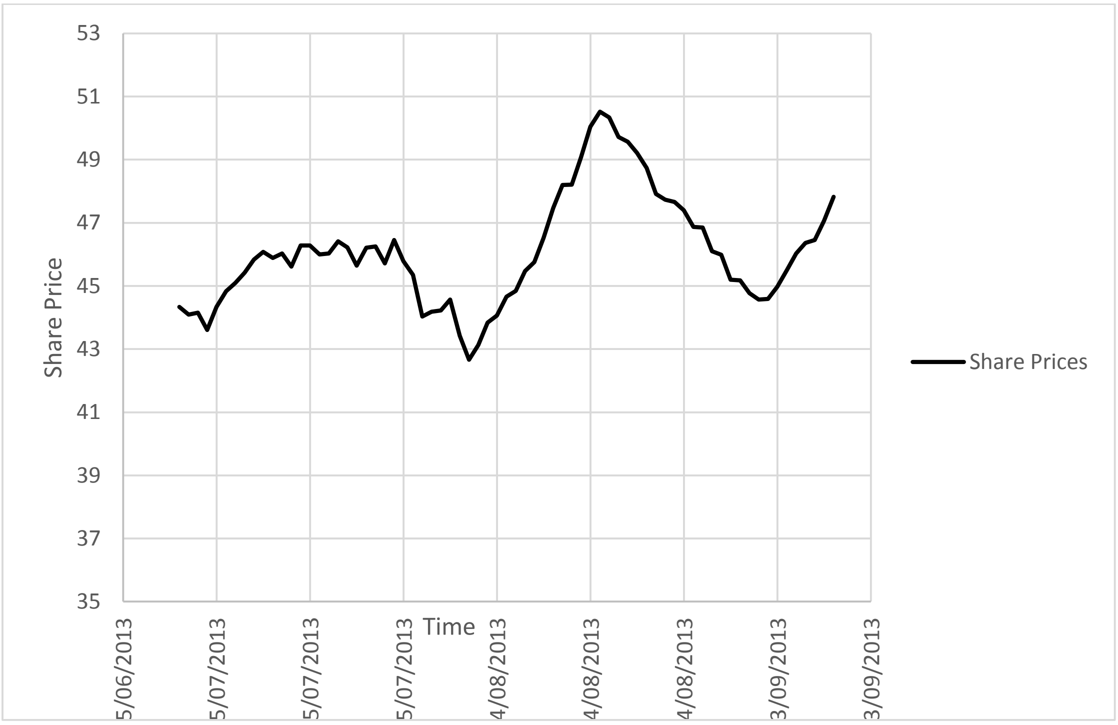


Figure 3: Share prices used to calculate RSI

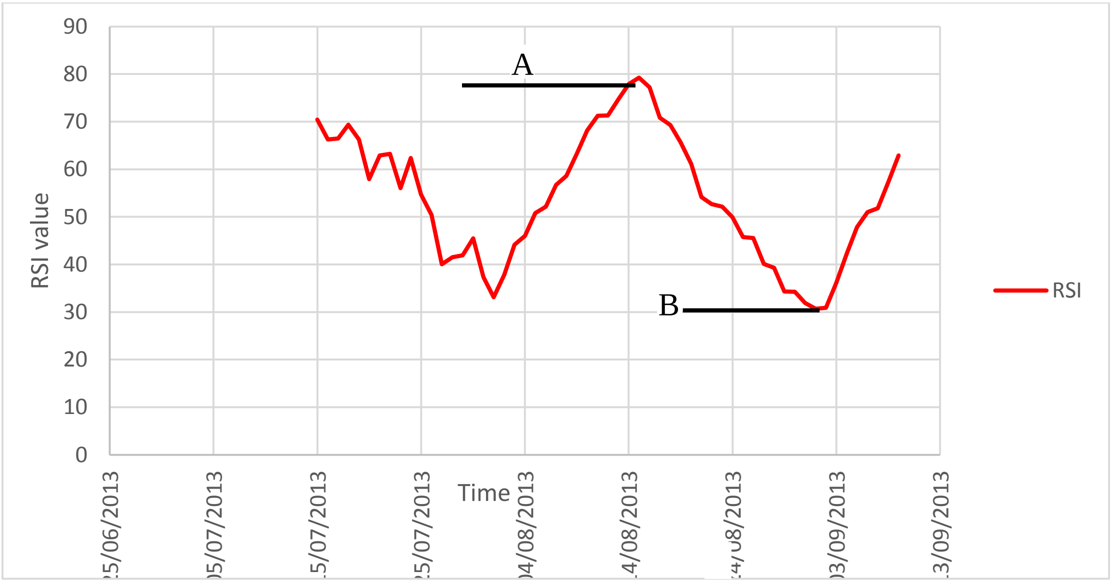


Figure 4: RSI calculated from share prices in Figure 3

Figure 3 shows sample share prices with the corresponding RSI underneath it in Figure 4. In Figure 4 point A shows where the share is said to have been overbought and likely to fall and so returns a ‘sell’ indicator, whereas point B shows where the share is said to have been oversold and thus rise and so returns a ‘buy’ indicator.

2.5.4. Moving Average Convergence Divergence (MACD)

This indicator looks at the difference between two moving averages known as the MACD Line. If this then crosses a third moving average (the signal line) then a buy/sell flag is triggered (Anon., 2015). It is usual to calculate the MACD Line as the 12-day moving average minus the 26-day moving average. A 9-day moving average of the MACD Line is plotted as a signal line. The values of 12, 26 and 9 are the typical setting used with the MACD, however other values can be substituted depending on your trading style and goals.

The shorter moving average is faster and responsible for most MACD movements whereas the longer moving average is slower and less reactive to share price changes. The MACD Line oscillates above and below the zero line, which is also known as the centerline. These crossovers signal that the shorter moving average has crossed the longer moving average (Stockcharts, 2015).

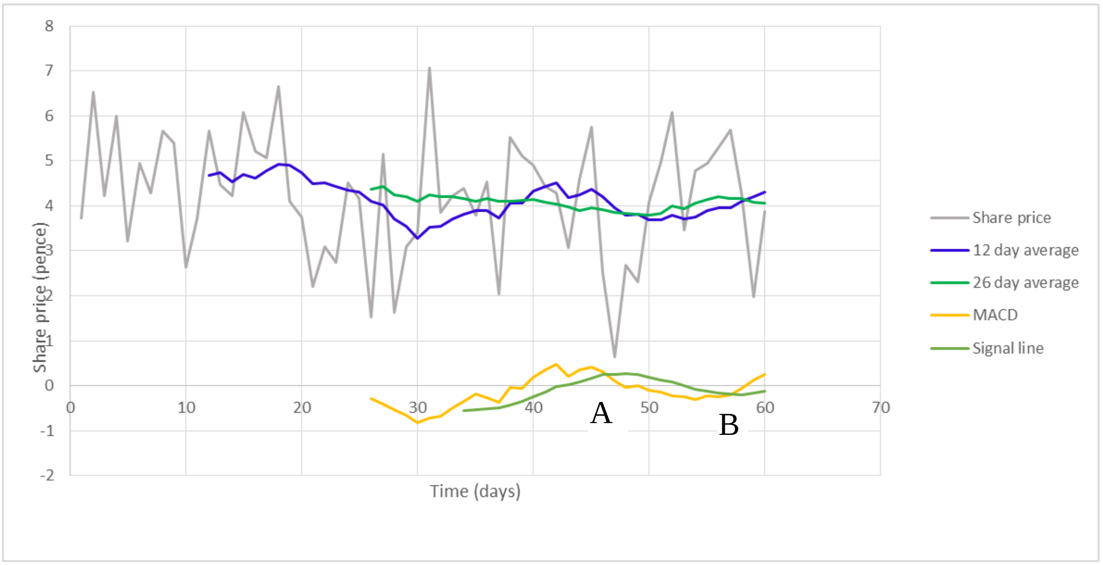


Figure 5: MACD with corresponding share prices and moving averages

In Figure 5 point A shows the MACD falling below the signal line indicating that it may be time to sell whereas point B shows the MACD rising above the signal line indicating that it may be time to buy.

2.5.5. Oscillators

An oscillator looks at the trend behind stock prices and will return a buy or sell flag accordingly. The Stochastic Oscillator compares a closing price to its price range over a given period. By adjusting *n* the oscillator's sensitivity to market movement can be reduced or increased. Two values, *SK* and *SD* are calculated.

$$SK = 100\left(\frac{C-L}{H-L}\right)$$

Where *C* is the recent closing price, *L* is the low of the previous n days and *H* is the high of the previous n days. *SD* is a 3-period moving average of *SK*.

According to Dempster & Jones (2001) there are many rules for trading using stochastic oscillators, some of which are highly subjective and cannot be easily automated. However a commonly used rule is to buy when $SK > SD$ and sell when $SK < SD$.

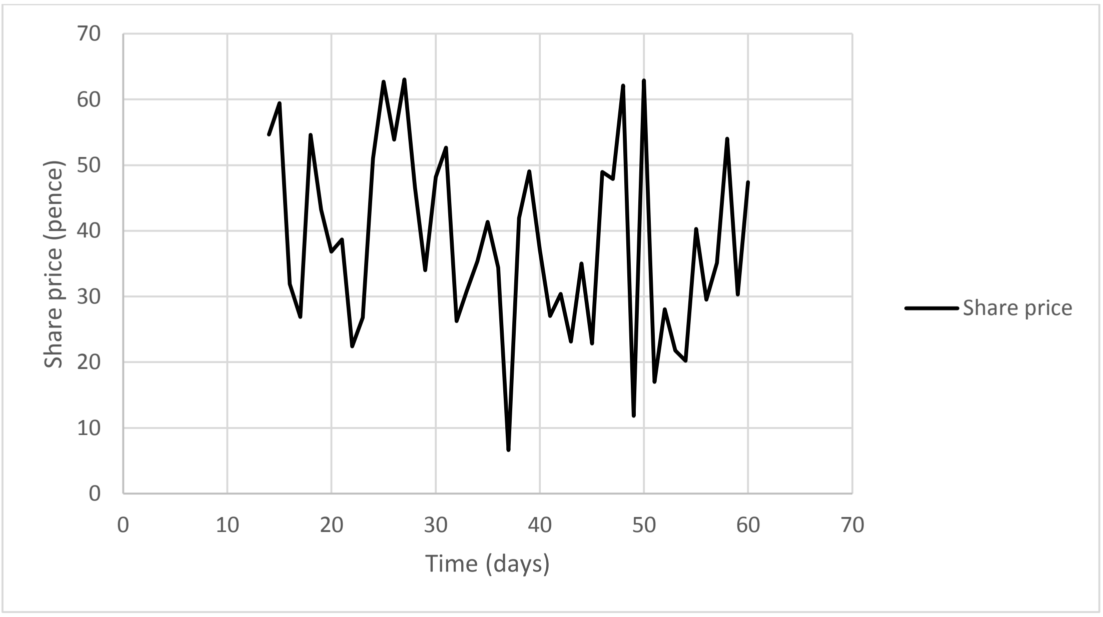


Figure 6: Share prices used to calculate stochastic oscillator

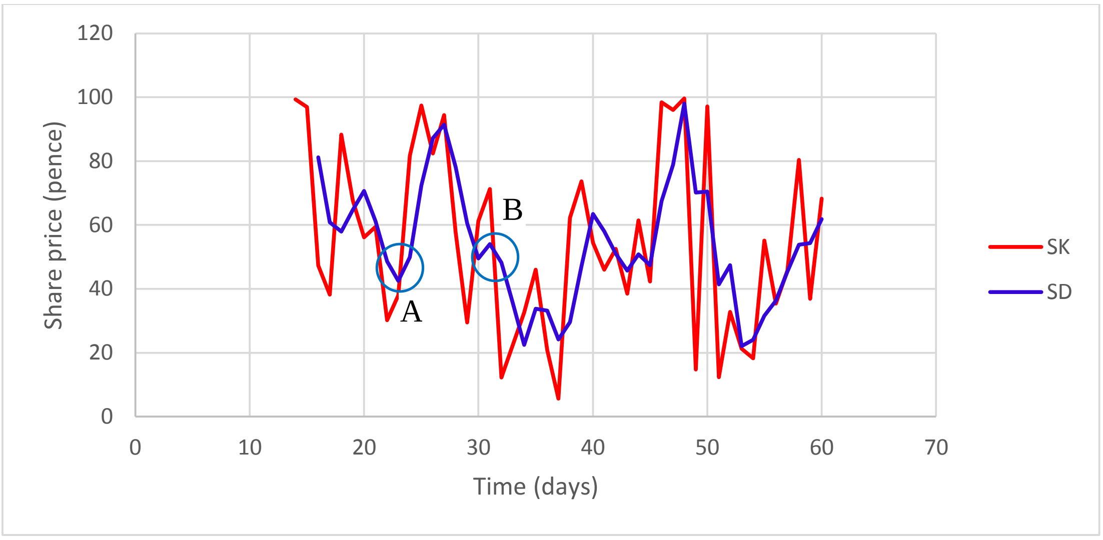


Figure 7: SD and SK calculated from share prices in Figure 6

In Figure 7 point A is where the SK rises above the SD which is where a 'buy' signal is generated and point B is where the SD line is above the SK line and then a 'sell' signal is generated. Figure 6 shows the corresponding shareprices; it can be seen from this that point A occurs just as share prices start to rise so a good time to buy and point B when share prices are beginning to fall so a good time to sell.

2.6. Portfolio Management

A portfolio is a collection of assets. In general for most people these are likely to include real assets such as house, car, electrical appliances as well as financial assets (Elton, et al., 2011). Like Elton, this thesis, is confined to financial assets.

Modern Portfolio Theory is based on Harry Markowitz's work, some of which was published by the Journal of Finance in 1952 (Markowitz, 1952). According to Markowitz it is possible to construct an 'efficient frontier' of optimal portfolios. These are portfolios that have the highest possible return for a given measure of risk. If you are willing to tolerate a particular level of risk you want to maximise the amount of return obtained. Thus you want to be able to identify the optimum portfolios for particular risk measures and this set of portfolios are said to be on the efficient frontier.

Nowadays portfolios of assets can include bonds, shares, stocks, treasury bills and all sorts of other financial investments. This chapter will look at some of the issues and mathematics behind portfolio management.

2.6.1. Risk-Return trade off

Everyone who wants to invest is trying to make the most money that they can whilst taking into account how much risk they are prepared to incur. For some this will mean a safe investment where the risk is minimal but so is the return, whilst others are content to experience significant losses if they can also expect significant return at other points. There is no such thing as a perfect investment; for every amount of return there will be a degree of risk and it is up to the investor to say what level of risk they are prepared to accept (Elton, et al., 2011).

