Portfolio Optimisation with Transaction Cost

by

Maria Woodside-Oriakhi

A thesis submitted for the degree of

Doctor of Philosophy

January 2011

School of Information Systems, Computing and Mathematics,

Brunel University

Abstract

Portfolio selection is an example of decision making under conditions of uncertainty. In the face of an unknown future, fund managers make complex financial choices based on the investors perceptions and preferences towards risk and return. Since the seminal work of Markowitz, many studies have been published using his mean-variance (MV) model as a basis. These mathematical models of investor attitudes and asset return dynamics aid in the portfolio selection process.

In this thesis we extend the MV model to include the cardinality constraints which limit the number of assets held in the portfolio and bounds on the proportion of an asset held (if any is held). We present our formulation based on the Markowitz MV model for rebalancing an existing portfolio subject to both fixed and variable transaction cost (the fee associated with trading). We determine and demonstrate the differences that arise in the shape of the trading portfolio and efficient frontiers when subject to non-cardinality and cardinality constrained transaction cost models. We apply our flexible heuristic algorithms of genetic algorithm, tabu search and simulated annealing to both the cardinality constrained and transaction cost models to solve problems using data from seven real world market indices. We show that by incorporating optimization into the generation of valid portfolios leads to good quality solutions in acceptable computational time. We illustrate this on problems from literature as well as on our own larger data sets.

Acknowledgements

My journey through the PhD project was a enjoyable and painstaking experience. Every step of the way was filled with the intrigues of emotional, intellectual and financial natures. Through it all, the ups and downs, there is an instructive awareness that the successful accomplishment on the goal was made possible by a team of hardworking individuals who worked tirelessly with me and on my behalf to make this childhood dream a reality.

While words are not enough to express my profound gratitude to all involved, it suffices to say many thanks and appreciation for your efforts.

First of all, I would like thank my supervisors, Dr. Cormac Lucas and Professor John Beasley for their tireless efforts in taking me through this process, from start to finish. They offered me as much advice as you can get, with patient supervision, guiding me in the right direction; without which this thesis would not be possible.

My sincere thanks to my employer, the college of the Bahamas for providing this invaluable opportunity to study for a doctorate degree.

To my friends and supporters, especially Yvonne Rolle, Zendal Forbes, Eric Rolle and the Unity Fellowship Prayer Band, I thank you for all your encouragement.

Many thanks and appreciation to my family for unselfishly bearing with the time away from them and the resources away from other family commitments to pursue this life goal.

To my only other sibling, Claudette Thomas thank you for believing.

To my mother, Thelma Rose you are the wings beneath my wings.

To my kids, Raymond and Anne-Marie this thesis is for both of you. Never stop trying don't ever give up.

To my life partner, my best friend, my proof-reader, my protector, my husband Raymond Oriakhi, if it were not for your ever continuing encouragement, the very thought of obtaining a PhD would probably still be a dream.

Maria Woodside-Oriakhi
London, January 2011

Contents

1 Introduction **1**

1.1 Research Objective . . . 1

1.2 Research Issues . . . 1

1.3 Thesis Structure . . . 2

2 Literature Review **4**

2.1 Introduction . . . 4

2.2 The Markowitz Mean-Variance Model . . . 5

2.2.1 The Efficient Frontier . . . 7

2.3 Alternatives to the Markowitz Mean-Variance Model . . . 10

2.3.1 Developments to the MV Model . . . 10

2.3.2 Departures from the MV Model . . . 12

2.4 The Buy-in Threshold and Cardinality Constraint . . . 15

2.4.1 Buy-in Threshold . . . 16

2.4.2 Cardinality Constraint . . . 17

2.5 Heuristic Algorithms . . . 19

2.5.1 Genetic Algorithms 20
2.5.2 Tabu Search 24
2.5.3 Simulated Annealing 26
2.5.4 Heuristic Algorithms for Cardinality Constrained Portfolio Optimisation 28
2.6 Portfolio Theory with Transaction Cost 30
2.7 Summary 32

3 Transaction Cost: Optimal Solutions 34

3.1 Introduction 34
3.2 Formulation 35
3.2.1 Notation 35
3.2.2 Constraints 38
3.2.3 The Objective Function 40
3.2.4 The Complete Transaction Cost Formulation 41
3.3 Cardinality Extension 43
3.3.1 The Complete Transaction Cost Cardinality Formulation 45
3.4 Computational Considerations 47
3.5 Data Sets 49
3.6 Transaction Cost Frontiers 49
3.6.1 Comparing Transaction Cost to the Unconstrained Problem 50
3.6.2 Fixed and Variable Transaction Cost Efficient Frontier 50
3.6.3 Four Asset Example 51

3.7 Non-Cardinality Constrained Transaction Cost Frontiers 58
3.8 Cardinality Constrained Transaction Cost Frontiers 68
3.9 Summary 69

4 Heuristic Algorithms for the CCEF 77

4.1 Introduction 77
4.2 Subset Optimisation 78
4.3 Heuristic Algorithms 80
4.3.1 A Genetic Algorithm Heuristic 80
4.3.2 A Genetic Algorithm Heuristic Example 84
4.3.3 A Tabu Search Heuristic Algorithm 86
4.3.4 A Simulated Annealing Heuristic Algorithm 88
4.4 Computational Results 90
4.4.1 Data Sets 90
4.4.2 CPLEX Results 91
4.4.3 Percentage Deviation Calculations 92
4.4.4 Heuristic Algorithms' Parameter Values 93
4.4.5 Heuristic Algorithms' Results for the CCEF 96
4.5 Summary 105