2.6.2. Portfolio Construction and Diversification

In order to spread the risk out, it is important to have a portfolio that is not all concentrated in one area or sector. If one has investments that are all linked to, say, the construction industry and then the economy slows to the extent that all new builds stop, this portfolio is going to lose a considerable amount of money. However if a portfolio is carefully constructed so that different industries, with little overlap, are represented then the portfolio can weather the losses in one industry and survive times of economic downfall because they are compensated elsewhere.

Nowadays, it is becoming more common to have portfolios that are internationally diversified so that investments span different currencies and operate in different countries with diverse economic climates (Bekaert & Hodrick, 2011).

2.6.3. Portfolio Optimisation

A key question in portfolio optimisation is how much should you invest in any particular asset so that, across the whole portfolio, you maximise your return within your accepted measure of risk?

The usual way of describing this is to suppose that there are X risky assets, whose rates of return are denoted by random variables $u_1, \cdots, u_X$. Recall that

$$u_i = \frac{S_i}{S_{i-1}}$$

Let $w = (w_1 \cdots w_X)^T$, where w_i denotes the proportion of the portfolio invested in asset i, with

$$\sum_{i=1}^{X} w_i = 1$$

Thus the rate of return for the portfolio is

$$u_p = \sum_{i=1}^{X} w_i \, u_i$$

Traditionally academics have taken the standard deviation or variance of the returns of assets in a portfolio as the measure of risk associated with a portfolio. The variance and covariance of individual assets are characterized by a covariance matrix

$$V = \begin{bmatrix} \sigma_{11} & \cdots & \sigma_{1X} \\ \vdots & \ddots & \vdots \\ \sigma_{X1} & \cdots & \sigma_{XX} \end{bmatrix}$$

where σ_{ii} is the variance of asset *i* and σ_{ij} is the covariance of asset *i* and asset *j*. The portfolio variance is given by

$$\sigma_p^2 = w^T V w = \sum_{i=1}^{X} \sum_{j=1}^{X} w_i \, w_j \sigma_{i,j}$$

However this has been criticised by many practitioners who prefer to use VaR or CVaR (Adams, 2003; Maringer, 2005).

Markowitz (1952) proposed a mean-variance portfolio theory which formulated the problem as:

$$\text{minimise} \sum_{i=1}^{X} \sum_{j=1}^{X} w_i \, w_j \sigma_{i,j}$$

$$\text{subject to}$$

$$u_p = \sum_{i=1}^{X} w_i \, u_i$$

$$\sum_{i=1}^{X} w_i = 1$$

$$0 \leq w_i \leq 1 \quad i = 1, \ldots\ldots, X$$

This is a static problem in that the problem parameters stay unchanged during the planning period.

2.7. Summary

This chapter has provided the reader with the background information on the aspects of financial mathematics necessary for the understanding of the remainder of this thesis and in particular the financial theory behind the new algorithms developed in Chapter 4.

It began by looking at how share price returns can be used to analyse performance and discussed several different measures of volatility: Historic, Exponentially Weighted Moving Average (EWMA), Autoregressive Regressive Conditional Heteroskedacity (ARCH) and Generalised Autoregressive Regressive Conditional Heteroskedacity

(GARCH). This information will be referred to again in Chapter 4 where a simulated annealing program for calculating the parameters for GARCH (1,1) is discussed.

The Efficient Market Hypothesis was briefly discussed and some of the issues that financial practitioners and academics disagree over were highlighted. This was followed by a section on various trading indicators including: Moving Averages, Price Channel Breakouts, Relative Strength Index, Moving Average Convergence Divergence and Oscillators. It was necessary to include a description of these as background information for the Evolutionary Algorithm for determining these which is discussed in Chapter 4.

The concept of risk, and in particular value at risk and conditional value at risk, is discussed which is followed by a section on the theory of portfolio optimisation where risk is an important factor. This is returned to later in Chapter 4 in conjunction with the description of the new multi-objective evolutionary algorithm for portfolio optimisation: Adaptive Cell Resolution Evolutionary Algorithm (ACREA) which is the main thrust of this research.

3. Optimisation

The previous chapter provided the background financial mathematics necessary for the understanding of the new work contained in this thesis whereas this chapter contains the background information on optimisation that is necessary to understand the material in Chapter 4. The chapter starts with the basics of finding the global minimum and maximum using calculus and then various numerical iterative processes such as gradient search before going on to look at two heuristics; simulated annealing and evolutionary algorithms. These can be used when the function is complex and simpler methods tend to result in the finding of local minima and maxima. The terminology and parameters used in evolutionary algorithms are discussed in detail in order to prepare the reader to understand the different algorithms discussed later in the chapter.

Whilst new versions of these two algorithms have been developed as part of this research, the main thrust has been to create a multi-objective evolutionary algorithm (MOEA) and a general introduction to MOEAs is in the next section in this chapter. After a discussion as to why MOEAs are needed there then follows a description of some of the main competing algorithms for the new algorithm in Chapter 4. These are: The Elitist Non-Dominated Sorting Genetic Algorithm (NSGA and NSGAII), the Pareto Archived Evolution Strategy (PAES), the Strength Pareto Evolutionary Algorithm (SPEA and SPEAII) and the Pareto Envelope-Based Selection Algorithm (PESA and PESAII). These algorithms are initially described generally for a variety of purposes.

The next section looks at various research that has used these MOEAs to solve the Portfolio Optimisation problem and how it is possible to measure how well an algorithm has performed. An understanding of this is important in order to see how well the new MOEA described in chapter 4 performs.

3.1. Introduction

Optimisation is concerned with finding values for one or several decision variables that best meet the objective(s) without violating any constraint(s) (Maringer, 2005). Obviously the definition of the word ‘best’ is key in this explanation. Deb (2001) says that the need for optimisation arises from the desire to find a solution for real-life situations such as the minimum cost of production or the maximum reliability of a particular material.

There are several analytical solutions for optimising certain mathematical functions in order to find the global maximum or the global minimum. The simplest method of finding the maximum and minimum point of a function where there are no constraints requires finding the root of the first derivative,

$$f'(x) = 0$$

followed by the second derivative to determine if the point is a maximum, minimum or saddle. However this only works for a continuous, differentiable function.

For complex functions with more than one variable or with several turning points finding the global maximum or minimum is much harder and requires the use of computerised algorithms. Common gradient-based search algorithms include Newton's method and the method of steepest decent (Yang, 2008). These are usually local search methods which start with a 'guess' in the feasible region and then follow an iterative process to improve the solution. Such algorithms will often find local maxima and minima (e.g. A and B) and not necessarily the global maximum (e.g. C) or global minimum point (e.g. D). To combat this, heuristic optimisation techniques are often used (Yang, 2008).

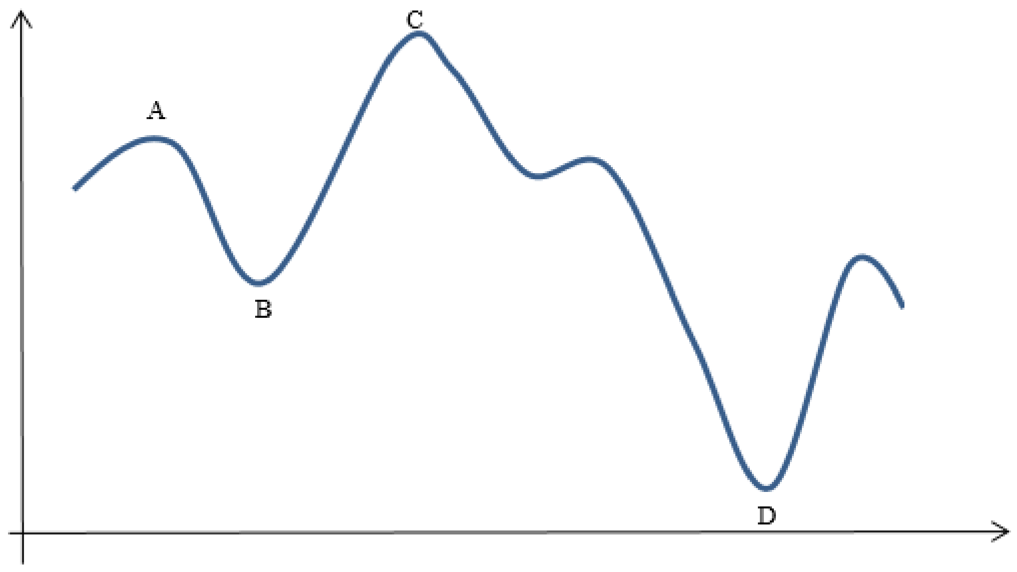


Figure 8: Diagram to show maxima and minima of a function

There are many definitions of the word heuristic but in the context of optimisation Barr and Feigenbaum's (1986) definition seems particularly apt:

'A heuristic is a rule of thumb, strategy, trick, simplification, or any other kind of device which drastically limits search for solutions in large search spaces. Heuristics do not guarantee optimal solutions: in fact they do not guarantee any solution at all; all that can be said for a useful heuristic is that it offers solutions which are good enough most of the time.'

The challenge is therefore to find a heuristic that produces as near an optimum solution as often as possible.

There are far too many optimisation heuristics to list and describe them all. This research has concentrated on two main algorithms: Simulated Annealing and Evolutionary Algorithms. Simulated Annealing was chosen as being a relatively straight forward technique to use to begin exploring optimisation heuristics. However the focus moved on to evolutionary algorithms as these were being used with financial applications (Coello Coello, 2006). Finally the bulk of the work concentrates on investigating improvements to Multi-Objective Evolutionary Algorithms as these are widely used in Portfolio Optimisation (Castillo Tapia & Coello Coello, 2007).

3.2. Simulated Annealing

Simulated annealing (SA) owes its name to the annealing process that takes places when metal cools. This physical process involves careful control of the temperature and cooling rate (Yang, 2008). The application to search optimisation originated from research by Kirkpatrick, Gelatt and Vecchi (1983). Since then it has been proved that for certain problems SA will converge to a global maximum or minimum if there is enough randomness and the process is gradual. The SA algorithm allows for the solution to be improved but when no improved solution can be found then, with some probability, a poorer solution is accepted. This acceptance of weaker solutions means that the search algorithm does not stay in a local minimum but can climb out to find the global minimum.

As time goes on the probability of taking a weaker solution decreases so convergence to the optimal solution is more likely. The pseudo code for this algorithm, based on Yang (2008), follows.

```
1   begin
2       Objective function $f(x)$
3       Initialize $T = T_0$ and $n = 0$
4       Set initial guess $\mathbf{x_0}$ randomly;
5       Set final temperature $T_f$ and number of iterations N
6       Define parameter $\alpha$ for cooling schedule
7       while ($T>T_f$ and $n<N$)
8           Neighborhood search: $\mathbf{x_{new}}=\mathbf{x}_n$+rand
9           Calculate $\delta f=f(\mathbf{x}_{new})-f(\mathbf{x}_n)$
10          if solution is improved
11              $\mathbf{x_{n+1}}=\mathbf{x_{new}}$
12              $x_*=x_{new}$
13          else
14              Generate a random number r
15              if ($p=\exp[-\delta f/T]>r$)
16                  $x_{n+1}=x_{new}$
17              end if
18          end if
19          $n=n+1$
20          update T
21      end while
22  end
```

The code begins in line 2 with the objective or cost function that needs minimizing. Line 3 sets the initial 'temperature' and the number of iterations to zero. Line 4 sets the initial guess (x^0) as a random number. Line 5 sets the final 'temperature' and so determines how long the algorithm is going to run for if the final temperature is reached before the final number of iterations *N* is reached. Some versions just specify number of iterations. Line 6 defines the way in which the next temperature is set ie the rate of cooling. Line 7 sets the conditions for the while loop so this runs until either the final temperature is reached or the number of iterations set is met. Line 8 conducts a basic neighborhood search; the next value of *x* is determined by taking the previous value and adding a random number. Line 9 calculates the gradient of the function at this new point i.e. the change in energy. Line 10 determines whether this is accepted. If the new value is better, $\delta f < 0$, then this new value is accepted with x_* being the best solution found. Otherwise, in lines 14-16 a new random number is obtained and the solution is accepted with the probability $P = e^{\frac{-\delta f}{T}}$ modelled using the Boltzmann distribution and comes from the second law of thermodynamics (Rutenbar, 1989). This allows for a worse solution to be accepted thus enabling the algorithm to jump out of a local minimum.

3.3. Evolutionary Algorithms

An evolutionary algorithm (EA) is a general term used for algorithms that mimic the Darwinian process of evolution by natural selection. Throughout the different iterations of the algorithm 'fitter' populations are produced (Jones, 2004).

There are a number of different types of evolutionary algorithm and these generally fall into one of three main categories: Evolutionary Programming, Evolution Strategies and Genetic Algorithms (De Jong, 2006). This work focusses on Genetic Algorithms but uses the generic term Evolutionary Algorithm as there are aspects of these EAs that do not totally coincide with the specifications of a Genetic Algorithm.

As the name suggests, genetic algorithms (GAs) were inspired by natural selection and evolution as seen in the birth of successive generations of plants and animals. The first was developed by John Holland in 1960s. At the starting point are two solutions (parents) taken from an initial population of solutions. These are combined to produce a new solution (children) (Hopgood, 2001). The children keep some characteristics from their parents but new characteristics are also introduced thus enabling more of the search space to be examined. The 'value' of each child's fitness is calculated and the best ones kept and added to the original population. Thus the population evolves towards an optimal solution.

In order to implement a GA the following parameters need to be determined:

- The size of the initial population (*P*).
- The size of the population of parents (*M*).
- The size of the population of off-spring (*N*).
- The method for determining the fitness of a solution and who goes into the new population (*M*).
- The methods used for reproduction.
- The stopping criterion.

Unlike SA which looks at one solution at a time GAs involve creating a population of several solutions. Also, whereas SA uses nearest neighbor to find new solutions, GAs can combine solutions some distance away.

Each solution is termed a chromosome (string) which is made up of various genes which take values called alleles. These usually take binary values and where this (and other criteria) is relaxed then the term Evolutionary Algorithms (EAs) is usually used (De Jong, 2006).

3.3.1. Size of initial population (*P*)

The initial population is made up of approximations to the desired solution that are used as inputs for the EA. Diaz-Gomez, et al. (2007) acknowledge that determining a suitable population size for a particular problem is very important. There is a trade-off between increasing the size of the population sufficiently to allow for the algorithm to search as wide a space as possible and converge with each and having a small enough population so that computation time is reduced. Alander (1992) favours a population of around 50 solutions but concedes that as in nature there are situations where much larger populations and families are necessary.

3.3.2. Size of parent population (*M*)

As with the initial population it is important that this population is large enough to maintain diversity of solutions but not so large that speed of convergence of the algorithm is reduced. De Jong (2006) discusses this and provides examples to show that where large numbers of parents are used (circa 100) convergence to the global optimum is reached more quickly.

3.3.3. Size of offspring population (*N*)

Whilst the parent population reflects the areas of the solution space where the EA is focusing its search based on feedback form earlier generations, the offspring population determines how long solutions remain in the parent population (De Jong, 2006). Replacing parent solutions before they have had an opportunity to reproduce can reduce the diversity of the solution space.

3.3.4. How to determine which solutions survive

In this simple EA we are assuming that a population of M solutions is maintained. Each iteration of the algorithm will produce N offspring from a subset of population M and then the ensuing $M + N$ population is reduced back to M. Both these steps, choosing parents and then choosing the next population can be done using deterministic or stochastic selection methods (De Jong, 2006).

Using deterministic methods, solutions in the initial population P are assigned a number which equates to the number of times they can be selected as parents. To reduce the population back to M a fitness measure is employed and the M best solutions within the $M + N$ population are chosen.

Using stochastic methods, solutions in the initial population are chosen to be parents with probability p_i based on their fitness. A similar method can be deployed to reduce the $M + N$ population back to M.

De Jong (2006) provides information about traditional EA selection categories in Table 1 and observes that it has been demonstrated that stochastic methods provide the greatest chance of preserving diversity and thus increasing the algorithms chances of converging to the optimal solution.

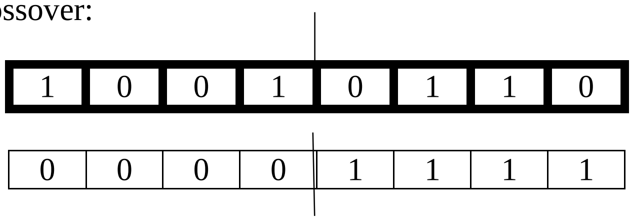
Table 1: Traditional EA selection categories

Evolutionary Algorithm	M	N	**Parent Selection**	**Survival Selection**
Evolutionary Programming	< 20	$N = M$	Deterministic	Deterministic
Evolutionary Strategies	< 10	$N \geq M$	Stochastic	Deterministic
Genetic Algorithms	> 20	$N = M$	Stochastic	Deterministic

Methods of reproduction

Methods of reproduction in EAs are analogous to biological methods of reproduction. Solutions are usually combined using 'crossover' and 'mutation'. The point of crossover needs to be determined and also the number of times crossover takes place. A simple example might be:

Parents before crossover:



Children after crossover:

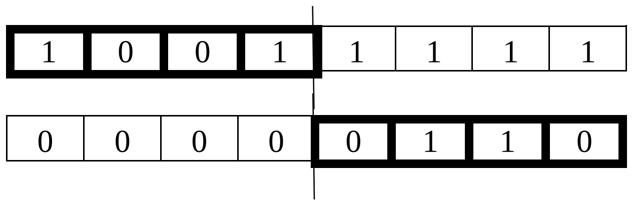


Figure 9: Crossover

Another decision to make is whether both children from this coupling are retained or just one.

In mutation a parent is cloned and then the value of an allele relating to a particular gene in the chromosome is changed. It needs to be decided how many genes can be changed in any one mutation. There follows an example where just one gene is mutated and this is chosen randomly.

Parent before mutation:

0	$\boxed{0}$	0	0	1	1	1	1

Child after mutation:

0	$\boxed{1}$	0	0	1	1	1	1

Figure 10: Mutation

All these variables can be decided stochastically as well as by following a deterministic procedure.

The general algorithm can be described with the following pseudo code:

```
1   begin
2       Define fitness function
3       Generate initial population
4       Define parameters of reproduction
5       while (n<max generations)
6           Generate new solutions using Crossover and Mutation
7           Evaluate new solutions
8           Accept the new solutions if they increase fitness
9           Select the current best solutions for new generation (elitism)
10      end while
11  end
```

According to the pseudo code, the basic features of the evolutionary algorithm are as follows: for the objective function that is being optimized you obtain a population of initial solutions (line 3). Line 4, the parameters of reproduction are then determined. These can consist of the number of parents, the proportion of the population that combines and mutates and the maximum number of generations the algorithm runs for. Lines 5-10 form a loop that runs until the maximum number of generations is completed. This loop will continue to combine and mutate solutions and keep the best (according to the parameters set down by the specific algorithm).

Evolutionary algorithms are similar to GAs except that some of the strict criteria are relaxed. In some applications this means that EA can be more likely to find the global minimum/maximum (De Jong, 2006).

3.3.5. Tournament Selection

At various points during the running of an EA a selection process is required. Tournament Selection is the common choice (Deb, 2001). This involves randomly choosing q solutions from the population, ranking them and selecting the fittest to survive into the next generation. Sometimes this ranking is just based on the fitness measure but other factors can be used. Binary tournaments ($q = 2$) are the most frequently used. In this case two solutions are compared and the fitter solution is chosen and then another pair of solutions is randomly selected (Hopgood, 2001).

3.4. Multi-Objective Optimisation

So far we have been looking at single objective optimisation. But supposing we are looking at a problem like portfolio optimisation which we know has two objectives: maximising return whilst minimising risk. What is the best strategy for solving this?

The problem here is that there is usually no one single optimal solution but rather a number of solutions that are good compromises (or trade-offs). This notion of many optimal solutions was first proposed by Francis Ysidro Edgeworth (Edgeworth, 1881) and later by Vilfredo Pareto (Pareto, 1896). Whilst sometimes referred to as the Edgeworth-Pareto optimum it is more generally called the Pareto optimum.

Here is the standard definition of Pareto optimality taken from Castillo Tapia & Coello Coello (2007):

"A solution $x \in \Omega$ is said to be Pareto Optimal $w.r.t\ \Omega\ iff$ there is no $x' \in \Omega$ for which $\mathrm{v} = \mathrm{F}(\mathrm{x}') = \left(f_1(x'), \ldots, f_k(x')\right)$ dominates $u = \mathrm{F}(x) = \left(f_1(x), \ldots, f_k(x)\right)$."

The phrase Pareto Optimal is taken to mean with respect to the entire decision variable space unless otherwise specified.

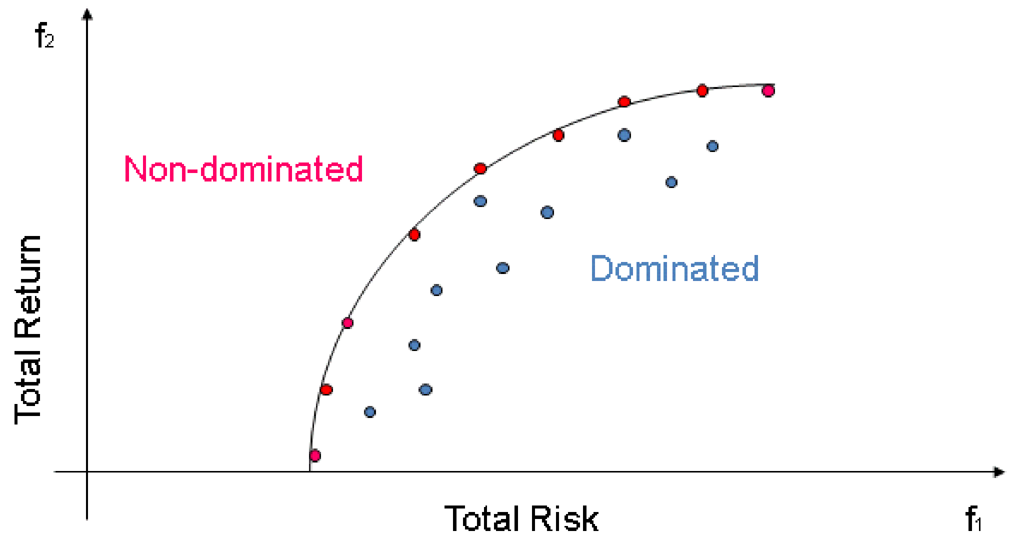


Figure 11: An example of the multi-objective portfolio optimisation problem. The Pareto front or trade-off surface is delineated by a curved line.

Pareto solutions (solutions on the Pareto front) are those solutions within the decision space (Ω) that cannot be improved for all objectives simultaneously. Any such solution means that it can only be improved in terms of one objective which would lead to worsening it in terms of another. In Figure 11 these are the red solutions and they are referred to as non-dominated or undominated solutions. Remaining solutions (the blue ones in Figure 11) are referred to as dominated solutions as other solutions have been found to improve them with regards to all objectives.

It is not easy, and is in most cases impossible, to find an analytic method to compute the Pareto front (Coello Coello, et al., 2006). The standard method would be to produce as many points in Ω as possible and then determine the non-dominated (Pareto optimal) solutions. This is a non-trivial task.

3.5. Multi-Objective Evolutionary Algorithms

Evolutionary Algorithms are well-suited for solving multi-objective optimisation problems due to the fact that they can deal simultaneously with a set of possible solutions (Coello Coello, et al., 2006). Rather than find one solution at a time, as in traditional mathematical programming techniques, they can produce a suite of solutions with each iteration of the program.

The main difference between an EA and a MOEA is that the latter has to consider a vector of objectives rather than a single one. The rest of the characteristics described above remain.

There are numerous different multi-objective algorithms (MOEAs) that have been designed and tested for a wide range of problems. Some of this work has been carried out by computer scientists (this is the focus taken here) and others from an operational research background also taking on the need for the algorithms to work with multi-constrained optimisation problems. This research will focus on the algorithms mentioned below which are the ones most discussed in the literature regarding portfolio optimisation.

It is helpful to provide some definitions for standard MOEAs to enable the reader to see clearly how those below and the new proposed algorithm differ. The notation used by Coello Coello, et al. (2006) has been adopted.

At any time during the running of the MOEA there exists a current population of Pareto solutions represented by $P_{current}(t)$, where t represents the current generation number.

Many MOEAs use a second population, often called an archive, to store previously found non-dominated solutions. This is represented by $P_{known}(t)$. Note that $P_{known}(0)$ will be empty and P_{known} is the final set of Pareto optimal solutions. P_{true}

represents the population of solutions that, if there was an analytic solution, would be non-dominated. The aim of every MOEA is for $P_{known} = P_{true}$.

3.5.1. Dealing with several objective functions

There are various methods for dealing with the often conflicting objectives. Before deciding on the method to use it is best to consider what results are needed for the particular problem. For example if ultimately only one answer is needed then combining objectives into one single objective function might be sensible. This method is called a priori or preference-based approach (Deb, 2001). Weights are assigned to objectives and then summed and normalised into one fitness measure. Its simplicity aids its popularity despite its limitations. Whilst Fonseca and Fleming (1995) show that for any positive set of weights and fitness function a Pareto optimal solution can always be found, Das and Dennis (1997) demonstrate that if all the non-dominated solutions in P_{true} are nonconvex, then not all the front can be obtained. So care must be taken in order to avoid losing solutions.

In the a posteriori approach the full set of non-dominated solutions (PF_{true}) are looked for along the whole length of the Pareto front. Therefore the emphasis is on widening the search as far as possible and not initiating the decision making process until the search has been completed. This can include varying the weights assigned to objectives during the running of the algorithm (Srigiriraju, 2000).

In the Vector Evaluated Genetic Algorithm (VEGA) proposed by David Shaffer (1985) the population is split into groups and the fitness of the solutions in these groups are measured using a different objective chosen at random. VEGA is regarded as being the first documented version of an MOEA; its weakness is that it tends to converge to a local rather than the global optimum.

Deb (2001) discusses an ideal approach where the weights associated with certain objectives are not considered until after the Pareto optimal solutions have been obtained. Then a weight vector is calculated and a trade-off made on each of the solutions. This enables different users to compute different solutions with different weight vectors depending on their priorities.

3.5.2. Methods for Preserving Diversity

In an ideal situation the P_{known} solutions would usually be distributed along the full length of the solution space. In order for this to happen it is important that diversity in the population is maintained and to find some way of actively encouraging less populated areas to reproduce and create more non-dominated solutions. The main method is often referred to as fitness sharing or niching.

In this approach the solution space is broken down into neighbourhoods or niches and the number of solutions located in each niche are counted. The fitness of a niche decreases proportionally to the number of solutions contained in the niche thus ensuring that less populated areas and thus more isolated solutions are kept and allowed to reproduce. The size of the niche is controlled through a niche share radius σ_{share}.

All the algorithms mentioned below use some form of this density estimation and the differences are discussed later.

3.6. Examples of MOEAs

3.6.1. A generic MOEA

According to Coello Coello, et al. (2006) in general an effective MOEA should contain certain observations including removing dominated solutions from the population based on some sort of fitness measure, using a density measure to preserve diversity, performing reproductive operations to generate new individual solutions, removing Pareto dominated and infeasible solutions from the population, and storing non-dominated solutions in an archive.

3.6.2. Elitist Non-Dominated Sorting GA (NSGA and NSGA-II)

Proposed by Srinivas and Deb (1995) NSGA sorts the non-dominated solutions into different fronts. These fronts are then assigned a dummy fitness value corresponding to which front they are in. So, for example, those in the first front might all be assigned a fitness value of 10 while those in the second front might all be assigned 8 etc. Once all the non-dominated solutions have been categorized, then solutions for mating are chosen. The higher fitness ranking for the first front means that these are most likely to be chosen to reproduce.

This enables the algorithm to converge towards P_{true} more quickly. However it is also necessary to preserve diversity so the full front is finally obtained. This means that it is necessary to include solutions from less populated areas even if they are not in the first front of solutions. This niching method was first suggested by Goldberg and Richardson (1987) and has since been widely used in algorithms including NSGA and NSGA-II. The procedure is conducted using the following function *Sh(d)* where d is the Euclidean distance between two solutions and σ_{share} is the niche radius described in 3.5.2 with α normally being 1.

$$Sh(d) = \begin{cases} 1 - \left(\dfrac{d}{\sigma_{share}}\right)^{\alpha}, \text{ if } d \leq \sigma_{share}; \\ 0, \quad \text{otherwise.} \end{cases}$$

This means that whilst solutions in the first front have a greater chance of being selected for reproduction, if they are in a densely populated area fewer of these solutions will be accepted. Conversely solutions from a lower front might have less chance of selection but if they are in a sparsely populated region then this chance will be increased.

However, Coello Coello (2006) reports that this fitness sharing mechanism created a bottleneck in these algorithms. This is because this method involves finding the distance *d* with every population member (Deb, 2001). Although Deb (2001) goes on to say that the computational complexity ($O(MN^3)$) may not be critical, it is clear from his work on NSGA-II (Deb, et al., 2000) that there is a need to reduce this. NSGA-II alleviates this issue by using niching after reproduction which reduces complexity to $O(MN^2)$.

NSGA-II starts off by creating an initial population P size N and then using this to reproduce in order to create an offspring population Q also size N. These are amalgamated to form population R, size $2N$. At this point a nondominated sorting, similar to NSGA, is used to classify the population. The new population is made up from the solutions from the best fronts using tournament-selection and a crowding density measure to enable solutions from an isolated area but on a weaker front to still have a chance to be chosen, thus preserving diversity.