5 Transaction Cost: Heuristic Algorithms 107

5.1 Introduction 107
5.2 Subset Optimisation 108

5.3 Heuristic Algorithms 110

5.4 Data Sets 111

5.5 Non-Cardinality Constrained Transaction Cost Heuristic Algorithms' Results . 112

5.5.1 Genetic Algorithm 112

5.5.2 Tabu Search 118

5.5.3 Simulated Annealing 120

5.5.4 Pooled Heuristic Algorithms 129

5.6 Cardinality Constrained Transaction Cost Heuristic Algorithms' Results 130

5.6.1 Genetic Algorithm 139

5.6.2 Tabu Search 142

5.6.3 Simulated Annealing 148

5.6.4 Pooled Heuristic Algorithms 152

5.7 Summary 156

6 Conclusion **166**

6.1 Summary 166

6.2 Contribution to Knowledge 168

6.3 Future Work 169

7 Appendix **171**

List of Figures

2.1 A typical Trading Portfolio Frontier 8

2.2 A typical Trading Portfolio Frontier and UEF 8

2.3 The UEF traced from Isoprofit Lines 10

2.4 The UEF and CCEF for a Four Asset Example 18

3.1 An example of the TCEF for the Hang Seng Market Index 52

3.2 A Four Asset Example Trading Portfolio Frontier for $V^{new} = 0$ 54

3.3 A Four Asset Example Trading Portfolio Efficient Frontier for $V^{new} = 0$ 54

3.4 A Four Asset Example Trading Portfolio Frontier for $V^{new} > 0$ 55

3.5 A Four Asset Example Trading Portfolio Efficient Frontier for $V^{new} > 0$ 55

3.6 A Four Asset Example Trading Portfolio Frontier for $V^{new} < 0$ 57

3.7 A Four Asset Example Trading Portfolio Efficient Frontier for $V^{new} < 0$ 57

3.8 A Four Asset Example with $V^{new} = 0$, $V^{new} > 0$ and $V^{new} < 0$ 58

3.9 Hang Seng Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier 60

3.10 Hang Seng Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier . . . 61

3.11 DAX 100 Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier . . . 62

3.12 DAX 100 Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier . . . 62

3.13 FTSE 100 Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier . . . 63

3.14 FTSE 100 Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier . . . 63

3.15 S&P 100 Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier . . . 64

3.16 S&P 100 Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier . . . 64

3.17 Nikkei 225 Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier . . . 65

3.18 Nikkei 225 Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier . . . 65

3.19 S&P 500 Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier . . . 66

3.20 S&P 500 Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier . . . 66

3.21 Hang Seng Transaction Cost Cardinality Constrained Trading Portfolio Frontier 70

3.22 Hang Seng Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier . . . 71

3.23 DAX 100 Transaction Cost Cardinality Constrained Trading Portfolio Frontier 72

3.24 DAX 100 Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 72

3.25 FTSE 100 Transaction Cost Cardinality Constrained Trading Portfolio Frontier 73

3.26 FTSE 100 Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 73

3.27 S&P 100 Transaction Cost Cardinality Constrained Trading Portfolio Frontier . 74

3.28 S&P 100 Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 74

3.29 Nikkei 225 Transaction Cost Cardinality Constrained Trading Portfolio Frontier 75

3.30 Nikkei 225 Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 75

4.1 GA Heuristic Algorithm Crossover and Mutation Example 85

4.2 DAX 100 Trade-off Curves for $K = 2$ and $K = 4$ 98

4.3 DAX 100 Trade-off Curves for $K = 3$ and $K = 5$ 99

4.4 Russell 2000 GA, TS and SA Heuristic Algorithms' Results 101

4.5 Russell 2000 pooled GA, TS and SA Heuristic Algorithms' Results 102

5.1 Hang Seng GA Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier 115

5.2 Hang Seng GA Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier 116

5.3 S&P 100 GA Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier 117

5.4 S&P 100 GA Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier 117

5.5 FTSE 100 TS Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier 122

5.6 FTSE 100 TS Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier 123

5.7 Nikkei 225 TS Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier 124

5.8 Nikkei 225 TS Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier 124

5.9 S&P 500 SA Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier 127

5.10 S&P 500 SA Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier 128

5.11 Hang Seng Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier 132

5.12 Hang Seng Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier 133

5.13 DAX 100 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier 134

5.14 DAX 100 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier 134

5.15 FTSE 100 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier 135

5.16 FTSE 100 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier 135

5.17 S&P 100 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier 136

5.18 S&P 100 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier 136

5.19 Nikkei 225 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier 137

5.20 Nikkei 225 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier 137

5.21 S&P 500 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier 138

5.22 S&P 500 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier 138

5.23 Hang Seng GA Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Frontier 143

5.24 Hang Seng GA Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 144

5.25 DAX 100 GA Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Frontier 145

5.26 DAX 100 GA Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 145

5.27 FTSE 100 TS Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Frontier 149

5.28 FTSE 100 TS Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 150

5.29 Nikkei 225 TS Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Frontier 151

5.30 Nikkei 225 TS Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 151

5.31 S&P 100 SA Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Frontier 154

5.32 S&P 100 SA Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 155

5.33 Hang Seng Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Frontier 158

5.34 Hang Seng Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 159

5.35 DAX 100 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Frontier 160

5.36 DAX 100 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 160

5.37 FTSE 100 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Frontier 161

5.38 FTSE 100 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 161

5.39 S&P 100 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Frontier 162

5.40 S&P 100 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 162

5.41 Nikkei 225 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Frontier 163

5.42 Nikkei 225 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 163

5.43 S&P 500 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Frontier 164

5.44 S&P 500 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier 164

List of Tables

3.1 A Four Asset Example 51

3.2 Non-Cardinality Constrained Transaction Cost Trading Portfolio Frontier Times 59

3.3 Percentage Error Results for the Transaction Cost Non-Cardinality Constrained Efficient Frontier 67

3.4 Percentage Error Results for the Transaction Cost Cardinality Constrained Efficient Frontier 67

3.5 Cardinality Constrained Transaction Cost Trading Portfolio Frontier Times . . 68

4.1 GA Heuristic Algorithm Selection Example 84

4.2 Computation time (seconds) for the DAX CCEF using CPLEX 91

4.3 GA Heuristic Algorithm Parameter Test Values 95

4.4 TS Heuristic Algorithm Parameter Test Values 95

4.5 SA Heuristic Algorithm Parameter Test Values 96

4.6 Heuristic Algorithms' Results for the CCEF 100

4.7 Pooled Heuristic Algorithms' Results for the CCEF 104

5.1 GA Heuristic Algorithm Computational Times for the Non-Cardinality Constrained Transaction Cost Model 113

5.2 GA Percentage Error Results for the Transaction Cost Non-Cardinality Constrained Efficient Frontier 114

5.3 GA Percentage Error Difference for the Transaction Cost Non-Cardinality Constrained Frontiers 114

5.4 TS Heuristic Algorithm Computational Times for the Non-Cardinality Constrained Transaction Cost Model 119

5.5 TS Percentage Error Results for the Transaction Cost Non-Cardinality Constrained Efficient Frontier 121

5.6 TS Percentage Error Difference for the Transaction Cost Non-Cardinality Constrained Frontiers 121

5.7 SA Heuristic Algorithm Computational Times for the Non-Cardinality Constrained Transaction Cost Model 125

5.8 SA Percentage Error Results for the Transaction Cost Non-Cardinality Constrained Efficient Frontier 126

5.9 SA Percentage Error Difference for the Transaction Cost Non-Cardinality Constrained Frontiers 126

5.10 Pooled Heuristic Algorithms' Results for the Transaction Cost Non-Cardinality Constrained Frontiers 131

5.11 GA Heuristic Algorithm Computational Times for the Cardinality Constrained Transaction Cost Model 140

5.12 GA Percentage Error Results for the Transaction Cost Cardinality Constrained Efficient Frontier 141

5.13 GA Percentage Error Difference for the Transaction Cost Cardinality Constrained Frontiers 141

5.14 TS Heuristic Algorithm Computational Times for the Cardinality Constrained Transaction Cost Model 142

5.15 TS Percentage Error Results for the Transaction Cost Cardinality Constrained Efficient Frontier 147

5.16 TS Percentage Error Difference for the Transaction Cost Cardinality Constrained Frontiers 147

5.17 SA Heuristic Algorithm Computational Times for the Cardinality Constrained Transaction Cost Model 148

5.18 SA Percentage Error Results for the Transaction Cost Cardinality Constrained Efficient Frontier 153

5.19 SA Percentage Error Difference for the Transaction Cost Cardinality Constrained Frontiers 153

5.20 Pooled Heuristic Algorithms' Results for the Transaction Cost Cardinality Constrained Frontiers 157

7.1 Hang Seng parameter values 172

Chapter 1

Introduction

1.1 Research Objective

Fund (portfolio) management deals with creating, adjusting and implementing an investment strategy for financial assets of which an integral part is the selection of a portfolio of assets. Fund managers strive to provide returns for investors by assembling a portfolio of financial instruments that best represent the investor's risk level and yield (return) expectation. Over time, fund managers are under fiduciary obligation to rebalance such portfolios to reflect current and ongoing changes in a fast-paced financial market. They also where necessary (or possible), create new portfolios taking advantage of market situations (or available cash) to provide investors with appropriate returns.

The objective of this thesis is to contribute to the development of efficient and effective portfolio selection algorithms and their application to portfolio optimisation problems involving cardinality constraints and transaction cost.

1.2 Research Issues

The Nobel Prize (for Economics) winning work of Markowitz [48] set up a clear quantitative framework for the selection of a portfolio, summarising the process of portfolio selection as an allocation of resources so as to tradeoff expected return and risk. Through the use of statistical measurements of expectation and variance of return (variance being equated to risk), Markowitz described the benefit and risk associated with an investment.

His approach formulates and solves a parametric quadratic program and has become the core decision model of many portfolio analytic and planning systems in constructing efficient frontiers, which can be viewed as the set of Pareto optimal (expected return, variance of return) combinations under conditions of uncertainty. The beauty of this simplistic unconstrained risk return model is that it is capable of being extended to capture market realisms such as cardinality constraints (a fixed number of assets) and transaction cost (fees associated with trading).

In this thesis, we determine optimal solutions for portfolio optimisation problems with transaction cost. Our transaction cost model is an extension of the standard Markowitz model. We model problems involving fixed and variable transaction cost both where there is no cardinality constraint and where there are cardinality constraints.

Additionally, in this thesis three heuristic techniques are used to solve the portfolio optimisation problems involving cardinality constraints and transaction cost. These are genetic algorithm, tabu search and simulated annealing. Heuristic algorithms have been successfully applied by many researchers and are attractive because they are independent of the objective function. This means that a portfolio manager can replace the normal mean variance objective function with whichever function he considers relevant for the universe of assets he is considering.

1.3 Thesis Structure

The structure of this thesis is as follows. In Chapter 2 we present a literature review of modern portfolio theory including work involving cardinality constraints and transaction cost.

In Chapter 3, we give our formulation of the portfolio optimisation problem involving transaction cost and extend it to include cardinality constraints. We investigate the shape of the transaction cost efficient frontier and consider whether discontinuities arise in the transaction cost efficient frontier from either fixed cost or a cash investment. We further consider the trading portfolio and efficient frontiers for the transaction cost optimisation problem with and without cardinality constraints. We solve these problems using A Mathematical Programming Language (AMPL, a modeling language) and CPLEX involving publicly available data sets drawn from six major market indices. We then present graphical illustrations for the frontiers, give computational times and compare the models (those with and those without cardinality constraints).

In Chapter 4, we present the model for the portfolio optimisation problem involving cardinality constraints. We develop a subset optimisation problem (a very controlled and simple partial optimisation which we apply to all potential portfolios) to solve the model. Then, we present a genetic algorithm, tabu search and simulated annealing heuristics. We continue by applying our heuristic algorithms to the cardinality constrained optimisation problem (involving data sets drawn from seven major market indices) and compare them to previous work in the literature.

In Chapter 5, we apply our heuristic algorithms (described in Chapter 4) of genetic algorithm, tabu search and simulated annealing to the portfolio optimisation problem involving transaction cost. For the transaction cost model without cardinality constraints and transaction cost model with cardinality constraints, we develop a subset optimisation problem (as was done in Chapter 4) to solve the model. Then, in each case we present graphical illustrations of the frontiers (portfolio and efficient), give percentage error results for the efficient frontiers produced from the unconstrained efficient frontier and compare heuristic algorithm results for the transaction cost models with and without cardinality constraints.

In Chapter 6, we conclude the thesis presenting a summary of the thesis, our contribution to knowledge and future directions.

Chapter 2

Literature Review

2.1 Introduction

Portfolio theory is concerned with the allocation of an individual's wealth among the various available risky assets. The pioneering work of portfolio theory was developed by Harry Markowitz [48, 49] in 1952 and 1956. His work suggested that, for any given level of risk, the rational investor would select the portfolio with maximum expected return, and for any given level of expected return, the rational investor would select the portfolio with minimum risk. The model assumes a perfect market without transaction cost or taxes where short selling is not permitted, but securities are infinitely divisible and can therefore be traded in any non-negative fraction. Since the development of the Markowitz model, it has become the core decision engine of many portfolio analytic and planning systems.

In this Chapter, we describe the history of portfolio theory since Markowitz. Then, we present and discuss related research in the literature for discrete practical constraints in portfolio theory, heuristic algorithms and transaction cost. The majority of the studies in these areas, focus on discrete constraints or discrete constraints with heuristic algorithms but most do not focus on transaction cost. Papers that focus on discrete constraints tend to focus on exact solutions with problems involving up to 500 assets. Research in the area of heuristic algorithms normally use one algorithmic approach, with the majority of papers using five test problems or less. The papers involving transaction cost tend to use a modified quadratic programming model, with researchers adopting different mathematical models.

This thesis will consider creating the exact solution of portfolios with discrete constraints for seven test problems involving up to 1318 assets. We will also present the transaction cost model using a formulation that involves bounds on buying and selling of assets and constraints on assets held, the budget and transaction cost. Our formulation considers creating a portfolio from cash and rebalancing an existing portfolio.

We organize this Chapter in the following way. The Mean-Variance (MV) model of Markowitz is considered in Section 2.2. Alternatives to the Markowitz Mean-Variance model are in Section 2.3. The discrete extensions to the MV model of buy-in threshold and cardinality constraint are presented in Section 2.4. Heuristic algorithms in portfolio theory are examined in Section 2.5, while transaction costs in portfolio theory is considered in Section 2.6. The Chapter is concluded with a summary in Section 2.7.

2.2 The Markowitz Mean-Variance Model

Markowitz set up a clear quantitative framework for the selection of a portfolio, summarizing the process of portfolio selection as an allocation of resources depending on expected return and risk. Through the use of statistical measurements of expectation and variance of return, Markowitz describes the benefit and risk associated with an investment. The objective is either to minimise the risk of the portfolio for a given level of return, or to maximize the expected level of return for a given level of risk. His model justifies the observable phenomenon of *diversification* in investment. He defines the solutions of this single period static portfolio planning model as *efficient.* By formulating and solving a parametric quadratic program (QP), Markowitz determined the efficient portfolio from the investment opportunity set.

Let:

N	be the number of assets (securities) available for an investment,
w_i	be the fraction $(0 \leqslant w_i \leqslant 1)$ held of an asset i $(i = 1, \ldots, N)$,

μ_i	be the expected return of asset i ($i = 1, \ldots, N$),
σ_{ij}	be the covariance between the return of asset i and asset j ($i = 1, \ldots, N$; $j = 1, \ldots, N$), and
R	be the desired level of expected return for the portfolio.

Given these variables and parameters, Markowitz captures the risk averse investor preferences for portfolio mean and variance. The Markowitz MV model is:

$$\text{Minimise} \sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j \sigma_{ij} \tag{2.1}$$

subject to

$$\sum_{i=1}^{N} w_i \mu_i = R \tag{2.2}$$

$$\sum_{i=1}^{N} w_i = 1 \tag{2.3}$$

$$w_i \geqslant 0 \qquad\qquad i = 1, \ldots, N. \tag{2.4}$$

Equation (2.1), the portfolio variance (σ^2), involves the covariance matrix minimising the volatility associated with the portfolio. This equation is sometimes written as $\sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j \rho_{ij} s_i s_j$. When expressed this way in terms of an asset's standard deviation, s_i ($i = 1, \ldots, N$), it makes use of the correlation coefficient, $\rho_{ij} = \sigma_{ij}/(s_i s_j)$ where $|\rho_{ij}| \leqslant 1$. Equation (2.2) is the expected rate of return of the portfolio; it is found by taking the weighted sum of the individual rates of return. Equation (2.3) is the budget constraint, i.e. the investor invests the entire capital available, while equation (2.4) is the non-negativity constraint, prohibiting short selling from taking place.

Markowitz MV model is based on several assumptions regarding an investor's behaviour. These assumptions adopted from Reilly and Brown [58] are listed below.

1. An investor will think about each investment alternative as being represented by a

particular probability distribution of expected returns over some period.

2. An investor maximizes one-period expected utility and the utility curve demonstrates diminishing marginal utility of wealth.

3. An investor equates the risk of the portfolio to the variability of expected returns.

4. An investor bases decisions solely on expected return and risk, therefore their utility curves are a function of the expected variance (or standard deviation) of returns and expected return only.

5. An investor prefers higher returns to lower returns, for a given level of risk. Similarly, for a given expected return level, an investor prefers less risk to more risk.

2.2.1 The Efficient Frontier

The set of points that correspond to the least risk portfolios at all possible return levels is called the feasible set. The feasible set is a parabola and is also called the minimum variance set or a trading portfolio frontier. On the trading portfolio frontier there exists a point with the least variance termed the minimum variance point. In Figure 2.1 we illustrate a trading portfolio frontier.