3.6.3. Pareto Archived Evolution Strategy (PAES)

This algorithm by Knowles and Corne (1999) uses a combination of local search and evolutionary methods to reach the Pareto front. A single parent is mutated to form a single child. There is an archive of best solutions and every mutated solution is compared to this archive and either replaces a solution in the archive or is rejected and the parent is mutated again. The maximum size of the archive is maintained.

From the archive, parents for the next generation are chosen. This is done using a density measure so solutions in less crowded areas are most likely to be chosen as parents.

The density calculation is based on a grid or mesh, or set of hypercubes. The number of solutions in each hypercube is counted and, if the offspring is in a less densely populated hypercube than the parent, it is chosen as a parent in the next generation.

If there is no room in the non-dominated archive then the offspring is compared to the solutions in the most densely populated hypercube. If the new solution is not from this box then it is kept in the archive and one of the solutions from this densely populated hypercube is removed.

3.6.4. The Strength Pareto Evolutionary Algorithms (SPEA and SPEA2)

In 1998 Zitzler and Thiele proposed SPEA (Zitzler & Thiele, 1998). This stores an external archive of the best non-dominated solutions and remains a fixed size. Every solution in the archive is given a strength value and a fitness level which takes into account its nearness to the Pareto front and the distribution of other solutions.

The algorithm begins with a randomly created population P of size N and an empty external population $\bar{P}$ with maximum capacity $\bar{N}$. In any generation t the best nondominated solutions in P_t are copied to population $\bar{P}_t$. In each successive generation dominated solutions in $\bar{P}$ are replaced with better ones.

To preserve diversity in the archive, a clustering method is used. Distances between solutions is calculated and the closest solutions merged to keep the archive to $\bar{N}$.

In 2001, Zitzler, Laumanns and Thiele proposed a variation on SPEA which has been claimed to surpass NSGA-II for the portfolio optimisation problem (Zitzler, et al., 2001). SPEA2 differs from SPEA as it has a nearest neighbour density measure, an improved fitness assignment taking into account the number of solutions each solution dominates and is dominated by and an enhanced truncation method which guarantees the preservation of boundary solutions.

The algorithm is similar to NSGA-II in that there is a population and an archive; these are merged, duplicates removed and this represents the population. Each solution in this population is assigned a strength value and a fitness level. The strength value represents the number of solutions that are dominated by this solution.

Solutions are then mated; parents being chosen using binary tournaments. This new population then replaces the old population which remains static in number.

3.6.5. Pareto Envelope based Selection Algorithms (PESA and PESA2)

Proposed by Corne, Jerram, Knowles and Oates (2000) PESA has an internal population and a larger external population. Like PAES it divides the solution space up into hypercubes and a crowding measure is used to make sure that diversity in the selection of parents is maintained.

In 2001 the same team developed PESA2 (Corne, et al., 2001). The main differences between this and PESA is that instead of the solution being selected the hypercube is selected and then any individual solutions used in mating are chosen randomly. The fitness of the hypercube is determined by the number of solutions in it, with the least populated having more chance of being chosen and thus preserving diversity of the new population.

3.7. MOEAs for Portfolio Optimisation

Markowitz’s foundations of Modern Portfolio Theory only works by applying simplifications and assumptions in order to reduce computational complexity (Maringer, 2005). These include assuming that markets are perfect in that there are no taxes nor transactional costs, assets are infinitely divisible, investors make decisions at one point in time and the means, standard deviations and correlations are sufficient to describe the asset’s returns. Heuristics such as Evolutionary Algorithms can take care of these assumptions and provide more accurate solutions because they can iteratively search for and test new improved or modified solutions until some convergence criterion is reached (Maringer, 2005).

Another issue with Portfolio Optimisation is that it is important to have good results quickly. The customer does not want to wait for an algorithm to slowly examine the problem and eventually reach an optimum solution but would rather have a quick but accurate range of solutions depending on how risk-averse they are. The traditional mean-variance approach can take a long time when various constraints have to be taken into account.

In recent years many groups have begun working on fast, efficient Evolutionary Algorithms specifically for Portfolio Optimisation. In 2007 Castillo Tapia and Coello Coello surveyed a number of problems in economics and finance and work done to address these using evolutionary algorithms (Castillo Tapia & Coello Coello, 2007). They noted that the area of Portfolio Optimisation was where the bulk of work had taken place.

Maringer (2005) lists a number of heuristics that can be used in different areas of modern portfolio theory and concludes that "there is not one best heuristic that would be superior to all other methods. It is rather a 'different courses, different horses' situation where criteria……influence the decision which heuristic to choose."

He also notes that genetic algorithms are complex and need experience to implement. Despite this, evolutionary algorithms are the focus for many research papers in the last twenty years as they provide a way of addressing the additional constraints more effectively that then traditional approach described by Markowitz (Schlottmann & Sesse, 2004).

In 2007 Skolpadungket, Dahal and Harnpornchai discussed Portfolio Optimisation using Multi Objective Genetic Algorithms (Skolpadungket, et al., 2007). They compared several algorithms including SPEA2 and NSGA-II on data in the OR library. Their conclusions were that SPEA2 performed best in terms of finding a diverse set of Pareto optimal solutions.

In 2009 Branke, Scheckenbach, Stein, Deb and Schmek published on Portfolio Optimisation with an envelope-based multi-objective evolutionary algorithm (Branke, et al., 2009). They noted that various real-world cardinality constraints led to a non-convex search space and, because of this, the traditional methods can no longer be applied. Their algorithm involved allowing the MOEA to find convex subsets of feasible portfolios and then merging partial solutions to form a full solution of the original non-convex problem.

Previously in 2008 Chiam, Tan and Mamum published on constrained Portfolio Optimisation (Chiam, et al., 2008) and compared different algorithms with various constraints using the data from the OR-library (Beasley, 2012).

However these papers have built on earlier research experimenting with using multi-objective evolutionary algorithms to solve variations of the Portfolio Optimisation problem. These include Shoaf & Foster (1996) who describe a genetic algorithm for the efficient set portfolio problem based on the Markowitz model. This offers significant benefits over the quadratic programming approach including the ability to simultaneously optimize risk and return. Vedarajan, et al. (1997) propose a MOEA approach to portfolio selection searching for non-dominated feasible solutions with respect to two objectives and also include a variant where a transaction cost is an additional component which can cause problems for standard quadratic programming problems. Chang, et al. (1999) consider the problem of finding the efficient frontier including cardinality constraints that limit a portfolio to a specified number of assets. Lin, et al. (2001) consider the variation to the Markowitz problem which included constraints on the amount of money that could be invested in each asset as well as a fixed transactional cost.

This work shows that not only is there a value in using multi-objective evolutuonary algorithms to solve the portfolio selection problem but that there is still more work to be done to find fast, efficient algorithms that incorportate all necessary constraits.

3.8. Performance

It is not easy to judge how well a MOEA has performed as one has to compare fronts as opposed to individual solutions (Branke, et al., 2009). Zitzler, et al. (2003) discuss a number of different performance measures such as measuring the distance of an approximated front to the Pareto-optimal front, measuring the diversity of the approximated front using a chi-squared-like deviation measure and measuring the volume of space dominated by the approximated front.

Branke, et al. (2009) use the difference in area from the known Pareto front as calculated using the critical line algorithm and their approximation. This was based on the Portfolio Optimisation problem. They acknowledge that it is not ideal as it is necessary to define values for maximum variance and maximum return. If there is a large gap between these values then this can have a significant impact on the quality of the solution whereas if they are set too close then some parts of the front may be cut off.

3.9. Summary

This chapter has provided the theory of optimisation necessary for understanding how to create a variety of new optimisation algorithms. It began with an overview of how to find the global minimum and maximum using calculus and various numerical iterative processes such as gradient search before going on to look at two heuristics; simulated annealing and evolutionary algorithms. These can be used when the function is complex and simpler methods tend to result in the finding of local minima and maxima.

The terminology and parameters used in evolutionary algorithms, such as methods of reproduction and ways of creating populations, were discussed in detail in this chapter, in order to prepare the reader to understand the way in which the algorithm works.

The rest of the chapter concerned Multi Objective Evolutionary Algorithms (MOEAs). After an introduction as to why MOEAs are needed there then followed a description of some of the key aspects of the general algorithm such as how to deal with multiple objectives and different methods for preserving diversity.

Following this, several important algorithms: The Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II), the Pareto Archived Evolution Strategy (PAES), the Strength Pareto Evolutionary Algorithm (SPEA and SPEA2) and the Pareto Envelope-based Selection Algorithm (PESA and PESA2) were discussed in more detail.

The final section looked at various research where these MOEAs have been used to solve the Portfolio Optimisation problem and how it is possible to measure how well an algorithm has performed. An understanding of this is important in order to see how well the proposed new MOEA described in chapter 4 performs.

4. New Algorithms for Financial Optimisation

This chapter describes the main body of new work that this research has focused on. It begins with a simulated annealing algorithm designed to calculate the parameters for GARCH (1,1). In section 4.1 some results are presented which compare the parameters found using this algorithm against those found using the maximum likelihood method.

This is followed in section 4.2 by an evolutionary algorithm that calculates the values of nine different trading indicators in order to see if a share should be bought or sold on a particular day. The evolutionary algorithm then combines and mutates combinations of these indicators in order to find rules that optimise the amount of profit made.

The above work was exploratory and was used to direct and inform the subsequent research. Results given for both these algorithms are brief due to the fact that these paved the way for the more substantive work on multi-objective evolutionary algorithms.

The main work developed a multi-objective algorithm for portfolio optimisation and this is where the majority of the chapter is focused. The discussion of this algorithm in section 4.3 explains how the reproduction process is carried out, how solutions are stored, and how the adaptive cell-based system ensures that diversity is preserved.

The performance of the algorithm using the comparison between the area under P_{true} and the area under the found Pareto front (P_{known}) is discussed in section 4.4 and results are presented for varying methods of cell adaptivity. The algorithm was also tested to see how sensitive it is to random numbers.

4.1. A Simulated Annealing Algorithm for GARCH

The Stock Exchange of Mauritius (SEM) is classed as one of the emerging markets. Established in 1989, it has grown considerably in size and now attracts much interest from foreign investors. Thus understanding the volatility in the SEM is increasingly important as well as providing a suitable test case for our algorithm.

Data from the SEMDEX index was collected from 2000 to 2007 and from this 12 companies from a range of industries were chosen for the study. The main reason for the choice of these companies was that their data was clean and there were no breaks.

A Simulated Annealing program was written to find the GARCH (1,1) parameters for the SEM data similar to the program described and discussed by Maringer (2005). The Maximum Likelihood approach was used.

Firstly historic volatility was calculated from the daily returns in the dataset. Then with a small tolerance level the program perturbs values for α, β, ω in order to maximise

$$\sum_{i=1}^{m}\left[-\ln(\sigma_i^2) - \frac{u_i^2}{\sigma_i^2}\right]$$

Where m is the number of recent observations, u_i the daily return and σ_i the daily variance.

The daily long term volatility (V) is then found by combining the weights so that

$$V = \frac{\omega}{1 - \alpha - \beta}$$

Initial results with the SEM data looked promising but the algorithm was then run on data from the US Standard and Poor (S&P) index to enable them to be compared to other studies. The algorithm was found to converge quickly with results comparable to those found using MS Excel Solver and other commercial software when running for 1000 iterations with shareprices from the S&P500. Further work could be done to test the speed of convergence with different levels of tolerance. Before this was carried out the work moved on to look at ways of determining rules of trading using Evolutionary Algorithms.

Table 2: Comparison of parameters for GARCH(1,1) using data from the S&P500

Software	**Maximum**	**Long-term Volatility**	α	β	ω
GARCH SA	11833.2440	1.259427	0.0795888	0.7678411	0.0000242
MS Excel Solver	11833.3493	1.702257	0.0693761	0.8291634	0.0000294
Other software	11833.2752	1.438048	0.0728163	0.7927531	0.0000278

Table 2 shows the values for the parameters α, β and ω as well as the value for the maximum obtained with the new SA algorithm for GARCH and compared with the values found using Excel MS solver and a recommended solver found on the internet. This shows that the SA algorithm was finding values close to that of reputable solvers.

As mentioned in Section 2.2.4 it can be seen that the values for β are high thus indicating that volatility is persistent as any shocks will take time to die out. The long-term volatility is also high as this is usually below 0.8 (Hull, 2006). However the important aspect is that the new SA algorithm found similar results to publically available solvers.

4.2. Finding Trading Rules using Trading Indicators

4.2.1. Overview

There is evidence in the literature that applying the results of individual technical indicators to the forecasting of asset pricing has a limited increase on the investment return (Fama, 1970). Thus it has become practice to combine the indicators to form rules. Whilst Brock, et al. (1992) discuss the importance of computational methods for determining profitable trading rules Allen & Karjalainen (1999) demonstrate how this can be done with a genetic algorithm.

To investigate the workings of a single objective evolutionary algorithm and to gain practice of Java programming it was decided to investigate technical analysis by forming trading rules. This was done by writing an evolutionary algorithm in Java, similar in format to that described by Allen & Karjalainen (1999). The algorithm was initially written and tested with the GARCH (1,1) problem described in section 3.2 to make sure it was performing as expected. Similar results were achieved for the GARCH (1,1) parameters thus the new algorithm was considered to be functioning well.

The program was then adapted to calculate the best trading rules for different shares. The algorithm uses data from 12 companies from the Standard and Poor 500 index in the US. Mirroring the work conducted by Allen & Karjalainen (1999) it calculates the value of 8 different trading indicators shown below (described in section 2.5) and evaluates their return (positive or negative) at the end of the day. If the indicator gives a positive return then shares should be bought, if negative then shares should be sold and if zero then shares are held.

Table 3 List of trading indicators used in evolutionary algorithm

Trading Indicator
Short Price Channel Breakout
Long Price Channel Breakout
Short Moving Average
Stochastic Oscillator
Long Moving Average
Simple moving average crossover
Moving Average Convergence Divergence
Relative Strength Index

4.2.2. Finding the rules

The program started off using training data ie historic data where the buy and sell signals are known and performed calculations to evaluate the various trading indicators listed in the previous section.

Then the evolutionary algorithm combines and mutates different indicators in order to find out which rules (combinations of indicators) should be used in order to maximise profit for each share. The fitness of a rule is determined by evaluating how much extra return would be acquired as compared to a traditional buy and hold policy. The simple return from a single trade (R_i) which is bought at date b_i and sold at date s_i is

$$R_i = \frac{P_{s_i}}{P_{b_i}} \times \frac{1-c}{1+c} - 1$$

Where P_t is the closing price on day t and c is the one-way transaction cost (expressed as a fraction of the price).

From this the daily continuously compounded return (r) is calculated

$$r = \log P_t - \log P_{t-1}$$

Letting T be the number of trading days, $r_f(t)$ denoting the risk-free rate on day t , m the number of trades and defining two indicator variables, $I_b(t)$ and $I_s(t)$, which equate to one if a rule returns a buy or sell signal respectively and equate to zero if otherwise we have

$$r = \sum_{t=1}^{T} r_t \, I_b(t) + \sum_{t=1}^{T} r_f \, (t) I_s(t) + n\log\frac{1-c}{1+c}$$

The total simple return is $R = e^r - 1$

The return for the buy and hold strategy is

$$r_{bh} = \sum_{t=1}^{T} r_t + \log\frac{1-c}{1+c}$$

And thus the excess return or fitness for a trading rule is given by

$$\Delta r = r - r_{bh}$$

The actual algorithm can be described using the following steps (Allen & Karjalainen, 1999):

Step 1

- Create a random rule.
- Compute the fitness of the rule as the excess return over the buy and hold strategy.
- Do this 500 times to form the initial population.

Step 2

- Apply the fittest rule in the population to the selection period and compute the excess return. Save this as the initial best rule.

Step 3

- Pick two parent rules at random.
- Create a new rule using crossover and calculate the fitness.
- Replace one of the old rules with this new rule.
- Repeat 500 times in each generation.

Step 4

- Apply the fittest rule in the population and calculate the excess return.
- If the excess return improves upon previous best rule then this is saved as the new best rule.
- If no improvements after 25 generations or after a total of 50 generations stop. Otherwise go back to step 3.

4.2.3. Results

The outcome was that different shares gravitated to different rules. For BP the successful rule was made up of the following indicators

- Price Channel Breakout
- Short Moving Average
- Stochastic Oscillator
- Short v long Exponentially Weighted Moving Average
- Moving Average Convergence Divergence

However data from some shares (such as BP) converged more easily than others (such as General Motors) showing that in some cases it is not possible to determine particular rules. The rules were then used on test data; this consists of share price data that is known but not used in formulating the rules. By using the rules to determine when shares would have been bought and sold the amount of profit/loss over the period was calculated.

More work would be required on this before results could be presented however the writing of this algorithm led to interest in multi-objective evolutionary algorithms and portfolio optimisation which is where the rest of this research is focused.

4.3. Adaptive Cell Resolution Evolutionary Algorithm (ACREA)

4.3.1. Overview

As previously stated, in order for a MOEA to find a close approximation to the Pareto front it needs to be able to maximise the diversity of the population in the solution space and have a method of controlling the density of the solutions. This proposed Adaptive Cell Resolution Evolutionary Algorithm (ACREA) does this using three methods:

- A dynamic population.
- A grid whose cell size changes dynamically.
- A depth of solutions along the solution front which changes dynamically.

These are described in more detail below and in the rest of this section.

It is hard for a fixed population size to maintain a diverse population so a dynamic population that will grow as necessary is proposed.

The problem chosen to test ACREA was an unconstrained portfolio optimisation problem where the variance of the portfolio was minimized

$$\sigma_p^2 = w^T V w = \sum_{i=1}^{x} \sum_{j=1}^{x} w_i \, w_j \sigma_{i,j}$$

and the return of the portfolio was maximized

$$u_p = \sum_{x=1}^{X} w_x \, u_x$$

subject to

$$\sum_{x=1}^{X} w_x = 1$$

as previously described in section 2.6.3 .

Each individual solution is made up of a series of weights which refer to specific assets. So for example the solution:

w_1	w_2	w_3	w_4

Each of the w_x is a proportion of a specific asset.

This algorithm is similar to PESA in that it divides the solution space into boxes which are called grid cells. If the problem were more than two dimensional this could be generalized to hypercubes like PESA. Another algorithm that uses this grid as a base is the DMOEA (Yen & Lu, 2003). The difference that is proposed here is that as well as a changing population size ACREA also has a dynamic grid where the individual grid cells can change in size both horizontally and vertically. As the number of generations increase, the number of grid cells in the solution spaces increases, thus shrinking the dimensions and surface area of each grid cell. The population size is also dynamic, increasing in size with each generation as mutations and combinations are added to it. The number of non-dominated solutions is not limited and thus grows as the population grows.

Because of the increased computation resulting from the population growth, the algorithm is slowed down. To minimize this, and as not all the solutions are worth keeping, the width of the solution front is restricted and can change dynamically. All grid cells beyond the front width are emptied. Also, only a certain number of the non-dominated solutions are retained. The pseudocode that describes this algorithm follows:

```
begin
    Initialise Fitness Function
    Generate initial population
    Define parameters of reproduction
    Define density levels
    Define grid and rate at which cells grow
    Populate grid with population depending on normlised risk and variance
    while nGenerations<maxGenerations
        Perform mutation and crossover
        Measure fitness of new solutions
        Keep mutated solution and best crossover from each parent
        if (grid size trigger point reached)
            Increase grid size
            if (density>q)
                remove weakest solutions in each cell
                if (population size trigger point reached)
                    Decrease solution front width in x and y directions
                end if
            end if
        end if
    end while
end
```

After the initialization of the algorithm where all the parameters are set (lines 0-5) the solution space (grid) is populated with random solutions based on their values of risk and variance in line 7. The fitness of these solutions is calculated and stored. Line 8 sets up a loop to run for each generation until the stopping criteria is reached. In line 9 mutation and crossover operations are performed. Line 10 the fitness of these solutions is measured. Line 11 the fittest crossover and single mutation is kept for each mating. Line 12 starts an If statement declaring the trigger point for grid expansion. This is discussed more in section 4.3.4. The trigger point mentioned in the code can be the start of each new generation or every n generations depending on the method used. This is discussed further in section 4.3.3.

In line 13 the grid increases its mesh density as described in section 4.3.4. Lines 14 and 15 say that all solutions are kept until the density reaches *q*. At this point the weakest solutions are removed from individual cells so that the maximum number of solutions in each cell is *q*. Lines 16 and 17 declare the trigger point for reducing the number of cells behind the solution front. Once this point is reached the cells furthest away from the front (behind a previously declared distance), in each direction x and y, are all emptied to reduce the number of solutions in the population.

Individual aspects of the algorithm are explained in more detail below.

4.3.2. Reproduction

4.3.2.1 Mutations

A solution is chosen at random to be mutated. Each solution is made up of weights (relating to assets). One of the non-zero weights in this solution is chosen at random. This weight is decreased by a small amount (*n*). This has changed the sum of all weights by *n* so this amount *n* is then added to another randomly chosen weight to maintain the original total. The fitness of the mutated solution is assessed and the solution is stored in the grid.

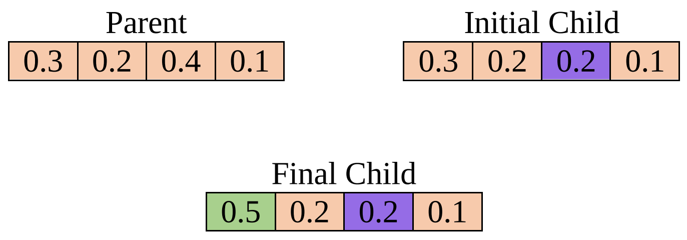


Figure 12 Mutations in ACREA

4.3.2.2 Combinations

Two solutions are chosen at random to be parents. The solutions are combined to produce two offspring and then the one with the greater fitness level is kept and stored in the grid.

Each weight (i) in a solution is assigned a random number in turn. If this is less than 0.5 then the weight (i) of the first child is equal to the weight (i) of parent 2 and the weight (i) of the second child is equal to the weight (i) of parent 1. Otherwise the other way around.

However, as with mutations, it can be seen that this changes the sum of the weights which is important in this problem. Again similarly to the mutations, a weight is chosen at random and altered to make sure that the original weight is maintained. In the following example child 1 needs to lose 0.3 and child 2 needs to gain 0.3. A solution, if weight one was chosen, is shown in the final step of Figure 13.

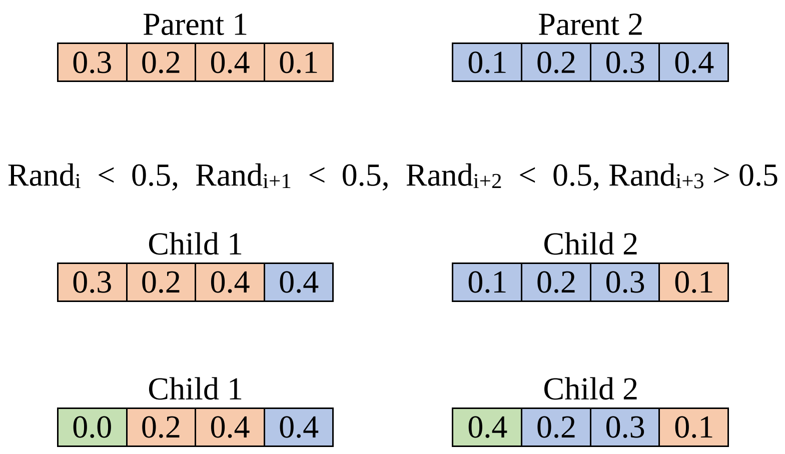


Figure 13: Combinations in ACREA

4.3.3. The Grid

As said before, the algorithm divides the solution space into a grid or mesh similar to PESA. The population is placed into this grid according to the solutions' fitness. Solutions are randomly chosen to mutate or to become parents and be combined. This new population (including mutants and combinations) is then put into the grid. This continues until the maximum number of generations is reached.

Initially, the grid has 50 grid cells in each direction x and y. This is an arbitrary value that tests show works well. The resolution of the grid then increases in both directions every generation based on a particular function. Different functions for the increase of the grid resolution have been investigated and are discussed below. If a simple increasing function was used every 10 generations and the initial number of grid cells in each direction was 5 then the grid would look like Figure 14 at 0, 10, 20 generations.

However, without some method of reducing the population this would increase by the number of mutations and combinations produced at each generation, which, if left to continue, would become so large that it would slow down the algorithm considerably. To combat this, the population is reduced by emptying grid cells that are a certain distance away from the solution front in both x and y directions.

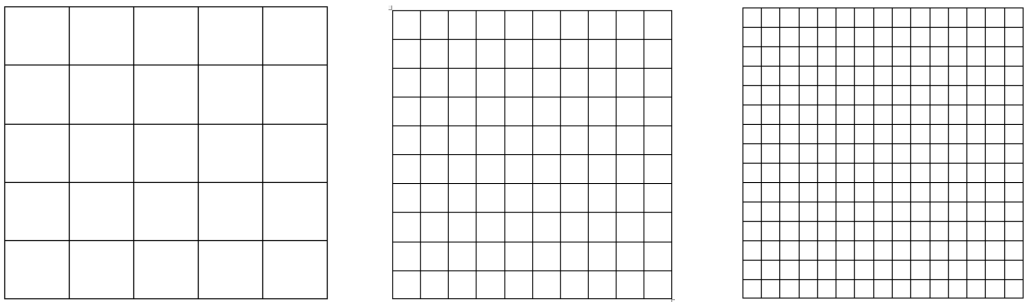


Figure 14: Example of changing grid cell size with initial grid of 5x5 increasing by 5 every 10 generations

Initially the algorithm runs for 10 complete generations to allow the population to grow by mutating and combining solutions. After this, only the solutions in the first 150 width of grid cells from the solution front in both x and y directions are kept. This value was chosen after testing, as it seems to keep enough solutions to increase diversity of the solution space but not too many that the algorithm is slowed down. If, at any point, the population size reaches the maximum number of solutions then the retained number of grid cells from the front (again in both x and y directions) decreases to 90. As the number of solutions increase the solution width needs to be reduced to avoid the algorithm taking too long to reach a solution. Various sizes for the retained number of grid cells were tested and these were found to perform best. However more rigorous testing is required to determine the optimal width of grid cells to maintain.

All the time this reduction is taking place, the number of grid cells is increasing (size of individual grid cell is shrinking). Figure 15, Figure 16 and Figure 17 provide a schematic view of this. The *'s represent grid cells containing non-dominated solutions. In this example the retained number of grid cells from the front (a cell containing non-dominated solution) is 3 so the shaded grid cells, being further away, are emptied. As the algorithm continues to run, the grid cells decrease in size and the solution begins to converge. As the generations increase, the grid cells shrink in size and thus more solutions are removed.

Figure 15 shows which cells are emptied if the algorithm was just doing this in the vertical direction whereas Figure 16 shows the cells to be emptied in the horizontal direction. One could choose to either combine these so that either the union or intersection cells are emptied. The current version of the ACREA algorithm empties the cells in the union.

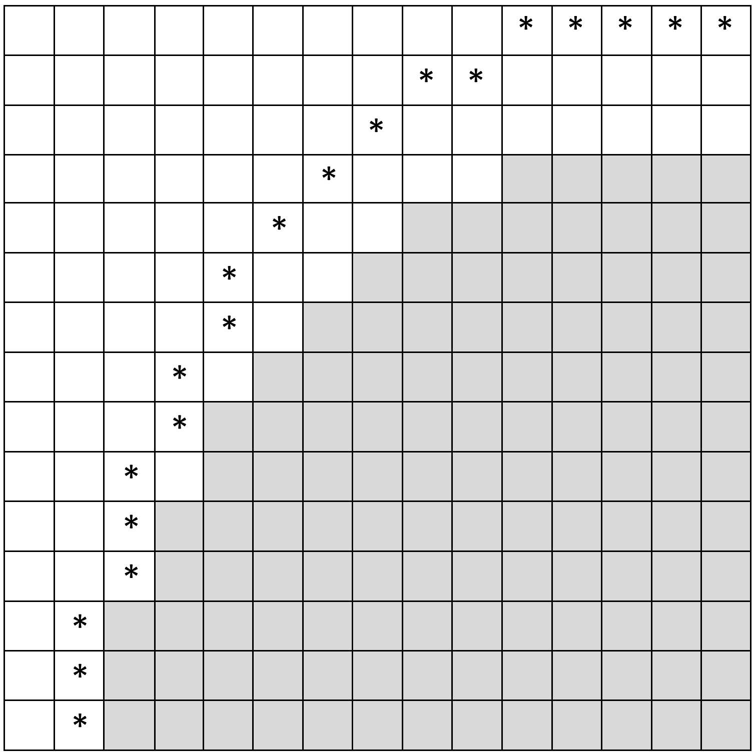


Figure 15: Diagram to show which cells would be emptied in vertical y direction

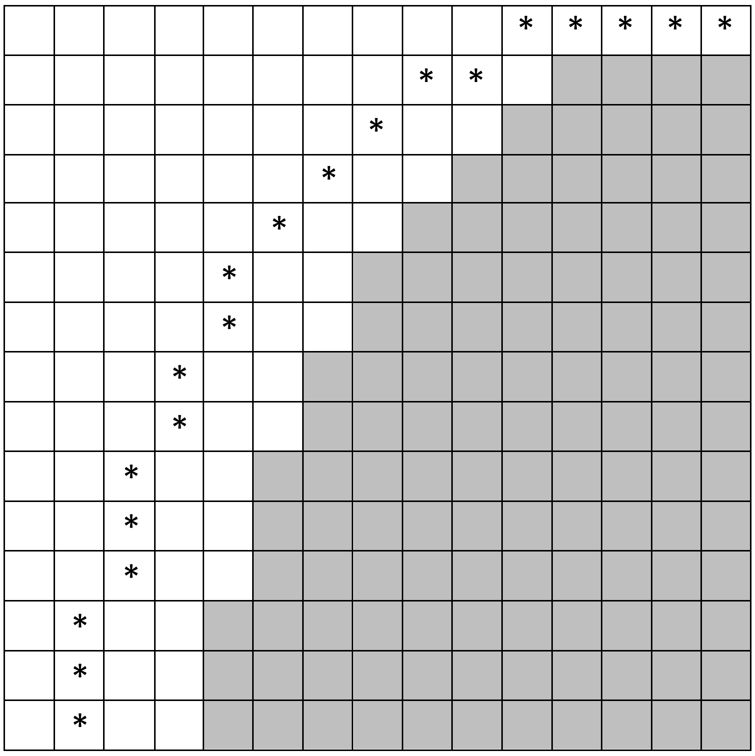


Figure 16: Diagram to show which cells would be emptied in horizontal x direction