When considering this Figure, it is clear to see that all portfolios on the trading portfolio frontier that lie below the minimum variance point are dominated (since for all portfolios on that line segment there exists another portfolio with the same risk which produces greater return). By eliminating all dominated portfolios from the trading portfolio frontier, Markowitz determines the *efficient frontier*, that is the set of (undominated) portfolios found by minimising variance as the desired return is varied. Throughout this thesis we refer to these non-dominated portfolios as the unconstrained efficient frontier (UEF). Figure 2.2 illustrates the trading portfolio frontier and the UEF.

In practice, it is common to model the MV trade-off using a parameter λ, $0 \leqslant \lambda \leqslant 1$, as stated below:

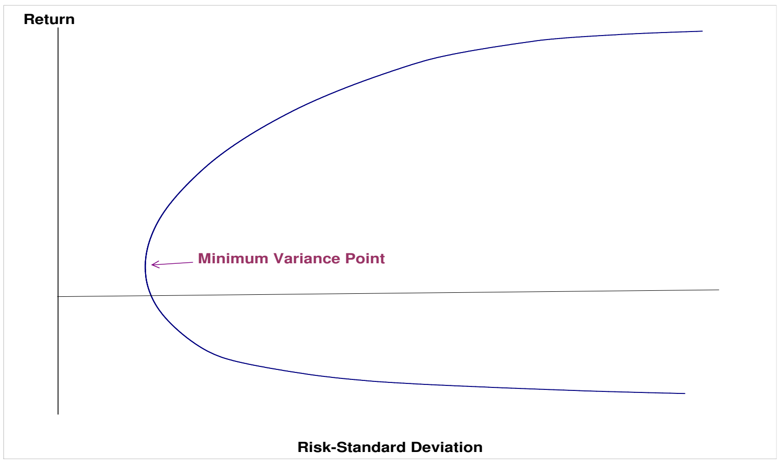

Figure 2.1: A typical Trading Portfolio Frontier

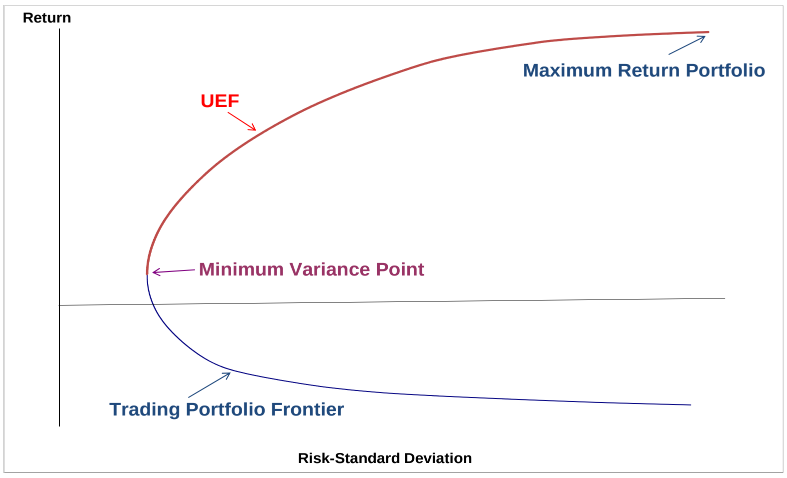

Figure 2.2: A typical Trading Portfolio Frontier and UEF

$$\text{Minimise } \lambda \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \sigma_{ij} - (1-\lambda)\sum_{i=1}^{N} w_i \mu_i \tag{2.5}$$

subject to

$$\sum_{i=1}^{N} w_i = 1 \tag{2.6}$$

$$w_i \geqslant 0 \qquad i = 1, \ldots, N. \tag{2.7}$$

In equation (2.5) the value $\lambda = 1$ minimises the variance of a portfolio (irrespective of the return involved). On the other hand $\lambda = 0$ corresponds to maximising the expected return of a portfolio (irrespective of the risk involved).

As with Markowitz's model above (minimise equation (2.1) subject to equations (2.2)-(2.4)) the UEF can be traced out by varying the value of λ between the high risk and low return portfolios and repeatedly solving equation (2.5) subject to equations (2.6) and (2.7). To see how this is possible we consider a particular value of λ, for example $\lambda = \frac{1}{3}$. Then equation (2.5) becomes, minimise $\frac{1}{3}$risk - $\frac{2}{3}$return. Considering Figure 2.2 we could plot a series of isoprofit lines of the form $\frac{1}{3}$risk - $\frac{2}{3}$return= Z and choose the minimum value of Z. Rearranging these isoprofit lines for return yields, return =$\frac{1}{2}$risk -$\frac{3}{2}Z$. Therefore the slope of the line is $\frac{1}{2}$ and the y-intercept is given by $-\frac{3}{2}Z$. Hence, minimising Z is the same as choosing amongst these lines of fixed slope the one with the maximum y-intercept which is only achieved at the unique point where the line of slope $\frac{1}{2}$ is a tangent to the efficient frontier.

In Figure 2.3 we illustrate this point. At $risk_1$, the Z value is minimised at Z_{3*} and at $risk_2$, the Z value is minimised at Z_{C*}. In each of those cases, Z_{3*} and Z_{C*} offer the highest return value for the given level of risk.

Thus, by varying the value of λ between the high risk and low return portfolios and repeatedly solving equation (2.5) subject to equations (2.6) and (2.7) we would obtain the same efficient frontier as minimising equation (2.1) subject to equations (2.2)-(2.4) for varying values of R.

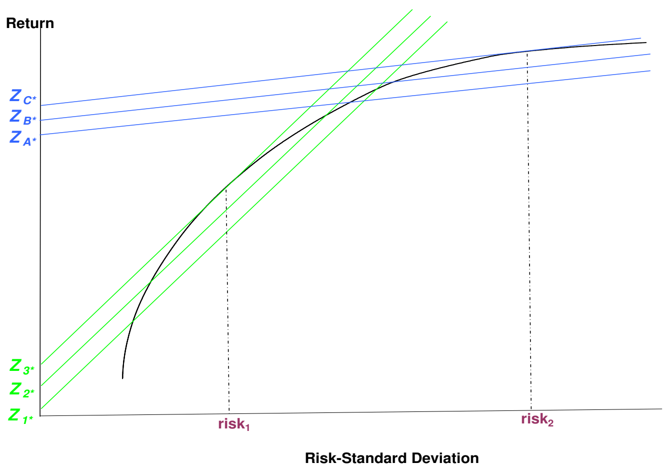

Figure 2.3: The UEF traced from Isoprofit Lines

2.3 Alternatives to the Markowitz Mean-Variance Model

The alternatives to the Markowitz Mean-Variance model are in two categories: those that build on the work of Markowitz (Section 2.3.1) and those which are a departure from Markowitz's work (Section 2.3.2). In this Section we examine these two categories.

2.3.1 Developments to the MV Model

Modern portfolio theory has developed in tandem with simplifications to the QP required by MV analysis. These simplifications centre around linearising the quadratic objective function or reducing the number of parameters to be estimated. Both approaches involve either an approximation or a decomposition of the covariance matrix.

Starting in the 1960s, numerous researchers built on the work of Markowitz. Sharpe [61] proposed a linear programming (LP) formulation known as the single-index, or market model, as a sufficient model of covariance. In this model, he supposed that there were n assets, which are indexed by i, and the assets have a rate of return for the period, r_i $(i = 1, \ldots, N)$

and return on the market index for the same period is R^*. The expected excess return on asset i $(i = 1, \ldots, N)$ due to firm specific factors is given by α_i, the sensitivity of asset i $(i = 1, \ldots, N)$ to market movements is given by β_i, and for the period there are random quantities representing the fluctuations in return for an asset i called errors represented by e_i. Hence, the return of the single-index model is:

$$r_i = \alpha_i + \beta_i R^* + e_i \; i = 1, \ldots, N$$

This single–index model is a diagonal model that reduces the computations required to determine the covariance. In Markowitz's model the covariance of the securities within a portfolio must be calculated using historical returns, and the covariance of each possible pair of securities in the portfolio must be calculated independently. In the single–index model the covariance of assets i and j $(i = 1, \ldots, N; j = 1, \ldots, N)$ can be found by multiplying the betas of those assets and the market variance, $\sigma_{ij} = \beta_i \beta_j \sigma_m^2$, ($\sigma_m^2$ is the market variance). With this equation, only the betas of the individual securities and the market variance need to be estimated to calculate covariance. Thus, the index model reduces the number of calculations that would otherwise have to be made for a large market.

Sharpe [62], Lintner [43] and Mossin [52] independently developed the capital asset pricing model (CAPM) which decomposes a portfolio's risk into specific and systematic risk. Specific risk represents the component of an asset's return which is uncorrelated with general market moves. It is the risk that is unique to an individual asset. Systematic risk is the risk of holding the market portfolio. This model considers the excess return on an asset as relating to the excess return on the market index. It introduces the risk–free interest rate for the period, r_f. Thus, the return is described as:

$$E(r_i) = r_f + \beta_i E(R^* - r_f)$$

where E denotes an expectation. This equation states that the asset's expected return equals the risk–free interest rate plus the asset's beta times the expected return of the market index minus the risk–free rate. Thus, CAPM estimates an asset's return according to it's contribution to market risk.

Rosenberg [59] presented a multi-factor model that incorporated industry and other factors. It introduces to the period, R_j, the return on the index j (where there are a total of J indexes), and β_{ij}, the sensitivity of asset i ($i = 1, \ldots, N$) to index j. The return on the multi-index model is therefore:

$$r_i = \alpha_i + \sum_{j=1}^{J} \beta_{ij} R_j + e_i \; i = 1, \ldots, N.$$

Ross [60] using factor analysis, developed the arbitrage pricing theory (APT), which is a multi-index equilibrium model. The APT model says asset prices are mainly driven by several factor prices that have some fundamental and plausible relationship to the underlying factors.

Index or factor models allow a simplification of the underlying QP. The covariance matrix can be expressed in a diagonal form and hence a linear approximation of the quadratic objective function can be obtained (see Sharpe [63]).

2.3.2 Departures from the MV Model

Many researchers question whether the variance is a good measure of the risk for a portfolio. Consequently, a number of alternative measures of risk have been proposed and investigated. In many cases these measures are linear, leading to a corresponding simplification in the computation.

Konno and Yamazaki [36] show that the mean absolute deviation (MAD) model is equivalent to Markowitz MV model, under the assumption of multivariate normal returns (i.e. under this assumption the sum of the absolute deviations of portfolio returns about the mean is equivalent to the minimisation of the variance). For this model let r_{it} denote the return of asset i ($i = 1, \ldots, N$) at time t (assumed to be available through the historical data or from some future projection). Therefore, the return of MAD model is approximated as:

$$r_i \approx \frac{1}{T}\sum_{i=1}^{N}\sum_{t=1}^{T} r_{it}w_i \tag{2.8}$$

and the risk is approximated by

$$\frac{1}{T}\sum_{t=1}^{T}\left|\sum_{i=1}^{N}(r_{it}-\mu_i)w_i\right|. \tag{2.9}$$

Konno and Yamazaki [36] claim that an advantage to this model is that it limits the number of assets held, allowing control of transaction cost. A generalization of the MAD model can be found in Worzel *et al.* [70]. Konno *et al.* [37] extend the MAD approach to include skewness in the objective function under possible asymmetry of returns.

Linear programming based heuristics are used by Speranza [66], considering the negative semi–MAD model with cardinality constraints. In that model, the risk associated with the portfolio is measured by the mean absolute deviation of the return below average instead of by variance. This negative semi–MAD model is extended in Kellerer, Mansini and Speranza [33] incorporating fixed costs and then in Mansini and Speranza [45] to incorporate roundlots (a discrete number of assets taken as the basic unit of investment).