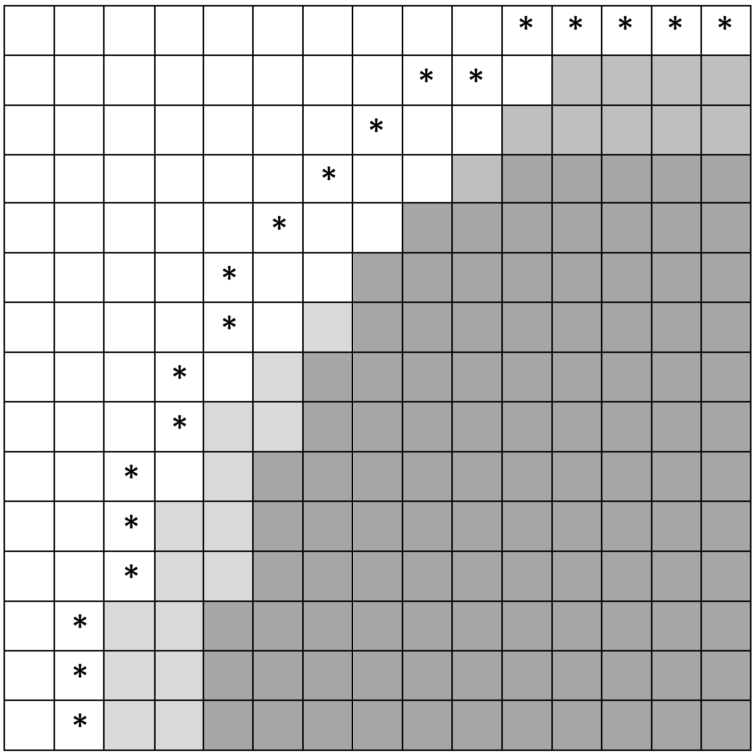


Figure 17: Diagram to show which cells would be emptied in both x and y directions

As the resolution of the grid increases and the grid cells become smaller, it is necessary to control the number of solutions in each cell. The number of solutions in each grid cell is determined by the density value. This value can be set to a specific level throughout the running of the algorithm or changed at different points. After experimentation, in the current implementation, the density value starts off at 6 then decreases to 4 after 100 generations, then 3 after 170 and 2 after 250. This helps to control the population size but also increases diversity because, even if a particular solution is not as close to the Pareto front as others, it will still be kept if it is in a sparsely populated grid cell.

4.3.4. Cell increase

The area of the grid is a finite space that does not change. This space is divided into grid cells which are easier to visualise with two dimensional problems. Several versions of the function used in the adaptive grid algorithm have been tested (here named Grid 1 through to Grid 4). Initially the algorithm ran with 100 cells in each direction and then, every 20 generations (Grid 1) the number of cells in each direction in this solution space increased by 20. This seemed to work adequately for small data sets but was not so good for large ones. To increase the rate at which the grid resolution increased, various experiments were carried out to see what would happen if the grid resolution equated to *kG* where *k* is a number that could be changed and *G* the number of the generation that the program is on. This made a slight improvement but was still not converging for larger datasets. Grid 2 in Figure 18 shows the resolution when *k*=4.

Thus a new algorithm was developed which meant that the number of cells increased as a logarithmic function. The advantage of this was that the population size increased quickly so there was far greater diversity early on but this did not carry on exponentially. Two variations of this, Grid 3 and Grid 4, were tried and for this method the starting point of 50 grid cells in each direction was used.

Grid 3: Number of cells = Max(50, 600 ln (G))

Grid 4: Number of cells = Min(x_{+}(0.02x),4000) where x=50 for n=2 as the starting point for the grid is 50 cells in both directions.

Figure 18 compares the 4 strategies, showing the slow growth of Grid 1, the slow but exponential increase for Grid 2, asymptotic increase for Grid 3, and the fast exponential followed by slow linear increase for Grid 4.

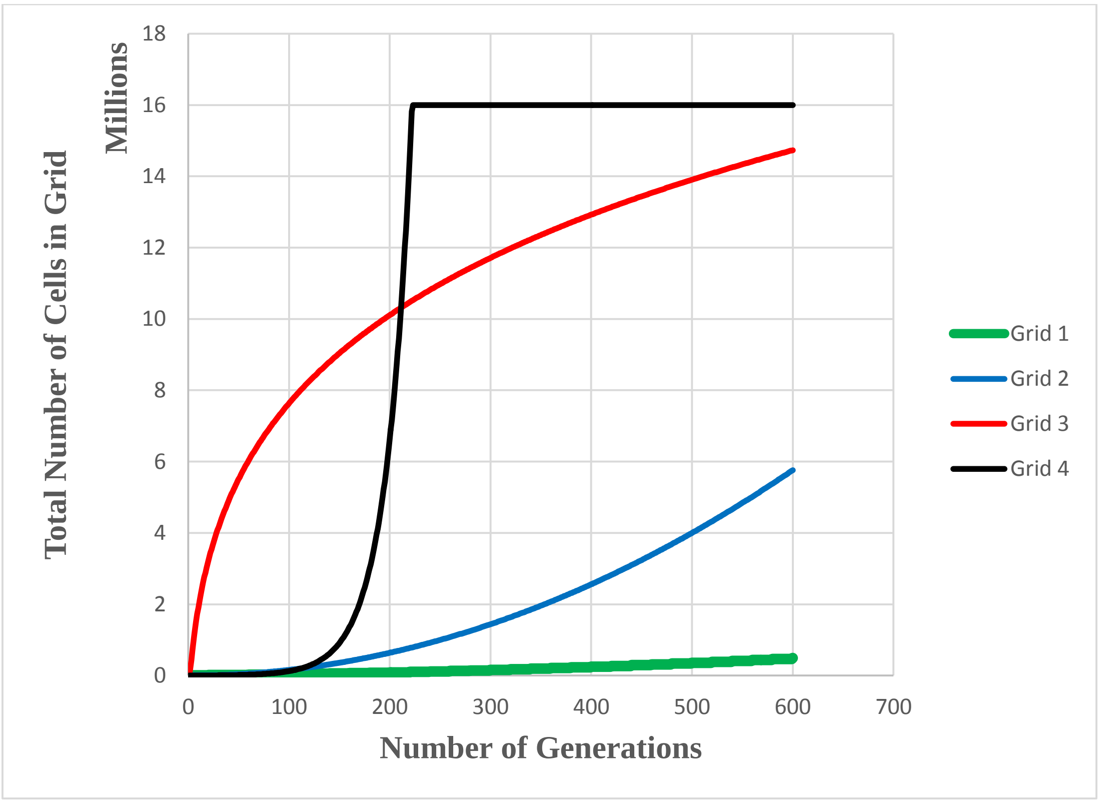


Figure 18: Graph to show how cell resolution increases over number of generations

4.3.5. Implementation

The algorithm was implemented with data from Beasley's OR Library (2012) consisting of estimated returns and variances for groups of assets in different stock market indices. The same data has been used in similar work by Chiam, et al. (2008)

Skolpadungket, et al. (2007) and Branke, et al. (2009). Table 4 shows the number of assets in each of the data sets as well as the original source of the data.

Table 4: Table to show the number of shares in each dataset

Problem index	Data Source	Number of assets
Port1	Hang Seng	31
Port2	Dax 100	85
Port3	FTSE 100	89
Port4	S&P 100	98
Port5	Nikkei 225	225

When testing the algorithm at different stages it was obvious that the problem became more complex and slower when dealing with larger quantities of data and thus it was important to have an algorithm that could accommodate larger data sets. For this reason this work concentrates on the largest dataset, Port 5. Other algorithms, such as the one described by Chiam, et al. (2008), struggled to find the Pareto front with this data set.

4.4. ACREA Results

4.4.1. Area Ratio

In line with other similar studies the difference in area between the true Pareto front (P_{true}) and the front generated by ACREA (P_{known}) was measured in order to provide a measure of performance of ACREA. The analytical solutions contained in P_{true} were given in the dataset provided by the OR Library (Beasley, 2012). This comparison of results can be seen in Table 8.

In order to calculate the difference between the areas of ACREA and P_{true} it was necessary to make certain adjustments where the maximum risk and minimum return values of P_{known} were greater than P_{true} and vice versa. This is in line with other research such as Branke, et al (2009).

In the case where the maximum risk and minimum return values of P_{known} were greater than P_{true} boundary points were established to remove some of the most extreme solutions. This avoided the situation where large areas bounded by very few solutions at the extreme points are compared. These boundary points were identified as being the smallest return value (A) and the largest risk value (B) for P_{true}. This is demonstrated in Figure 19.

In the situation where the maximum risk and minimum return values of P_{true} were greater than P_{known} then the smallest risk and largest return values were extended and the resulting area CDE calculated. This is shown in Figure 20

This enables sensible comparisons to be made without outliers making otherwise good solutions seemingly have a very large difference between the areas.

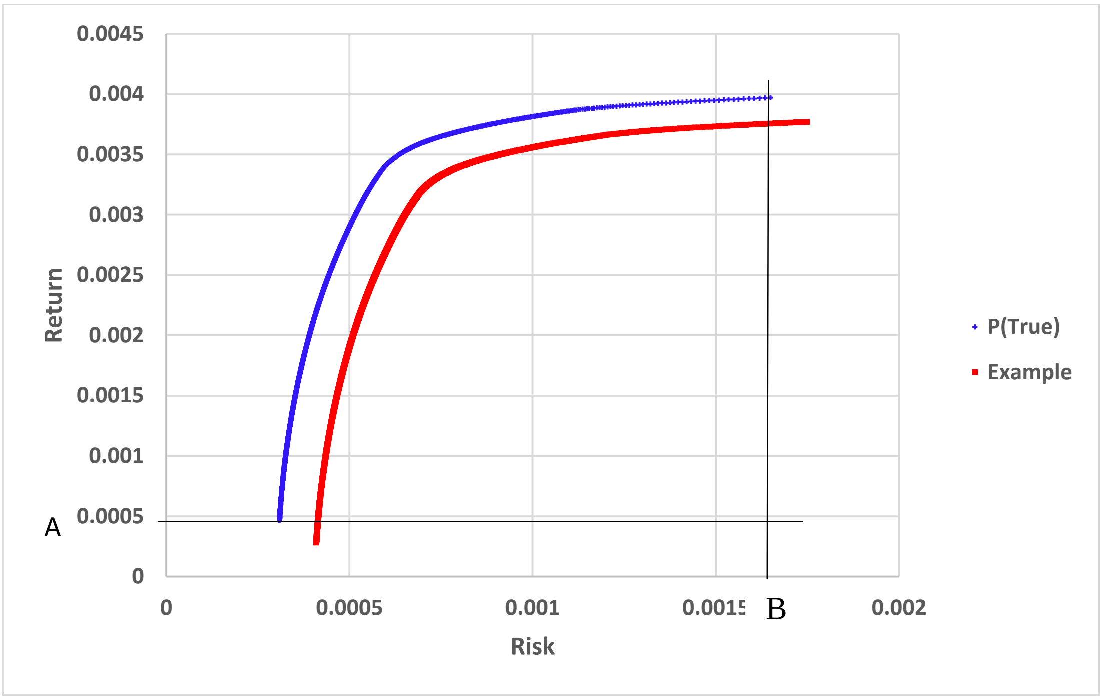


Figure 19: Area boundaries where the maximum risk and minimum return values of P_{known} are greater than P_{true}

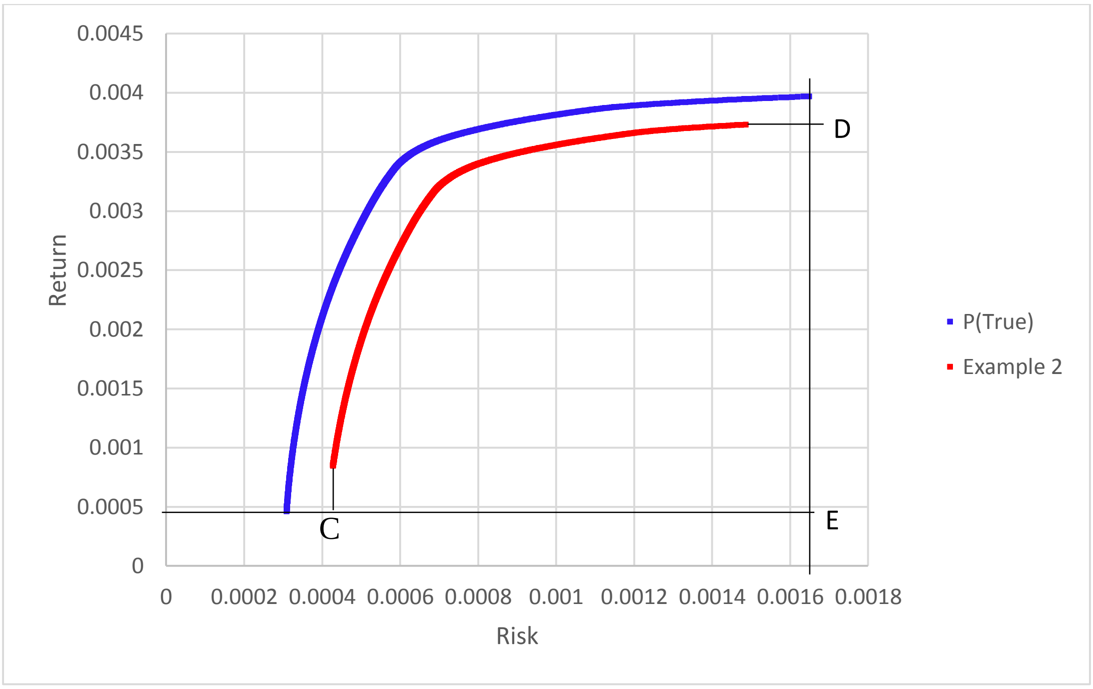


Figure 20: Area boundaries where the maximum risk and minimum return values of P_{true} are greater than P_{known}

4.4.2. Parameters

The following table shows a list of the different parameters that are used within the ACREA algorithm along with the maximum and minimum values as well as the value used in the best performing version of the algorithm. Some systematic testing was carried out which has determined some of the optimal values but further tests need to be carried out.

Table 5 table showing the different parameters used in the ACREA algorithm

Parameter	**Minimum value**	**Maximum value**	**Value in best performing version**
Initial population size	20	500	20
Proportion of mutations	0.1	0.9	0.9
Proportion of combinations	0.1	0.9	0.9
Cell density value	4	10	4
Front width	1	1	1
Number of generations	400	600	600
Scale grid (function)	100 (increase by 20 Grid 1)	4000 (Grid 4)	
Population ceiling	150000	500000	150000

4.4.3. Implementation

ACREA was programmed using Java. This language was chosen as it was one that was most familiar. The mean processing time of the runs in Table 6 is 9639 seconds with the median 9536 seconds, so just over two and a half hours. All the tests in this thesis were run on a machine with a dual processor at 2.53GHz with 2.96GB of RAM.

4.4.4. Sensitivity

In order to check how sensitive the algorithm was to different random numbers a short experiment was constructed where the best algorithm was run 10 times so with 10 different random seeds. Although the time taken for each run, total population and the number of non-dominated solutions all changed, the area ratio (discussed in section 4.4.1) remained very similar. The results are shown in Table 6.

Table 6: Sensitivity

Run	Final Population	Non-Dominated	Area Ratio	Processing time
1	100350	2802	1.000957531	9608
2	100197	2809	1.00099939	8990
3	99410	2916	1.001035822	9131
4	98677	3022	1.000972169	9760
5	99188	2688	1.000993987	9150
6	99339	2775	1.001056081	10498
7	101626	2963	1.00096566	9005
8	101979	2880	1.000942046	9464
9	96426	2958	1.000967243	10016
10	99785	2795	1.000968716	10769

In order to measure the variability of the data in Table 6 the mean, range, standard deviation and interquartile range were calculated. These measures are shown in Table 7. From this it can be seen that the very small values for the range, standard deviation and IQR show that there is little spread in the data and thus demonstrate that the algorithm is not sensitive to random numbers.

Table 7: Measures of variability

Variability Measure	Area Ratio
Mean	1.000985865
Range	0.000114035
Standard deviation	3.59544E-05
Inter-quartile range	3.19835E-05

4.4.5. Main Results

Initial results for ACREA are very positive. In a short time the algorithm was able to find a solution very close to the analytically obtained Pareto front. As said before all results discussed, result from using the share price dataset Port 5 from the OR library's Portfolio Optimisation dataset (Beasley, 2012) which contains the largest number of assets.

Several experiments were undertaken to determine which parameters produced the best results. The main set of experiments were undertaken with the Grid 4 method for the adaptive grid. Table 8 shows the results for different variations of mutations and combinations and cell density with the difference in area under the Pareto front between P_{known}, the new algorithm, and P_{true}, the analytical solution, being the key measure. This final measure is described in section 4.4.1. The ratio rather than the difference is used because when there is very little difference between the two areas it is easier to see how close they are to one. All runs listed in Table 8 used 600 generations and a population ceiling of 150,000. These parameters had previously been found to be particularly good. This is just a selection of all the runs and different permutation of parameters in order of performance (best first) in terms of area ratio.

Table 8: ACREA results

Run #	Initial population	Proportion of mutations	Proportion of combinations	Cell density	Area ratio
1	20	0.9	0.9	4	1.00094
2	100	0.9	0.9	6	1.00096
3	20	0.9	0.9	6	1.00097
4	20	0.9	0.9	4	1.00097
5	20	0.5	0.9	4	1.00098
6	20	0.9	0.9	4	1.00099
7	50	0.9	0.9	4	1.00099
8	20	0.9	0.9	6	1.00104
9	20	0.9	0.5	4	1.00121
10	50	0.1	0.9	4	1.00131
11	20	0.9	0.1	4	1.00215

The final population sizes for the best three runs are given in Table 9 where ‘best’ is defined as the smallest area ratio.

Table 9: Three best ACREA results

Run #	Final Population	# Non-dominated Sol	Area ratio
1	101979	2880	1.000942
2	100350	2802	1.000958
3	96426	2958	1.000967

Secondly the difference between the methods for cell increase was investigated. Grid 4 and Grid 3 were compared along with variations of Grid 2 where *k* was taken as 5, 10, 15 and 20. These larger values of *k* had not been examined before and have produced better results than the previous experiments with Grid 3 and Grid 4. This is presumably because, for higher values of *k*, the final number of grid cells is much larger as shown in Figure 21. The grid size for these new versions of Grid 2 are shown in Figure 21 alongside the previous methods used in Grid 3 and Grid 4.

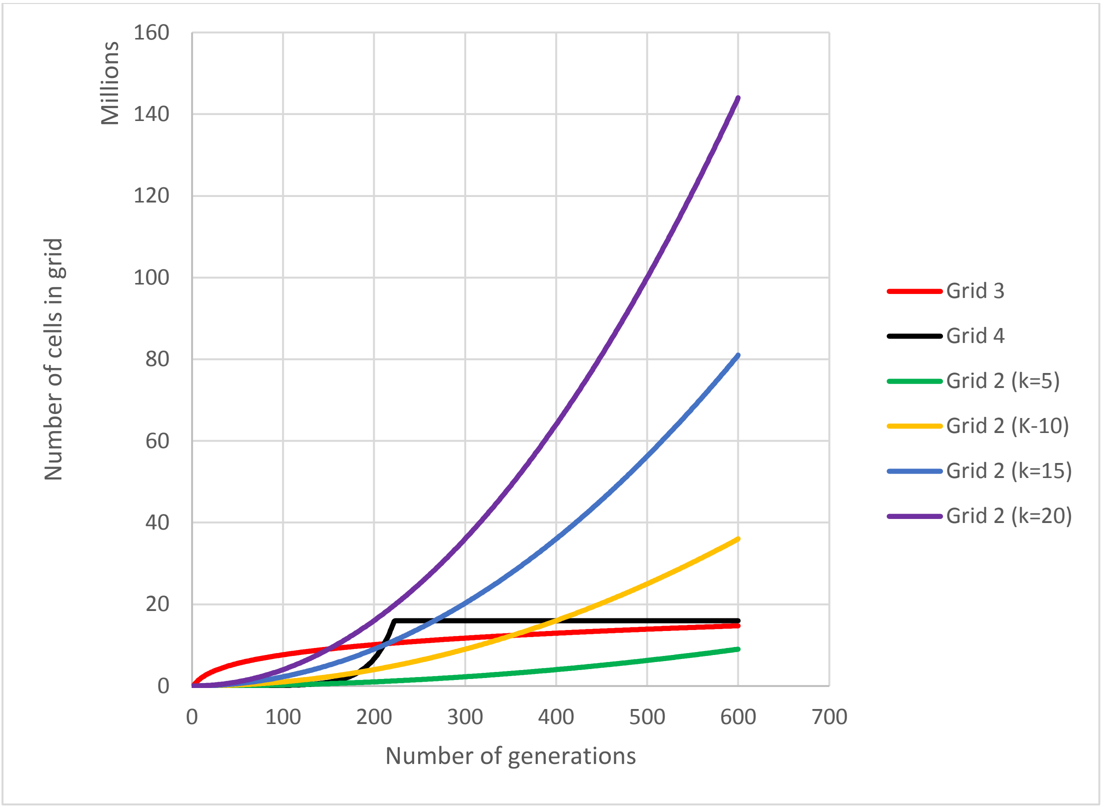


Figure 21: Grid cell multiplication

The same parameters as run 1 were chosen in order to compare the different algorithms. However it is possible that different methods of increasing the mesh size might work best with different parameters. This is something that could be investigated later.

Table 10 shows the results comparing the final population size, non-dominated population size and area ratio of the different methods for determining the rate at which the grid cells increase.

Table 10: Grid multiplication

	Final population	**Non-dominated pop**	**Area ratio**
Grid 4	101979	2880	1.000942
Grid 3	84297	2078	1.001086
Grid 2 (10)	124387	2801	1.001085
Grid 2 (15)	98965	4303	1.000736
Grid 2 (20)	130688	5697	1.0006

The function used in Grid 2 had previously been discounted as it failed to converge for small values of k with the larger data sets. However with larger values of k it can be seen that it out performs the other functions and future work would include running further tests with this type of function.

4.4.6. Performance

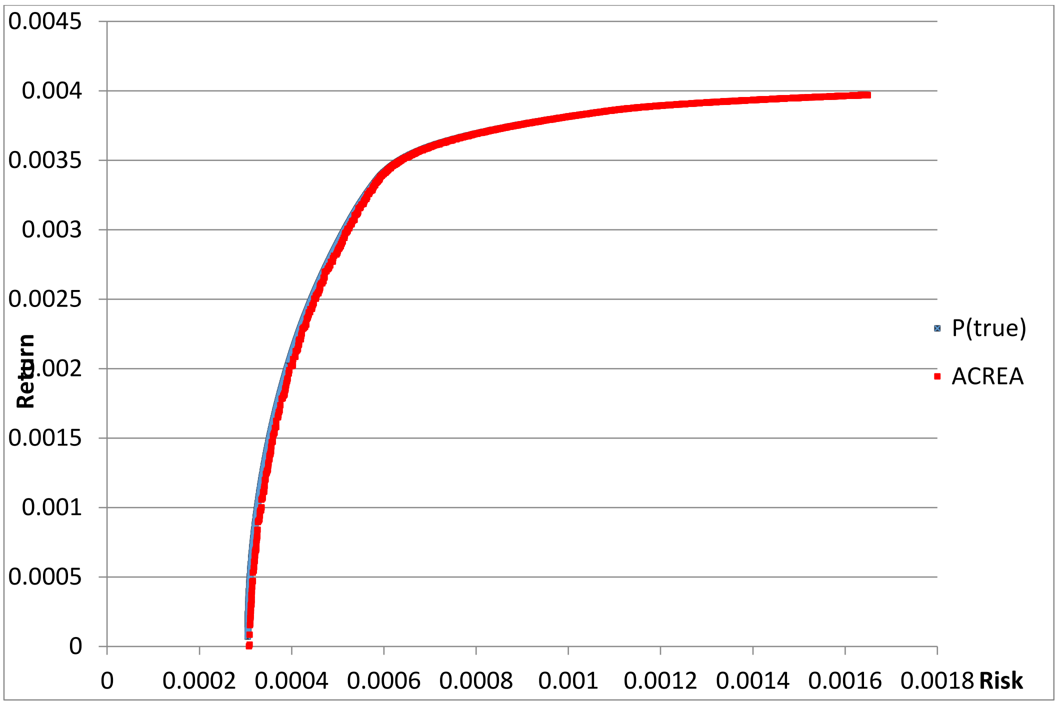


Figure 22: Graph to show non-dominated solutions from ACREA compared to known Pareto front.

As discussed in section 4.4.1 the difference in area between the true Pareto front (P_{true}) and the front generated by ACREA (P_{known}) was measured in order to give some idea as to the performance of ACREA. The best results show no discernable difference when graphed, however, where the algorithm does not perform so well (where both risk and return are low), graphing the results shows the gaps in the front. An example of this is shown in Figure 22. Figure 23 shows one of the sparser sections of the graph when magnified to make it easier to see where the gaps occur.

It is possible that if the algorithm was set to perform more iterations that these gaps would be filled. Further tests need to be carried out before any conclusions can be drawn from this phenomenon.

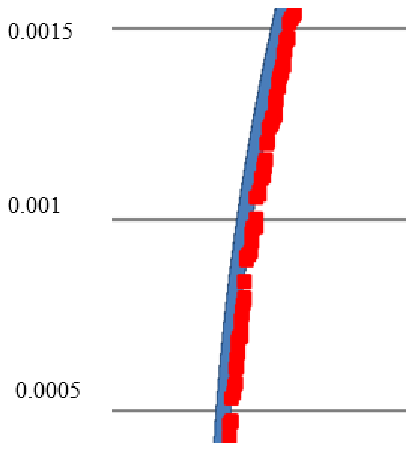


Figure 23: Diagram to show gaps in Pareto front produced by ACREA

4.5. Summary

This chapter described the main body of new work that this research has focused on. It began with a simulated annealing algorithm designed to calculate the parameters for GARCH (1,1). Some results were presented which compared the parameters found with this algorithm against those found using the maximum likelihood method.

After this an evolutionary algorithm was presented. This EA used data from 12 companies in the Standard & Poor 500 and calculated the values of nine different trading indicators in order to see if a share should be bought or sold on a particular day. The evolutionary algorithm then combined and mutated combinations of these indicators in order to find rules to optimise the amount of profit made. This work led to an interest in multi-objective optimisation which was the main area of research.

The main part of the chapter is devoted to a description of the multi-objective algorithm that was developed for portfolio optimisation. The algorithm is called Adaptive Cell Resolution Evolutionary Algorithm (ACREA). This is because it involves using a mesh of grid cells which adapt in terms of both the number of cells in all dimensions and the size of the grid cells as the number of generations increases.

As well as explaining how the reproduction process is carried out, this chapter details how the grid mechanism adapts by changing in resolution ie the number of grid cells in the solution space. The number of solutions stored in each cell varies throughout the algorithm. Also when the population gets too large the cells containing the weakest solutions are emptied allowing the best solutions to be the ones reproducing.

The chapter concludes with a discussion on the performance of the algorithm measured by comparing the area between the Pareto front, P_{known}, found by the algorithm and P_{true} followed by a set of results which look at the algorithm's sensitivity to random numbers and the difference between various methods of calculating how quickly the cell resolution changes.

5. Opportunities for further research and Conclusions

This final chapter starts by discussing future work that could be done both to develop ACREA further and also to assess its performance more thoroughly. This includes collecting more results, vigorous analysis on the method for adapting the grid cell size and testing it against other known MOEAs. As well as this the algorithm would benefit from a user-friendly front end to allow the user to set the maximum level of risk that they were prepared to tolerate in order for them to have the choice of suitable portfolios.

5.1. Further work to be carried out

5.1.1. Testing against other known MOEAs

To be able to test if ACREA does outperform other MOEAs on the Portfolio Optimisation problem it needs to be tested against other algorithms on the same machine and problem. This includes testing against the MOEA used in the Matlab optimisation toolbox which is based on NSGA-II.