Multi-objective goal programming approaches have also been proposed. Lee [40] first introduced Lexicographic Goal Programming. This model separates the objective into a number of priority levels where the satisfaction of goals with higher priority is regarded as infinitely more important than the satisfaction of lower level goals. Tamiz *et al* [69] using a factor model of stock returns, measure the risk of a portfolio as the sum of absolute deviations of the portfolio's factor sensitivities from those of a specified target. To force diversification of the stock specific risks, they apply a constraint on the holdings allowed in each sector of industry.

To show a simple case of this model, let W_1 denote the positive penalty weight associated with shortfalls in portfolio return below the target, W_2 denote the positive penalty weight associated with excess portfolio risk in relation to the target, $Risk_p$ denotes risk associated with the portfolio, and $Risk_i$ denotes the risk associated with asset i $(i = 1, \ldots, N)$. Then the decision variables are n_1 the negative deviation from the target level of portfolio return,

n_2 the negative deviation from the target risk level, p_1 the positive deviation from the target level of portfolio return, and p_2 the positive deviation from the target risk level.

The WGP model is therefore

$$\text{Minimise } W_1 n_1 + W_2 p_2 \tag{2.10}$$

subject to

$$\sum_{i=1}^{N} w_i \mu_i + n_1 - p_1 = \rho \tag{2.11}$$

$$\sum_{i=1}^{N} Risk_i w_i + n_{2p} - p_{2p} = Risk_p \qquad p = 1, \ldots, P \tag{2.12}$$

$$\sum_{i=1}^{N} w_i = 1 \tag{2.13}$$

$$n_1,\, n_2,\, p_1,\, p_2 \geqslant 0 \tag{2.14}$$

$$w_i \geqslant 0 \qquad i = 1, \ldots, N. \tag{2.15}$$

Young [74] employs a minimax investment rule measuring risk as the minimum return (maximum loss) that the portfolio would have achieved over all of the past observation periods. The minimum return that could have incurred in the past is used as the risk measure. To present the model we introduce the variable M_p which represents the minimum return achieved by the portfolio over all observations periods. The Minimax model is therefore:

$$\text{Maximise } M_p \tag{2.16}$$

subject to

$$\sum_{i=1}^{N} w_i r_{it} \geqslant M_p \qquad t = 1, \ldots, T \tag{2.17}$$

$$\sum_{i=1}^{N} w_i \mu_i = \rho \tag{2.18}$$

$$\sum_{i=1}^{N} w_i = 1 \tag{2.19}$$

$$w_i \geqslant 0 \qquad i = 1, \ldots, N \tag{2.20}$$

Young [74] also proposes an alternative formulation to this model that maximises the expected return for the portfolio subject to a given lower bound on the portfolio return for every observation period.

Risk measures concerned with the left tails of the distributions (the extremely unfavorable outcomes) have also been studied. The most widely used of these distributions is Value–at–Risk (VaR). VaR leads to non-convex which can be computationally intractible therefore CVaR (conditional value–at–risk) which controls the magnitude of losses beyond VaR and is easy to optimise is theoretically attractive. CVaR measures the expected loss corresponding to a number of worst cases, depending on the chosen confidence interval. Roman *et al.* [56] propose a model for portfolio selection which uses both variance and CVar for the decision making process.

2.4 The Buy-in Threshold and Cardinality Constraint

A salient feature of financial portfolios is that any additionally included securities might contribute to the diversification of risk while increasing the expected return. This results in an investor seeking an optimal ratio between risk and risk premium (the return in excess of the risk-free rate of return that an investment is expected to yield), within the Markowitz framework, by seeking to include as many different risky assets as possible. However, many investors prefer portfolios with a limited number of assets and Maringer [46] shows that most diversification can be achieved without using all of the assets in the market index. Hence, to capture these realisms of portfolio planning a number of discrete restrictions have been considered by different researchers. In this Section, we model two of them: buy-in threshold in Section 2.4.1 and the cardinality constraint in Section 2.4.2.

2.4.1 Buy-in Threshold

A buy-in threshold sets limits on the amount of capital to be invested in each asset and avoids small investments in an asset. This means the portfolio weights behave as semi–continuous variables (Beale and Forest [5]) and are modeled using finite variable upper and lower bounds, l_i and u_i respectively, where we must have $0 \leqslant l_i \leqslant u_i \leqslant 1$ associated with each asset i $(i = 1, \ldots, N)$. In reality it can be meaningful to place bounds on holdings because of institutional restrictions, to reduce unrealistic trades, to limit exposure of the portfolio to asset i $(i = 1, \ldots, N)$ or to control the transaction cost. Along with these limits, a decision variable, δ_i, is incorporated transforming the QP to a quadratic mixed-integer programming (QMIP) problem that is NP-hard (Moral-Escudero *et al.*, [51]).

Let

$$\delta_i = \begin{cases} 1 & \text{if any of asset } i \ (i = 1, \ldots, N) \text{ is held,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.21}$$

The buy-in threshold is represented by

$$l_i\delta_i \leqslant w_i \leqslant u_i\delta_i \quad i = 1, \ldots, N. \tag{2.22}$$

Equations (2.21) and (2.22), make certain that if an asset is not invested in, $\delta_i = 0$, the resulting weight, w_i, $(i = 1, \ldots, N)$ is zero. If an asset is held, $\delta_i = 1$, then its weight must lie between the upper and lower limits, $l_i \leqslant w_i \leqslant u_i$ $(i = 1, \ldots, N)$.

2.4.2 Cardinality Constraint

The cardinality constraint allows investors to specify the number of unique assets in the portfolio for monitoring purposes to reflect the existence of fees, diversification or regulatory issues. The constraint is achieved by extending the buy-in model to restrict the sum of the binary variables to be equal to K,

$$\sum_{i=1}^{N} \delta_i = K, \tag{2.23}$$

where K represents the number of assets to be in the portfolio. The cardinality constraints and buy-in threshold are intrinsically linked. For example, an lower limit on the buy-in threshold of 10% of the value of a portfolio implies that at least 10 assets can be bought. The cardinality constrained mean-variance model would therefore be to minimise equation (2.1) subject to equations (2.2)-(2.4), and equations (2.21)-(2.23).

The Cardinality Constrained Efficient Frontier

Through the introduction of the buy-in threshold and cardinality constraint, discontinuities are seen in an otherwise continuous efficient frontier. Figure 2.4 shows the $N = 4$ assets example of Chang *et al.* [13] the UEF and the cardinality constrained efficient frontier (CCEF) where $K = 2$. Note that although the UEF is continuous, the CCEF is not.

CCEF in Portfolio Theory

Since the work of Chang *et al.* [13] there have been 94 citations of his work in the Web of Science. In this Chapter we will refer to papers in the literature that have cited Chang *et al.* [13]. The research papers are categorized according to papers within portfolio theory that deal with the cardinality constraint (below), or cardinality constraint and heuristic algorithms (Section 2.5.4), or transaction cost (Section 2.6). We begin with the work of Chang *et al.* [13].

Chang *et al.* [13] illustrate the discontinuous nature of the efficient frontier in the presence

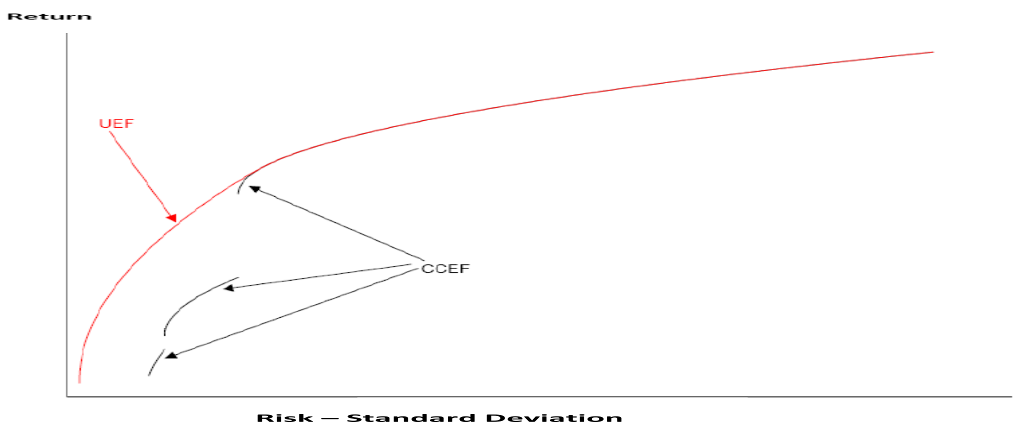

Figure 2.4: The UEF and CCEF for a Four Asset Example

of cardinality restrictions and present three heuristic algorithms based upon genetic algorithm, tabu search and simulated annealing for finding the cardinality constrained efficient frontier. Computational results are presented for five test problems (that are publicly available) involving up to 225 assets.

Li *et al.* [41] present an approach for the exact solution of the cardinality constrained portfolio optimisation problem when the amounts to be invested in each asset must be in specified lots. Any money not invested in assets is invested at a risk–free rate. Computational results are given for one problem involving 30 assets taken from the Hong Kong market.

Corazza *et al.* [16] deal with a quadratic mixed-integer programming problem which they formulate in terms of quantities of asset lots. They provide conditions for the existence of a non-empty mixed-integer feasible set and present some rounding procedures for finding, in a finite number of steps, a feasible mixed-integer solution. The computational experiment involved 100 simulations for portfolios of various sizes (5, 10, 25, 50, 100 and 250 assets).

Shaw *et al.* [64] present a lagrangean relaxation based procedure for the exact solution of the cardinality constrained portfolio optimisation problem. The cardinality constraint (equation (2.23)) is an inequality rather than an equality. In their approach the covariance matrix is decomposed into a diagonal asset risk matrix and a covariance matrix for the F factors adopted. This reduces the size of the quadratic term in the objective from N^2 to F^2. A well-known US equity model has $F = 68$ for example. Computational results are reported for eight test problems involving up to 500 assets. They report that CPLEX (version 8.1) failed to solve any of these problems to proven optimality in four hours of computation. By contrast their approach solved seven of the eight test problems.

Bertsimas and Shioda [8] present an approach for the exact solution of the cardinality constrained portfolio optimisation problem. In their approach the cardinality constraint (equation (2.23)) is an inequality rather than an equality. They use Lemke's pivoting algorithm (Lemke and Howson, [38]) to solve successive subproblems in the search tree. Computational results are presented for their approach as well as for CPLEX on problems involving up to 500 assets. One feature of their results is that for all of the portfolio optimisation test problems considered both their approach and CPLEX (version 8.1) failed to find even a single provably optimal solution within the computational time limit they allow (either two minutes or one hour depending on the size of the problem).

Gulpinar *et al.* [31] introduce the difference of convex functions programming and develop a difference of convex algorithm for the exact solution of the cardinality constrained portfolio problem. They illustrate worst-case portfolio selection with rival risk and return scenario specifications ensuring robustness by considering the optimal strategy in view of multiple rival scenarios and evaluating the portfolio simultaneously with the worst-case scenario. Computational results for $K = 5$ to 20 are compared to those produced by CPLEX (Version 10.1).

2.5 Heuristic Algorithms

The beauty of Markowitz simplistic unconstrained risk-return model is that it is capable of being extended to capture market realism. As the problem increases in size and becomes computationally complex, Section 2.4.1 showed that the introduction of a single binary, δ_i, changes the classical quadratic optimization model to a QMIP which is NP-hard. As a result, many researchers have applied heuristics to study this area. But, what exactly is a heuristic? To get a good description of what a heuristic is we consider the following definitions.

> A "rule of thumb"based on domain knowledge from a particular application, that gives guidance in the solution of a problem...Heuristics may thus be very valuable most of the time but their results or performance cannot be guaranteed. (Oxford Dictionary of Computing, 1996)

> A heuristic technique (or simply heuristic) is a method which seeks good (i.e. near-optimal) solutions at a reasonable computational cost without being able to guarantee optimality, and possibly not feasibility. Unfortunately, it may not even be possible to state how close to optimality a particular heuristic solution is. (Reeves, [57])

Heuristics are often used in tackling many types of complex problems, particularly those of a combinatorial nature. In this Section, we will examine some heuristic methods available namely: genetic algorithm (Section 2.5.1), tabu search (Section 2.5.2) and simulated annealing (Section 2.5.3). This will be followed by a review of heuristics algorithms for portfolio optimisation subject to cardinality constraint in Section 2.5.4.

2.5.1 Genetic Algorithms

Genetic Algorithms (GA) are a powerful stochastic global search mechanism which mimics the principles of natural selection and genetics. They work with a collection of solutions employing Darwin's principle of "survival of the fittest". At each generation, the quality of the population is expected to be better than the previous generation. The fittest solutions are selected and breed together using operators borrowed from biological evolution. The process leads to the creation of populations of individuals that are better suited to their environment than the individuals that they were created from.

This heuristic having it's theoretical foundations from Holland [32] is widely acknowledged.

Various chapters have been devoted to and books have been written on GA. Examples include Holland [32], Goldberg [27], Michalewicz [50], Glover and Kochenberger [28] and Burke and Graham [11]. Web tutorials focusing on genetic algorithms include
http://www.ai-junkie.com/ga/intro/gat1.html and
http://www.obitko.com/tutorials/genetic-algorithms/.

Genetic Algorithm Operators

In GA, the solutions are represented by their genetic make-up called genotypes. The decision variable of a search problem is encoded into finite-strings referred to as *chromosomes,* for which the standard representation is bits of zeros and ones. These strings are made up of a finite length of alphabets called *genes.* The responsibility for the variation in the hereditary characteristics lies in the hand of *alleles.*

To implement natural selection and evolve good solutions, the chromosomes are evaluated by a fitness criteria. The measure could be an *objective* function as in a computer simulation or a mathematical model, or it can be a *subjective* function in which humans choose better solutions over worse ones.

Genetic Algorithms typically rely on a candidate population of fixed cardinality, which it maintains throughout each iteration. GAs use four main operators of selection, crossover, mutation and replacement. The population experiences gradual changes through repetition of these operators, with stronger fitter genes dominating weaker ones. Each generation is expected to inherit the "good"characteristics from the previous generation.

The operators of GA are detailed below.

1. **Population Initialization.** The method by which individuals are chosen in the population along with it's size are important factors affecting the scalability and performance of the GA heuristic. A small population may lead to premature convergence, substandard solutions and insufficient room for exploring the search space. Conversely, a large population can lead to excessive computational times. Thus, population size leads to a trade-off of between efficiency and effectiveness. Initial populations are normally gen-

erated randomly with the size usually being a user-specified parameter. However, in generating the initial population domain specific knowledge or other information can be incorporated.

2. **The Fitness Function.** The fitness measure determines a candidate solution's relative fitness, which will subsequently be used by the GA to guide the evolution solutions. Fitness is calculated according to the chosen objective function. High fitness increases the chances of being reproduced while, low fitness could ultimately lead to extinction.

3. **Selection.** The selection operator allocates more copies of those with higher fitness values and thus the preference of better solutions to worse ones. Two broad probabilistic methods of selection are fitness proportionate selection and ordinal selection. The fitness proportionate selection includes methods such as roulette-wheel and stochastic universal selection; the ordinal selection methods are those such as tournament selection and truncation selection.

4. **Crossover.** The main purpose of the crossover operator is to select good strings in the population to form the mating pool. This operator combines two or more parental solutions to create offspring. In it simplest form, reproduction can take place by dividing two parents in half and then recombining a half from each parent to create two children. Other ways to merge information from parents include k-point crossover or uniform crossover.

5. **Mutation.** Mutation is an important secondary genetic operator that alters one or more gene values in a chromosome from its initial state. This simple operator performs a random walk in the vicinity of a candidate solution resulting in entirely new gene values being added to the gene pool and helping to prevent the population from stagnating at any local optima. With these new gene values, the genetic algorithm has the opportunity of creating an even better solution than that which was previously possible; it provides an opportunity for diversification and exploration of more of the search space. Mutation occurs during evolution according to a user-definable mutation proba-

bility usually set fairly small to ensure the benefits of selection and crossover are not lost.

6. **Replacement.** The offspring population created by selection, crossover and mutation needs to be introduced to the original parental population. This can take one of the following forms: delete–all, steady–state or steady–state no duplicates. Delete-all replaces all members of the current population and replaces them with the same number of chromosomes that have just been created. Steady-state deletes a specified number of old members and replaces them with an equal number of new members. Then, steady-state no duplicates operates in the same manner as steady state but it has the additional criteria of ensuring each member placed in the population does not have the same chromosomal make up as any other member already present in the population.

The termination criteria for GA is decided by one of the following processes:

- *Generation Number.* The user specifies the maximum number of generations to be run.
- *Evolution Time.* The elapsed evolution time exceeds the user-specified maximum evolution time.
- *Fitness Threshold.* The best fitness in the current population becomes less than the user-specified fitness threshold and the objective is set to minimise the fitness.
- *Fitness Convergence.* In fitness convergence two filters of different lengths are used to smooth the best fitness across the generations. The fitness is assumed as converged and the evolution terminates, when the smoothed best fitness from the long filter is less than a user specified percentage away from the smoothed best fitness from the short filter.
- *Population Convergence.* A population is supposed converged when the average fitness across the current population is less than the user-specified fitness.
- *Gene Convergence.* A user-specified percentage of the genes that make up a chromosome are deemed as converged.

The operators are brought together in the following simple step by step GA procedure laid out below.

```
Choose an initial population
Evaluate the fitness of each individual in the population
Repeat
    Select parents from the population
    Crossover parents to produce offspring's (and mutate the children)
    Evaluate the fitness of the children
    Replace some or all members of population by the children
Until the termination criteria are met.
```

2.5.2 Tabu Search

Tabu Search (TS) is a local search heuristic that uses deterministic control. This heuristic proposed by Glover [29] is widely recognized. Various chapters have been devoted to and books have been written on TS. Examples include Glover [30], Glover and Kochenberger [28] and Burke and Graham [11]. Web tutorials focusing on tabu search include
http://neo.lcc.uma.es/EAWebSite/web/TabuSearch.html and
http://www.ifi.uio.no/infheur/Bakgrunn/Intro_to_TS_Gendreau.htm.

Components of Tabu Search

TS starts with a initial solution and searches the neighbourhood of solutions to find a better solution. Each time a neighbourhood is generated and a new current solution is selected, we call the change from a current solution to a better solution a *move*.

A key concept in TS is that it uses a flexible memory which could be short term or longer term memory.

The **short term memory** guides the search to escape local minima. Within this memory structure the *tabu list* records the history of the search. The list is made of *tabus* (forbidden moves) which are used to prevent reversals and cycling when moving away from local optima through non–improving moves. Each member of the short term memory is given a *tabu tenure* i.e. they are given a fixed number of iterations (usually 7 to 20) to be tabu.

At times, tabus are too powerful. They can prohibit attractive moves even when there is no danger of cycling or they may lead to an overall stagnation of the search process. It is thus necessary to employ algorithmic devices that will allow one to revoke tabus. These devices are called *aspiration criteria.* They override a solution's tabu status, thereby including an otherwise excluded solution in the allowed set. A commonly used aspiration criterion is to allow solutions which produce better objective function values than the currently best known solution.

Sometimes neighbourhoods are very large and one may wish to reduce the number of solutions visited, hence a *candidate list.* A candidate list reduces the number of solutions visited on an iteration isolating regions of the neighbourhood containing moves with desirable attributes.

To bring the TS heuristic algorithm to an end the following termination criteria are commonly used: a user-defined number of iterations (or a fixed amount of CPU time), or after some number of iterations without an improvement in the objective function value, or when the objective reaches a pre-specified threshold value.

The components discussed thus far are brought together in the following simple step by step TS procedure laid out below.

```
Randomly generate an initial solution
Initialise tabu status
Repeat
    Search a set of neighbourhood solutions of the current solution
    Obtain the objective function values of the neighbourhood solutions
    Employ the aspiration criterion (or candidate list)
    Pick the best solution among the non-tabu solutions
    Replace current solution by the best solution
    Update tabu status
Until the termination criteria are met.
```

The **longer term memory** expands the neighbourhood to include solutions not found during a short term memory process. This course of action involves two important components namely: intensification and diversification. The *intensification* strategy is based on modifying choice rules to encourage move combinations and solution features previously found good. They may also initiate a return to attractive regions to search them more thoroughly. Conversely, the *diversification* strategy encourages the search process to examine new regions and to generate solutions that differ in various significant ways from those seen before. From the forgoing, it can be seen how both strategies enhance tabu search within the longer term memory process.

2.5.3 Simulated Annealing

Kirkpatrick *et al* [34] and independently Černý [12] proposed solving combinatorial optimisation problems using the simulated annealing (SA) heuristic algorithm, as a mechanism that performs a stochastic neighborhood search of the solution space. The SA heuristic algorithm was given it's name because it emulates the process of physical annealing with solids. This procedure (physical annealing with solids) heats a crystalline solid and then allows it to cool very slowly until it achieves it's minimum lattice energy state, hence making it free of crystal defects. If the cooling schedule is allowed to be sufficiently slow, the final product is a solid with superior structural integrity. For a discrete optimization problem, SA provides a heuristic algorithm with a means to exploit the connection between thermo dynamic behaviour and the search for a global minima.

Book chapters have been devoted to SA, examples include Glover and Kochenberger [28] and Burke and Graham [11]. Web tutorials focusing on simulated annealing include
http://www.autonlab.org/tutorials/hillclimb.html and
http://www.rosswalker.co.uk/tutorials/amber_workshop/Tutorial_seven/section5.htm.

Simulated Annealing Concepts

Simulated annealing is a local search concept that avoids the iterative improvement approach, which gets easily trapped in a local optima, by accepting solutions that lead to a deterioration of the objective function. It begins with a single starting solution and continues exploring the neighbourhood for a solution with a lower cost. SA incorporates a statistical component in that moves to worse solutions are accepted with a specified probability that decreases over the course of the algorithm.

The annealing process has two phases: the liquid phase and the solid phase. In the liquid phase all particles arrange themselves randomly while in the ground state of the solid the particles are arranged in a highly structured lattice. The ground state is obtained only if the maximum state of the solid is sufficiently high and the cooling is performed sufficiently slowly. Otherwise, the solid will be frozen into a meta-stable state rather than into the true ground state. The energy of the current state is E_i and the energy of the next state is E_j. The state j is accepted as the current state with a probability given by $e^{((E_j-E_i)/(k_bT))}$ where T denotes the temperature of the cooling bath, the energy difference $E_j - E_i \leqslant 0$ and k_b is a physical constant called the Bolzmann constant.

The process of controlling this probability is referred to as the **cooling schedule**. This cooling schedule specifies the initial temperature, and the rate at which temperature decreases. After each stage the temperature is multiplied by a constant factor α $(0 < \alpha < 1)$. Therefore the temperature at the state E_j is given by $T_j = \alpha T_i$.

The termination of the SA heuristic is the same as the termination of the TS above (Section 2.5.2).