5.1.2. Determine optimal parameters

Further tests also need to be carried out to find out what the optimal growth rate of the grid is, optimal population size, reproduction percentages and density values. After this the algorithm needs to be tested on larger data sets such as used the study by Branke, et al (2009).

5.1.3. Further surveying of recent literature

Also due to the time elapsed between this research and the writing of this thesis a new survey of the literature is essential to understand and take account of more recent work in this area.

5.1.4. Investigation of hybrid algorithms

Possible hybrid algorithms have been discussed looking at starting off with an evolutionary algorithm and then moving to a particle swarm algorithm to see if this speeds up computational time.

5.1.5. Inclusion of user-friendly front end

A user-friendly front end which allows the investor to easily choose the optimal portfolio from a range of optimal portfolios based on the level of risk that they are willing to accept would be a beneficial addition.

5.2. Conclusion

This thesis has presented work on three heuristics for financial applications. This work began with a Simulated Annealing algorithm to find the parameters for GARCH(1,1). The results produced compared favorably with results obtained using commercially available solvers. Then, in order to understand how to code an evolutionary algorithm, code was written to determine trading rules based on a series of trading indicators. Few results were produced with this algorithm as it was decided to move on to develop a multi-objective evolutionary algorithm for portfolio optimisation.

The main part of this thesis is devoted to a description of this multi-objective algorithm that was developed for portfolio optimisation. The algorithm is called Adaptive Cell Resolution Evolutionary Algorithm (ACREA). ACREA involves using a mesh of grid cells which adapt in terms of both the number of cells in all dimensions and the size of the grid cells as the number of generations increases.

ACREA provides an interesting addition to the suite of currently used MOEAs. Initial research shows that it performs at least as well as other MOEAs but further research is needed in order to make more detailed comparisons on performance.

The thesis explains how the reproduction process is carried out and how the grid mechanism adapts by changes to the resolution ie the number of grid cells in the solution space. The number of solutions stored in each cell varies throughout the algorithm. Also when the population gets too large the cells containing the weakest solutions are emptied allowing the best solutions to be the ones reproducing.

The performance of the algorithm is measured by comparing the area between the Pareto front, P_{known}, found by the algorithm and P_{true} followed by a set of results which look at the algorithm's sensitivity to random numbers and the difference between various methods of calculating how quickly the cell resolution changes.

Although this has only been tested on the Portfolio Optimisation problem to date there is no reason why it could not be used on MOEAs with more than two objective functions.

References

Adams, A. E. A., 2003. *Investment Mathematics.* Chichester: Wiley.

Alander, J., 1992. On optimal population size of genetic algorithms. *Computer Systems and Software Engineering Proceedings,* pp. 65-70.

Alexander, C., 2001. *Market Models: A guide to financial data analysis.* Chichister: John Wiley and sons.

Allen, F. & Karjalainen, R., 1999. Using genetic algorithms to find technical trading rules. *Journal of Financial Econometrics,* Volume 51, pp. 245-271.

Anon., 2015. *Investopedia.* [Online]
Available at: http://www.investopedia.com/active-trading/technical-indicators/
[Accessed 2 April 2015].

Artzner, P., Delbaen, F., Eber, J.-M. & Heath, D., 1999. Coherant Measures of Risk. *Mathematical Finance,* Volume 9, pp. 203-228.

Barr, A. & Feigenbaum, E., 1986. *The Handbook of Artificial Intelligence : Volume III.* Reading: Addison-Wesley.

Beasley, J., 2012. *OR-Library.* [Online]
Available at: http://people.brunel.ac.uk/~mastjjb/jeb/info.html
[Accessed 24 March 2015].

Bekaert, G. & Hodrick, R., 2011. *International Financial Management.* 2nd ed. Harlow: Prentice Hall.

Bollerslev, T., 1986. Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics,* 3(3), pp. 307-327.

Branke, J., Scheckenbach, B., Stein, M., Deb, K. and Schmeck, H., 2009. Portfolio optimisation with an envelope-based multi-objective evolutionary algorithm. *European Journal of Operational Research,* 199(3), pp. 684-693.

Brock, W., Lakonishok, J. & LeBaron, B., 1992. Simple technical trading rules and the stochastic properties of stock returns. *Journal of Finance,* Volume 471731-1764.

Brooks, C. & Tsolacos, S., 2010. *Real Estate Modelling and Forecasting.* New York: Cambridge University Press.

Castillo Tapia, M. & Coello Coello, C., 2007. Applications of multi-objective evolutionary algorithms in economics and finance: A survey. In: *IEEE Congress on Evolutionary Computation.* s.l.:IEEE, pp. 532-539.

Chang, T., Meade, N. & Beasley, J., 1999. Heuristics for Cardinality Constrained Portfolio Optimisation. *Computers and Operational Research,* 27(13), pp. 1271-1302.

Chiam, S., K.C., T. & Al Mamum, A., 2008. Evolutionary Multi-Objective Portfolio Optimisation in Practical Context. *International journal of Automation and Computing,* 05(1), pp. 67-80.

Coello Coello, C., 2006. Multi-objective optimisation and its use in finance. In: J. Rennard, ed. *Handbook of Research on nature Inspired Computing for Economy and Management.* Hershey, PA: Idea Group Publishing.

Coello Coello, C., Lamont, G. & Van Veldhuizen, D., 2006. *Evolutionary Algorithms for Solving Multi-Objective Problems.* 2nd ed. New York: Springer.

Corne, D., Jerram, N., Knowles, J. & Oates, M., 2001. PESA-II: Region-based Selection in Evolutionary Multi-objective Optimisation. *proceedings of the Genetic and Evolutionary Computation Conference GECCO,* pp. 283-290.

Corne, D., Knowles, J. & Oates, M., 2000. The Pareto envelope-based selection algorithm for multi-objective optimisation. In: M. Schoenauer, et al. eds. *Parallel problem solving from nature - PPSN VI Springer Lecture Notes in Computer Science.* s.l.:Springer , pp. 869-878.

Das, I. & Dennis, J., 1997. A Closer Look at Drawbacks of Minimising Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimisation Problems. *Structural Optimisation,* 14(1), pp. 63-69.

De Jong, K., 2006. *Evolutionary Computation.* New Delhi, India: Prentice Hall.

Deb, K., 2001. *Multi-Objective Optimisation using Evolutionary Algorithms.* Chichister: John Wiley & Sons Ltd.

Deb, K., Agrawal, A., Pratab, A. & Meyarivan, T., 2000. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimisation: NGSA-II. *Parallel Problem Solving from Nature VI,* pp. 849-858.

Dempster, M. & Jones, C., 2001. A real-time adaptive trading system. *Quanitative Finance,* Volume 1, pp. 397-413.

Diaz-Gomez, P. & Hougen, D., 2007. *Initial Population for Genetic Algorithms: A Metric Approach in roceedings of the 2007 International Conference on Genetic and Evolutionary.* Las Vegas, Nevada, USA , GEM .

Doerner, K., Gutjahr, W.J., Hartl, R.F., Strauss, C. & Stummer, C., n.d. Pareto Ant Colony Optimisation: A Metaheuristic Approach to Multi-objective Portfolio Selection. *Annals of Operations Research,* 131(1-4), pp. 79-99.

Donchian, R., 1957. Trend-Following Methods In Commodity Price Analysis. *Commodity Year Book*, p. 35.

Edgeworth, F., 1881. *Mathematical Psychics.* London: P.Keagan.

Ehrgott, M., Klamroth, K. & Schwehm, C., 2004. An MCDM approach to portfolio optimisation. *European Journal of Operational Research,* 155(3), pp. 752-770.

Elton, E., Gruber, M., Brown, S. & Goetzmann, W., 2011. *Modern Portfolio Theory and Investment Analysis.* 8th ed. Hoboken: John Wiley and Sons.

Engle, R., 1982. Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of U.K. Inflation. *Econometrica,* 50(4), pp. 987-1007.

Fama, E., 1970. Efficient Capital Markets: a review of theory and emperical work. *Journal of Finance,* Volume 25, pp. 383-417.

Fieldsend, J., Matatko, J. & Peng, M., 2004. Cardinality constrained portfolio optimisation. In: Z. Yang & e. al, eds. *Intelligent Data Engineering and Automated Learning, IDEAL2004.* Berlin: Springer-Verlag, pp. 788-793.

Fonseca, C. & Fleming, P., 1995. An Overview of Evolutionary Algorithms in Multi-Objective Optimisation. *Evolutionary Computation Journal,* 3(1), pp. 1-16.

Goldberg, D. & Richardson, J., 1987. Genetic Algorithms withsharing for multilodal function optimisation. *Proceedings of the First International Confernece on Genetic Algorithms and their Applications,* pp. 41-49.

Graham, T., 2004. *Dynamic Optimisation of Technical Trading Rules Using Genetic*

Programming. s.l.:University College London.
Hand, D., 2009. Mathematics or mismanagement: the crash of 2008. *Mathematics Today,* 45 (4), pp. 15-16.
Hopgood, A., 2001. *Intelligent Systems for Engineers and Scientists.* Florida, USA: CSC.
Hopgood, A., 2001. *Intelligent Systems for Engineers and Scientists.* Florida, USA: CSC.
Hull, J., 2006. *Options, Futures and other Derivatives.* 6th ed. New Delhi: Prentice-Hall.
Jones, G., 2004. Genetic and Evolutionary Algorithms. In: *Encyclopedia of Computational Chemistry.* s.l.:John Wiley & Sons, Ltd.
JP Morgan, 1996. *RiskMetrics Technical Document,* New York: J.P. Morgan/Reuters,.
Kirkpatrick, S., Gellat, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. *Science,* 220(4598), pp. 671-680.
Knowles, J. & Corne, D., 1999. The Pareto Archived Evolutionary Strategy: A New Baseline Algorithm for Multi-objective Optimisation. *1999 Congress on Evolutionary Computation,* pp. 98-105.
Krivo, 2012. *Daily FX.* [Online]
Available at:
http://www.dailyfx.com/forex/education/trading_tips/post_of_the_day/2012/05/10/Breakout_Trades_and_the_Power_of_Price_Channels.html
[Accessed 2 April 2015].
Lin, D., Wang, S. & Yan, H., 2001. *A multi-objective genetic algorithm for portfolio selection,* Beijing, China: Institute of Systems Science, Academy of mathematics and Systems Science Chinese Academy of Sciences.
Lo, A., Mamaysky, H. & Wang, J., 2000. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *Journal of Finance,* Volume 40, pp. 1705-1765.
Malkiel, B., 1989. Is the Stock Market Efficient?. *Science,* 243(4896), pp. 1313-1318.
Malkiel, B., 1996. *A Random Walk Down Wall Street: The Best Investment Advice Money Can Buy - Including a Life-Cycle Guide to Personal Investing.* New York: W.W. Norton and Company .
Malkiel, B., 2003. The Eficient Market Hypothesis and Its Critics. *Journal of Economic Perspectives,* 17(1), pp. 59-82.
Maringer, D., 2005. *Portfolio Management with Heuristic Optimisation.* Dordrecht: Springer.
Markowitz, H., 1952. Portfolio Selection. *The Journal of Finance,* 1(7), pp. 77-91.
Meucci, A., 2010. Quant Nugget 2. *Linear vs. Compounded Returns – Common Pitfalls in Portfolio Management* , May, pp. 49-51.
Murphy, C., 2015. *Investopedia: Moving Averages.* [Online]
Available at: http://www.investopedia.com/university/movingaverage/
[Accessed 2 April 2015].
Muruganantham, A., Yang, Z., Bong Gee, S., Qiu, X. & Chen Tan, K., 2013. Dynamic Multi-objective Optimisation Using Evolutionary Algorithm with Kalman Filter. *17th*

Asia Pascific Symposium on Intelligent and Evolutionary Systems, pp. 66-75.

Pareto, V., 1896. *Cours D'Economie Politique.* Volume ed. Lausanne: F.Rouge.

Ready, M., 2002. Profits from technical trading rules. *Financial Management,* 31(3), pp. 43-61.

Rutenbar, R., 1989. Simulated annealing algorithms: An overview. *IEEE Circuits and Devices Magazine,* January.pp. 19-26.

Schaffer, J., 1985. Muliple Objective Optimisation with Vector Evaluated Genetic Algorithms. *Proceedings of the First International Conference on Genetic Algorithms,* pp. 93-100.

Schlottmann, F. & Sesse, D., 2004. Financial Applications of Evolutionary Algorithms: recent developments and future research directions. In: C. Coello Coello & G. Lamont, eds. *Applications of Multi-objective Evolutionary Algorithms.* New Jersey: World Scientific, pp. 627-652.

Shoaf, J. & Foster, J., 1996. *A Genetic Algorithm Solution to the Efficient Set Problem: A technique for Portfolio Selection based on the Markowitz Model.* Orlando, Florida, s.n., pp. 571-573.

Skolpadungket, P., Dahal, K. & Harnpornchai, N., 2007. *Portfolio Optimisation using Multi-objective Genetic Algorithms.* Singapore, IEEE.

Srigiriraju, K., 2000. *Nonlinear Surface Tracing Evolutionary Algorithm (NSTEA) for Multi-Objective Optimisation,* Raleigh: North Carolina State University.

Srinivas, N. & Deb, K., 1995. Multi-objective function optimisation using nondominated sorting genetic algorithms. *Evolutionary Computation,* 2(3), pp. 221-248.

Srinvas, N. & Deb, K., 1995. Multi-objective optimisation using nondominated sorting in genetic algorithms. *Evolutionary Computation,* Volume 2, pp. 221-248.

Steinsaltz, D., 2009. Mathematics or mismanagement: the crash of 2008 (with a response by D.J. Hand). *Newsletter of the London Mathematical Scoiety,* September no 384, pp. 23-25.

Stockcharts, 2015. *Stockcharts.* [Online]
Available at:
http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:moving_average_convergence_divergence_macd
[Accessed 18 July 2015].

Vedarajan, G., Chan, I. & Goldberg, D., 1997. *Investment Portfolio Optimisation using Genetic Algorithms.* California, Stanford University, pp. 255-263.

Wilmott, P., 2001. *Paul Wilmott Introduces Quanitive Finance.* Chichister: Wiley.

Yang, X.-S., 2008. *Nature-Inspired Algorithms.* Frome, UK: Luniver Press.

Yen, G. & Lu, H., 2003. Dynamic multi-objective Evolutionary Algorithm: Adaptive Cell-based Rank and Density Estimation. *IEEE Transactions on Evolutionary Computation,* 7(3), pp. 253-274.

Zitzler, E., Laumanns, M. & Thiekle, L., 2001. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm,* s.l.: s.n.

Zitzler, E. & Thiele, L., 1998. *An Evolutionary Algorithm for Multi-objective Optimisation: The Strength Pareto Approach,* Zurich: Swiss Federal Institute of

Technology.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M. & da Fonseca, V.G., 2003. Performance assessment of multi-objective optimisers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2), pp. 117-132.