The concepts of the annealing process are brought together in the following simple step by step SA procedure laid out below.

```
Randomly generate an initial solution
Determine a suitable starting temperature and cooling factor
**Repeat**
    Randomly select a neighbourhood solution of the current solution
    Subtract the function value of the neighbourhood solution from the current solution
    If the value is positive, then replace the current solution with the neighbourhood
    solution
```

```
        Else, calculate the acceptance probability and draw a random probability
            If the random probability is greater than the acceptance probability, then re-
            place the current solution with the neighbourhood solution
            Else, keep the current solution
        Using a specific rule, cool the temperature
    **Until** the termination criteria are met.
```

2.5.4 Heuristic Algorithms for Cardinality Constrained Portfolio Optimisation

Heuristic methods are attractive because, while being a robust method for large size practical portfolio problems, they are independent of the objective function and offer solutions in a reasonable time. In this Section, a review of some previous portfolio management papers using heuristics for the cardinality constrained portfolio optimisation problem is given.

Crama and Schyns [17] present a simulated annealing approach. As well as a cardinality constraint they include constraints on turnover and trading related to the presence of an existing portfolio. Constraint violations are dealt with using a penalty function related to the magnitude of the violation raised to a power. Computational results are given for one test problem involving 151 assets.

Derigs and Nickel [19] present a simulated annealing based heuristic algorithm. In their approach stock returns and covariances are derived from a linear multi-factor model, where the factors are based on macro-economic variables. They present a case study based around an investment trust tracking the German DAX30 index. Their investment universe, some 202 assets, was taken from the DAX30 and STOXX200. Limited computational results are presented. More details of their work can be found in Derigs and Nickel [20].

Maringer and Kellerer [47] present an approach based on combining simulated annealing with evolutionary ideas. They maintain a population of portfolios that are improved in a simulated annealing fashion. As is normal in evolutionary approaches, poor portfolios in the population are replaced by better portfolios. Computational results are presented for two test problems involving 30 and 96 assets.

Ehrgott *et al.* [22] present an approach using multi-criteria decision making. In their problem they have a number of additional portfolio objectives (for example relating to dividends paid and Standard and Poors rating) and these are combined via weighted utility functions. They apply four different heuristic algorithm solution techniques (local search, simulated annealing, tabu search, genetic algorithm) to four test problems, involving up to 1416 assets.

Moral-Escudero *et al.* [51] present a genetic algorithm for the problem that uses two different crossover operators (random respectful recombination and random assorting recombination). Computational results are presented that make use of the test problems provided by Chang *et al.* [13].

Streichert and Tanaka-Yamawaki [68] combine a multi-objective evolutionary algorithm with QP local search. In their algorithm a variety of portfolios, each containing K assets, are generated. The proportion invested in each of the K assets is decided by solving a QP. Computational results are given for two of the five test problems used in Chang *et al.* [13] involving up to 85 assets.

Fernandez and Gomez [24] apply a Hopfield neural network to the problem. They also implement (albeit with minor changes) the three heuristics given in Chang *et al.* [13]. Computational results are presented that make use of the test problems provided by Chang *et al.* [13] which indicate that no one heuristic outperforms the others.

Branke *et al.* [10] use a multi–objective evolutionary algorithm in conjunction with the critical line algorithm of Markowitz [48]. They include a constraint (involving additional zero-one variables) based on German investment law. Computational results are given for three of the five test problems from Chang *et al.* [13], as well as for one further problem involving 500 assets.

Fieldsend *et al.* [25] use multi-objective evolutionary heuristics to determine the cardinality constrained frontiers. They provide empirical results on emerging markets and the S&P 100, for up to $K = 10$ assets. They compare their results to the UEF.

Cura [18] presented an approach based on particle swarm optimisation. In their heuristic each particle represents a portfolio. If a portfolio does not contain the appropriate number of assets then assets are added or deleted from the portfolio. Computational results are presented that make use of the test problems provided by Chang *et al*. [13]. They also report results for the same test problems using a genetic algorithm, tabu search and simulated annealing which indicate that no one heuristic outperforms the others.

Pai and Michel [54] apply a clustering approach to choosing the assets to include in the portfolio, thereby eliminating the cardinality constraint. They use an evolutionary strategy approach to decide the proportion to be invested in each of the assets. Computational results are presented for data drawn from the Bombay and Tokyo stock markets involving up to 225 assets.

Soleimani *et al*. [65] present a genetic algorithm for the problem. Their model includes constraints on the proportion invested in sectors (sets of assets). They present computational results for a number of test problem involving up to 2000 assets.

Anagnostopoulos and Mamanis [2] adopt a tri-objective view of the problem and apply three multiobjective evolutionary optimisation algorithms, specifically the Non-dominated Sorting Genetic Algorithm II (NSGA-II), the Strength Pareto Evolutionary Algorithm 2 (SPEA2) and the Pareto Envelope-based Selection Algorithm (PESA). Computational results are presented for two randomly generated problems involving 200 and 300 assets.

2.6 Portfolio Theory with Transaction Cost

Within the existing literature, the mathematical model for the transaction cost problem usually involves trying to move from a current portfolio to a new portfolio (or creating a new portfolio from cash) while the transaction costs are subtracted from expected returns. Many of the models call for the introduction of at least one additional binary variable as well as bounds on the portfolio. As a result of all these components for the transaction cost model the complexity of the problem increases. In this Section we consider papers found in the literature on transaction cost; the first four are those that cite Chang *et al*. [13] and the remaining seven are from the wider literature.

Angelelli *et al.* [3] consider solving a mixed integer linear programming model, involving transaction cost, heuristics and a cardinality constraint, with MAD model and the CVaR model. Historical data from Milan, Paris and Frankfort Stock Exchange is used with $N = 200, 300, 400$, and 600 assets and $K = 10$ or 20. with CPLEX (version 9) being used as the solver.

Chen and Cai [14] consider a generalization of the Markowitz MV model to include transaction cost and the buy-in threshold constraint. The transaction cost are assumed to be a known quantity at the beginning of the period, paid at the end of the period and a V-shaped function of difference between an existing and new portfolio. They compute the efficient frontier using the particle swarm optimization (PSO) heuristic. Eight different securities were used from the securities market in China.

Baule [9] considers the transaction cost model where short–selling is included. The model assumes transaction cost to be a non-convex function and it considers having a risk-free asset for which transaction cost is not incurred. They present results for a universe of 50 assets showing that for smaller investment volumes (up to 2000 Euros) the optimal number of assets could be just one asset while for larger investment volumes (above 20,000 Euro), the transaction costs become the major part of the total costs.

Chen *et al.* [15] present a possibilistic mean and variance portfolio selection model that includes changeable transaction costs which are assumed as a non-convex-non-concave function. PSO heuristic is proposed for this problem. Numerical results for five assets are given.

Adcock and Meade [1] considered incorporating variable transaction costs (via a weighting factor) into a mixed quadratic objective portfolio optimisation formulation. Computational results were presented for a tracking portfolio of 200 assets.

Yoshimoto [73] assumes transaction cost to be a V-shaped function of difference between an existing and new portfolio. He constructs a non–linear programming model for the portfolio problem with maximization of a quadratic utility function. He analyzes the effect of the transaction cost on the derived portfolio problem showing that inefficient portfolios are created by ignoring the transaction cost. The paper uses two securities from stock markets in Japan, the UK, the US, Germany, Canada, and France.

Perold [55] and Mulvey [53] estimate a transaction cost function using a piece-wise linear convex function. Konno and Wijayanayake [35] employ the MAD model for a branch and bound algorithm with concave transaction cost and minimal transaction unit constraints. Computational results for 200 assets chosen from the NIKKEI 225 Index are presented.

Li *et al.* [42] assume transaction cost to be a V-shaped function of difference between the existing and the new portfolio. They use a QP model to study the optimal portfolio selection problem with transaction cost and provide numerical results with three risky assets. This work is then extended by Xia *et al.* [71] to include a risk–free asset allowing short selling. Computational results are presented using 40 assets.

Lynch and Balduzzi [44] use dynamic programming to examine the decisions of the constant relative risk averse investors in the presence of proportional and fixed transaction cost. They consider two assets (the risky asset and the risk free asset) to show how portfolio choice and rebalancing behaviour are affected by return predictability.

Xue [72] presents a modified version of the Markowitz model to include concave transaction cost. The resulting model is a difference of two convex functions. To solve the model a branch and bound algorithm is designed. A series of numerical experiments for up to nine assets taken from the Shan Xi province in China is presented.

2.7 Summary

This Chapter gave a review of previous research in portfolio theory with particular focus on the cardinality constraint, heuristic algorithms and transaction cost. We began by presenting the MV model of Markowitz and discussing the several assumptions of the behavior of an investor. The efficient frontier for the MV model was considered and an alternative model used to obtain the efficient frontier was also given.

Next we presented alternatives to the Markowitz model divided into two categories: developments to the MV model and departures from the MV model. Within the first category (advances to the MV model), the single–index model of Sharpe [61], multi–factor model of Rosenberg [59], the CAPM model (independently developed) by Sharpe [62], Lintner [43] and Mossin [52] and the APT model of Ross [60] were all discussed. Then, in the second category (departures from the MV model), the portfolio return and portfolio risk of the MAD model by Konno and Yamazaki [36], the WGP model by Tamiz *et al.* [69] and the Minimax model by Young [74] were examined.

Then in Section 2.4, the discrete extensions to the MV model of buy-in threshold and cardinality constraint and the frontier produced by these extensions were presented. In the Section 2.5, a definition for heuristics was given and we presented the three heuristic techniques of genetic algorithm, tabu search and simulated annealing. In the final Section, transaction cost, a description of the mathematical model was given and we gave a review of the work found in the literature.

We noted that the focal point of most of the literature on the cardinality constraint and heuristic algorithms was from the work of Chang *et al.* [13] (with 94 citations in the Web of Science). As mentioned earlier the majority of the work used only one heuristic and considered only the data set used in Chang *et al.* [13].

As a result of our literature review above, we observed that for the transaction cost model, there was not a single mathematical perspective for the problem, because within the literature most authors adopted their own model. Computational results were often not detailed while computational times were missing and most authors used very few assets.

Chapter 3

Transaction Cost: Optimal Solutions

3.1 Introduction

Each time an investor buys or sells an asset an expense is incurred. In the classical work of Markowitz, the expenses associated with trading equities, were excluded from his QP model (minimise equation (2.1) subject to equations (2.2)-(2.4)). But today, the importance of integrating the transaction cost in a new portfolio and also in revising an existing portfolio are well acknowledged. Transaction cost should be desirably low, thus a portfolio manager must carefully consider trading and it's resulting cost. Transaction cost can be classified into two types: fixed and variable costs.

Fixed costs are paid on all transactions irrespective of the volume of the transaction. They consist of brokerage commissions and transfer fees.

Variable costs are dependent on the transaction volume. These costs are proportional to the volume traded when buying or selling an asset. They comprise of execution costs and opportunity costs. Execution costs can be further divided into market impact (the movement in the price of an asset that is the result of a trade plus the market-maker's spread: also called price impact) and market timing costs (the movement in the price of an asset at the time of a transaction that can be attributed to other market participants; it can be positive or negative). Opportunity costs are classified as the shortfall of an investment strategy resulting from a failure to execute all desired trades at the desired time.

In this Chapter, we expand on Markowitz's work to include transaction cost (both fixed and variable costs) and as well as an extension for cardinality constraints on the number of assets in the portfolio, bought, sold and traded.

The Chapter is organized as follows. In Section 3.2 we state the formulation of the problem which extends Markowitz's MV model to incorporate fixed and variable transaction cost. In Section 3.3 we present the cardinality extension to the transaction cost model. Computational considerations are next in Section 3.4. This is followed by Section 3.6 where we consider transaction cost frontiers. In Section 3.7 we show non-cardinality constrained transaction cost frontiers and in Section 3.8, we display cardinality constrained transaction cost frontiers. The Chapter is concluded with a summary in Section 3.9.

3.2 Formulation

In this Section, we present the mathematical notation (Section 3.2.1), problem constraints (Section 3.2.2), the objective function (Section 3.2.3) needed to solve the transaction cost optimization problem and then end the Section with a few remarks on our complete transaction cost formulation (Section 3.2.4).

3.2.1 Notation

In this Section, some of the notation is common with that adopted in Chapter 2, but for ease of reading we present it again below.

Let:

N	be the total number of assets available to be invested in,

μ_i	be the expected return of asset i ($i = 1, \ldots, N$),
σ_{ij}	be the covariance between the return of asset i and asset j ($i = 1, \ldots, N; j = 1, \ldots, N$),
R	be the desired level of expected return,
P_i	be the current price (value) of one unit (share) of asset i ($i = 1, \ldots, N$),
X_i	be current portfolio holding of asset i ($i = 1, \ldots, N$) in terms of number of units,
V^{new}	be new cash that can be invested in the current portfolio, (if $V^{new} > 0$ then we have new cash to be invested in the current portfolio, if $V^{new} < 0$ then we have cash to be taken out of the current portfolio),
f_i^b	be the fixed transaction cost ($\geqslant 0$) paid if we carry out any buying of asset i ($i = 1, \ldots, N$),
f_i^s	be the fixed transaction cost ($\geqslant 0$) paid if we carry out any selling of asset i ($i = 1, \ldots, N$),
c_i^s	be the variable transaction cost ($\geqslant 0$) for each unit of asset i ($i = 1, \ldots, N$) that is sold,
c_i^b	be the variable transaction cost ($\geqslant 0$) for each unit of asset i ($i = 1, \ldots, N$) that is bought,

When the price at which we can buy (sell) an asset i ($i = 1, \ldots, N$) differs from its current market price P_i, we have a bid–offer spread (the difference between the price available for an immediate bid (sale) and an immediate offer (purchase); also known as the bid–ask spread). This situation is an example of a market timing cost; consequently it is captured in the variable transaction costs, c_i^s and c_i^b. For example, suppose the current market price of an asset i ($i = 1, \ldots, N$) is 100, i.e. $P_i = 100$, the price at which we can sell asset i is 98 and the price at which we can buy asset i is 103 (we assume there are no other trading costs for simplicity). Therefore, $c_i^s = 100 - 98 = 2$ and $c_i^b = 103 - 100 = 3$ captures this bid–offer spread since each unit sold costs us 2 (we have turned an asset valued at 100 into 98 in cash), each unit bought costs us 3 (we pay 103 for an asset that is then valued at 100).

D	be the transaction cost limit, therefore the total transaction cost must be $\leqslant D$,
l_i	($\geqslant 0$) be the minimum proportion of the total investment held in asset i ($i = 1, \ldots, N$), if any investment is made in asset i,
u_i	($\geqslant 0$) be the maximum proportion of the total investment held in asset i ($i = 1, \ldots, N$), if any investment is made in asset i (so $0 \leqslant l_i \leqslant u_i \leqslant 1$),
L_i^b	be the minimum number of units of asset i ($i = 1, \ldots, N$) we must buy if we carry out any buying of asset i,
L_i^s	be the minimum number of units of asset i ($i = 1, \ldots, N$) we must sell if we carry out any selling of asset i,
U_i^b	be the maximum number of units of asset i ($i = 1, \ldots, N$) we can buy if we carry out any buying of asset i (so $U_i^b \geqslant L_i^b$), and
U_i^s	be the maximum number of units of asset i ($i = 1, \ldots, N$) we can sell if we carry out any selling of asset i (so $U_i^s \geqslant L_i^s$).

The decision variables are:

x_i	the number of units of asset i ($i = 1, \ldots, N$) in the new portfolio,
G_i	($\geqslant 0$) the total transaction cost (fixed and variable) incurred in trading (buying or selling) an asset i ($i = 1, \ldots, N$),
y_i^s	the number of units of asset i ($i = 1, \ldots, N$) sold,
y_i^b	the number of units of asset i ($i = 1, \ldots, N$) bought,
α_i^s	=1 if we sell any of asset i ($i = 1, \ldots, N$), =0 otherwise, and

α_i^b =1 if we buy any of asset i ($i = 1, \ldots, N$),

=0 otherwise.

Although the variables x_i, y_i^s and y_i^b should strictly be integer, without significant loss of generality since the sums invested are likely to be large, we regard them as continuous variables.

3.2.2 Constraints

In this Section, the constraints of the transaction cost portfolio optimization model are given below. As this model has not been presented in the literature before we give a detailed explanation of each constraint.

The return equation is

$$\frac{\sum_{i=1}^{N} \mu_i P_i x_i}{\sum_{k=1}^{N} P_k x_k} = R \tag{3.1}$$

where $P_i x_i$ is the invested amount in asset i ($i = 1, \ldots, N$) for an expected return of μ_i. The numerator is the total interest income, while the denominator represents the total investment made. Equation (3.1) relates to the expected return from the chosen portfolio and the assumption is that this expected return will continue over time.

The appropriate limits to the number of units bought or sold are given by the following equations:

$$L_i^s \alpha_i^s \leqslant y_i^s \leqslant U_i^s \alpha_i^s, \quad i = 1, \ldots, N, \tag{3.2}$$

$$L_i^b \alpha_i^b \leqslant y_i^b \leqslant U_i^b \alpha_i^b, \quad i = 1, \ldots, N, \tag{3.3}$$

where $\alpha_i^s = 0$ means that none of asset i is sold while $\alpha_i^s = 1$ requires that the number of units sold would be between the upper and lower bounds of selling, i.e. $L_i^s \leqslant y_i^s \leqslant U_i^s$. Similarly, $\alpha_i^b = 0$ means that none of asset i is bought while $\alpha_i^b = 1$ requires that the number of units bought would be between the upper and lower bounds of buying, i.e. $L_i^b \leqslant y_i^b \leqslant U_i^b$.

The equation

$$\alpha_i^s + \alpha_i^b \leqslant 1, \quad i = 1, \ldots, N, \tag{3.4}$$

makes certain that the selling and buying of an asset i ($i = 1, \ldots, N$) cannot simultaneously happen. This constraint does allow us not to trade the asset i when the binary decision variable for buying and selling of i are both zero (so $\alpha_i^s = \alpha_i^b = 0$).

The balance constraint on the number of shares in the portfolio is

$$x_i = X_i + y_i^b - y_i^s, \quad i = 1, \ldots, N. \tag{3.5}$$

Equation (3.5) states that the number of units in the new portfolio is equal to the number of units in the old portfolio plus any units bought minus any units sold of asset i ($i = 1, \ldots, N$).

A budget is arguably the most important tool for a fund manager. It puts in perspective an individual's ideas about risk and the resources available providing a game plan for operating a portfolio. The constraints relating to the budget limit the degree of total market exposure assumed by an investor, by requiring that the total value of the portfolio to be less than or equal to the available wealth. Properly used, a budget can help fund managers to meet their goals, create more profitable portfolios giving them the edge through tough financial times. Equations (3.6)-(3.8) all relate to the portfolio budget.

$$G_i = c_i^s y_i^s + c_i^b y_i^b + f_i^s \alpha_i^s + f_i^b \alpha_i^b, \quad i = 1, \ldots, N, \tag{3.6}$$

says that the total transaction cost for asset i equals the variable transaction cost plus the fixed transaction cost incurred in buying and selling an asset i ($i = 1, \ldots, N$).

$$\sum_{i=1}^{N} G_i \leqslant D \tag{3.7}$$

is the transaction cost limit constraint. It states that the sum of the transaction cost (fixed and variable) is less than or equal to the transaction cost limit.

Then,

$$\sum_{i=1}^{N} P_i x_i = \sum_{i=1}^{N} P_i X_i + V^{new} - \sum_{i=1}^{N} G_i \tag{3.8}$$

represents the monetary balance constraint that ensures that the monetary value of the new portfolio is equal to the monetary value of the current portfolio, plus any new cash, minus the total transaction cost for each asset i ($i = 1, \ldots, N$).

3.2.3 The Objective Function

The objective function is

$$\text{Minimise} \sum_{i=1}^{N} \sum_{j=1}^{N} \sigma_{ij} w_i w_j. \tag{3.9}$$

This risk objective seems to be the same as for the Markowitz model (equation (2.1)) where the summation involves terms written as the covariance multiplied by the proportion invested in each asset i ($i = 1, \ldots, N$). But, here the proportion invested in an asset i is equal to $P_i x_i / \sum_{k=1}^{N} P_k x_k$. Consequently, w_i is a nonlinear term since the value of the denominator is not constant due to the effect of transaction cost.

From the monetary balance constraint (equation (3.8)) we have that $\sum_{i=1}^{N} P_i x_i \approx \sum_{i=1}^{N} P_i X_i + V^{new}$ provided that the total transaction cost ($\sum_{i=1}^{N} G_i$) is relatively small compared to the sums invested. Hence, we set the "proportion" (w_i) invested in asset i to be defined as

$$w_i = \frac{P_i x_i}{\sum_{k=1}^{N} P_k X_k + V^{new}}, \quad i = 1, \ldots, N, \tag{3.10}$$

where the denominator is now a known constant. This seems a reasonable assumption to make in order to achieve a quadratic objective.

Although we have referred to "proportion" above, unlike the Markowitz budget constraint

($\sum_{i=1}^{N} w_i = 1$, equation (2.1)), in general in our transaction cost model the w_i do not sum to one, i.e. $\sum_{i=1}^{N} w_i \neq 1$. In fact we have that w_i actually underestimates the true proportion invested in asset i in the new portfolio.

From equation (3.10), we have $w_i = \frac{P_i x_i}{\sum_{k=1}^{N} P_k X_k + V^{new}}$ while, the monetary balance constraint, equation (3.8) states $\sum_{i=1}^{N} P_i x_i = \sum_{i=1}^{N} P_i X_i + V^{new} - \sum_{i=1}^{N} G_i$. Therefore,

$$\begin{aligned}
\sum_{i=1}^{N} w_i &= \frac{\sum_{i=1}^{N} P_i X_i + V^{new} - \sum_{i=1}^{N} G_i}{\sum_{k=1}^{N} P_k X_k + V^{new}} \\
&= 1 - \frac{\sum_{i=1}^{N} G_i}{\sum_{k=1}^{N} P_k X_k + V^{new}} \\
&\sum_{i=1}^{N} G_i \geqslant 0 \text{ and } \sum_{k=1}^{N} P_k X_k + V^{new} \geqslant 0 \\
&< 1
\end{aligned}$$

Therefore in general, we have $\sum_{i=1}^{N} w_i < 1$.

Although we have approximated the proportion invested in an asset for the purposes of achieving a quadratic objective, if we have l_i and u_i as the proportion limits on asset i in the new portfolio then these can be represented exactly using:

$$l_i \leqslant \frac{P_i x_i}{\sum_{k=1}^{N} P_k x_k} \leqslant u_i, \quad i = 1, \ldots, N. \tag{3.11}$$

In practice l_i and u_i are just guides as to the value the investor wishes to see. As asset values (prices) in the portfolio fluctuate, fluctuations occur in the value of the portfolio and thus, the actual proportions devoted to each asset also fluctuates.

3.2.4 The Complete Transaction Cost Formulation

Our complete formulation of the transaction cost optimization problem is

$$\text{Minimise } \sum_{i=1}^{N}\sum_{j=1}^{N}\sigma_{ij}w_iw_j \tag{3.12}$$

subject to

$$\frac{\sum_{i=1}^{N}\mu_iP_ix_i}{\sum_{k=1}^{N}P_kx_k} = R, \tag{3.13}$$

$$L_i^s\alpha_i^s \leqslant y_i^s \leqslant U_i^s\alpha_i^s, \qquad i = 1,\dots,N, \tag{3.14}$$

$$L_i^b\alpha_i^b \leqslant y_i^b \leqslant U_i^b\alpha_i^b, \qquad i = 1,\dots,N, \tag{3.15}$$

$$\alpha_i^s + \alpha_i^b \leqslant 1, \qquad i = 1,\dots,N, \tag{3.16}$$

$$x_i = X_i + y_i^b - y_i^s, \qquad i = 1,\dots,N, \tag{3.17}$$

$$G_i = c_i^sy_i^s + c_i^by_i^b + f_i^s\alpha_i^s + f_i^b\alpha_i^b, \qquad i = 1,\dots,N, \tag{3.18}$$

$$\sum_{i=1}^{N}G_i \leqslant D, \tag{3.19}$$

$$\sum_{i=1}^{N}P_ix_i = \sum_{i=1}^{N}P_iX_i + V^{new} - \sum_{i=1}^{N}G_i, \tag{3.20}$$

$$w_i = \frac{P_ix_i}{\sum_{k=1}^{N}P_kX_k + V^{new}}, \qquad i = 1,\dots,N, \tag{3.21}$$

$$l_i \leqslant \frac{P_ix_i}{\sum_{k=1}^{N}P_kx_k} \leqslant u_i, \qquad i = 1,\dots,N, \tag{3.22}$$

$$w_i, x_i, y_i^s, y_i^b, G_i \geq 0, \qquad i = 1,\dots,N, \tag{3.23}$$

$$\alpha_i^s, \alpha_i^b \in [0,1], \qquad i = 1,\dots,N. \tag{3.24}$$

There are a number of remarks we can make with respect to this formulation:

1. One equation (equation (3.13)) in our formulation is nonlinear, but can be easily linearised by multiplying both sides of the equation by the denominator.
2. Our formulation, a mixed integer program, is useful for all assets that have return values over time and can be handled by standard solvers such as CPLEX.

3. Our formulation aids the investment policy plan of portfolio managers. The formulation is a strategic asset allocation model in that it is useful in determining the long–term policy asset weights in a portfolio. It is not a tactical asset allocation model because it does not adjust for up to the minute changes in the relative values of the various asset classes.
4. Any computational solution times within reason does not pose any problem because they can be justified by the following:
   (a) given the operational allocation nature of our problem the investor decides which asset mix is appropriate for their needs during the planning phase.
   (b) given the long term nature of the investment any decision as to whether (or not) to change to a new portfolio are made relatively infrequently.
   (c) it is conceivable that fund managers could devote many computers to compute a new portfolio.

3.3 Cardinality Extension

Restrictions relating to the number of assets in the portfolio can also be incorporated in our transaction cost optimisation model. Once again, some of the notation is the same as in earlier chapters but, for ease of reading we present it below.

Let:

K	be the desired number of distinct assets we wish to hold,
K^S	be the maximum number of assets we can sell,
K^B	be the maximum number of assets we can buy,
K^T	be the maximum number of assets we can trade (buy or sell),

and a decision variable,

δ_i $= 1$ if we hold asset i $(i = 1, \ldots, N)$ in the new portfolio,

$= 0$ otherwise.

Then the constraints to represent these cardinality restrictions are:

$$\sum_{i=1}^{N} \delta_i = K, \tag{3.25}$$

$$\sum_{i=1}^{N} \alpha_i^s \leqslant K^S, \tag{3.26}$$

$$\sum_{i=1}^{N} \alpha_i^b \leqslant K^B, \tag{3.27}$$

$$\sum_{i=1}^{N} (\alpha_i^s + \alpha_i^b) \leqslant K^T, \tag{3.28}$$

$$w_i \leqslant \delta_i \quad i = 1, \ldots, N. \tag{3.29}$$

Equation (3.25) (as was the case in equation (2.23)), restricts the number of assets to be held to the desired cardinality of the investor. Equation (3.26) restricts the decision variable relating to selling to be less than or equal to the maximum number specified by the investor. Similarly, equation (3.27) restricts the decision variable relating to buying to be less than or equal to the maximum number specified by the investor. Equation (3.28) ensures that the number of assets bought or sold is not greater than the trading limit. Constraint (3.29) ensures that only assets in the portfolio have a weight; if an asset i $(i = 1, \ldots, N)$ is not in the portfolio then $\delta_i = 0$ means $w_i = 0$ and given the equality equation for w_i (equation (3.10)) then this means that $x_i = 0$ also.

When revising an existing portfolio $(X_i,\ i = 1, \ldots, N)$ to create a new portfolio $(x_i,\ i =$

$1, \ldots, N$) we include the following constraints:

$$x_i \geqslant L_i^b \delta_i \text{ if } X_i = 0, \qquad i = 1, \ldots, N, \tag{3.30}$$

$$y_i^b \geqslant L_i^b \delta_i \text{ if } X_i = 0, \qquad i = 1, \ldots, N, \tag{3.31}$$

$$\alpha_i^b \geqslant \delta_i \text{ if } X_i = 0, \qquad i = 1, \ldots, N, \tag{3.32}$$

$$y_i^s \geqslant X_i(1 - \delta_i) \text{ if } X_i > 0, \qquad i = 1, \ldots, N, \tag{3.33}$$

$$\alpha_i^s \geqslant 1 - \delta_i \text{ if } X_i > 0, \qquad i = 1, \ldots, N. \tag{3.34}$$

Equations (3.30)-(3.32) states that if we currently hold none of asset i (so $X_i = 0$) and we have it in our chosen portfolio (so $\delta_i = 1$), the new holding and the amount bought (y_i^b) must be at least the minimum amount we can buy (L_i^b) and the zero/one buy decision variable must also be one (so $\alpha_i^b = 1$). In a similar fashion, if we currently hold some of asset i (so $X_i > 0$), but we choose not to have it in our new portfolio (so $\delta_i = 0$), we must sell the current holding (y_i^s) and the decision variable for selling must be one (so $\alpha_i^s = 1$). This leads to the constraints in equations (3.33) and (3.34).

These constraints (equations (3.30) -(3.34)) would be implied in any optimal solution to the complete transaction cost cardinality formulation. However, adding these constraints potentially improves computational performance (via improvement of the continuous relaxation), which is why they are added here.

3.3.1 The Complete Transaction Cost Cardinality Formulation

Our complete formulation of the transaction cost optimization problem with cardinality restrictions is

$$\text{Minimise} \sum_{i=1}^{N} \sum_{j=1}^{N} \sigma_{ij} w_i w_j \tag{3.35}$$

subject to

$$\frac{\sum_{i=1}^{N} \mu_i P_i x_i}{\sum_{k=1}^{N} P_k x_k} = R, \tag{3.36}$$

$$L_i^s \alpha_i^s \leqslant y_i^s \leqslant U_i^s \alpha_i^s, \qquad i = 1, \ldots, N, \tag{3.37}$$

$$L_i^b \alpha_i^b \leqslant y_i^b \leqslant U_i^b \alpha_i^b, \qquad i = 1, \ldots, N, \tag{3.38}$$

$$\alpha_i^s + \alpha_i^b \leqslant 1, \qquad i = 1, \ldots, N, \tag{3.39}$$

$$x_i = X_i + y_i^b - y_i^s, \qquad i = 1, \ldots, N, \tag{3.40}$$

$$G_i = c_i^s y_i^s + c_i^b y_i^b + f_i^s \alpha_i^s + f_i^b \alpha_i^b, \qquad i = 1, \ldots, N, \tag{3.41}$$

$$\sum_{i=1}^{N} G_i \leqslant D, \tag{3.42}$$

$$\sum_{i=1}^{N} P_i x_i = \sum_{i=1}^{N} P_i X_i + V^{new} - \sum_{i=1}^{N} G_i, \tag{3.43}$$

$$w_i = \frac{P_i x_i}{\sum_{k=1}^{N} P_k X_k + V^{new}}, \qquad i = 1, \ldots, N, \tag{3.44}$$

$$l_i \leqslant \frac{P_i x_i}{\sum_{k=1}^{N} P_k x_k} \leqslant u_i, \qquad i = 1, \ldots, N, \tag{3.45}$$

$$\sum_{i=1}^{N} \delta_i = K, \tag{3.46}$$

$$\sum_{i=1}^{N} \alpha_i^s \leqslant K^S, \tag{3.47}$$

$$\sum_{i=1}^{N} \alpha_i^b \leqslant K^B, \tag{3.48}$$

$$\sum_{i=1}^{N} (\alpha_i^s + \alpha_i^b) \leqslant K^T, \tag{3.49}$$

$$w_i \leqslant \delta_i, \qquad i = 1, \ldots, N, \tag{3.50}$$

$$x_i \geqslant L_i^b \delta_i \text{ if } X_i = 0, \qquad i = 1, \ldots, N, \tag{3.51}$$

$$y_i^b \geqslant L_i^b \delta_i \text{ if } X_i = 0, \qquad i = 1, \ldots, N, \tag{3.52}$$

$$\alpha_i^b \geqslant \delta_i \text{ if } X_i = 0, \qquad i = 1, \ldots, N, \tag{3.53}$$

$$y_i^s \geqslant X_i(1 - \delta_i) \text{ if } X_i > 0, \qquad i = 1, \ldots, N, \tag{3.54}$$

$$\alpha_i^s \geqslant 1 - \delta_i \text{ if } X_i > 0, \qquad i = 1, \ldots, N, \tag{3.55}$$

$$w_i, x_i, y_i^s, y_i^b, G_i \geq 0, \qquad i = 1, \ldots, N, \tag{3.56}$$

$$\delta_i,\ \alpha_i^s,\ \alpha_i^b \in [0, 1], \qquad i = 1, \ldots, N. \tag{3.57}$$

There are a number of remarks we can make with respect to this formulation:

1. All points presented in Section 3.2.4 also, apply to this formulation. In particular we can use CPLEX to solve the problem.
2. The formulation presented above is sometimes referred to as a rebalancing or a revision problem because our aim is to move from the existing portfolio to a new one. Yet, our formulation includes the case where we wish to create a new portfolio from cash with $V^{new} > 0$ and $X_i = 0$ $i = 1, \ldots, N$.
3. Our formulation addressees the problem, "if we revise the present portfolio now, what are the optimal portfolios that can be found given the present market conditions?"In other words, through solving our formulation, investors and fund managers have information to answer the following questions:
   (a) "should we rebalance our current portfolio now or leave it unchanged?"
   (b) "which assets in particular should be purchased for the portfolio to improve it's performance?"

3.4 Computational Considerations

Although the transaction cost models (minimise equation 3.12 subject to equations (3.13)-(3.24) and minimise equation 3.35 subject to equations (3.36)-(3.57)) are valid there are a number of additional steps we have taken to tighten the continuous relaxation, hence potentially improving the computational performance. These are:

1. $$\text{If } X_i = 0 \text{ set } \alpha_i^s = y_i^s = 0 \quad i = 1, \ldots, N.$$

   This declaration statement says that if we do not have any of asset i in our portfolio, we set the binary variable for selling and the amount units to sell to zero.

2. $$\text{Set } U_i^s = \min[U_i^s, X_i] \quad i = 1, \ldots, N.$$

   In this statement we declare that the upper limit for selling is at most the current holding of asset i.

3. Consider selling asset i ($i = 1, \ldots, N$), assuming $X_i > 0$. If we sell y_i^s of asset i we incur the transaction cost $f_i^s + c_i^s y_i^s$ which cannot exceed the transaction cost limit D (i.e. we must have that $f_i^s + c_i^s y_i^s \leqslant D$). Therefore, we can update the upper limit U_i^s on the amount of asset i that can be sold using

   $$U_i^s = 0 \text{ if } f_i^s > D,$$

   $$U_i^s = \min[U_i^s, (D - f_i^s)/c_i^s] \text{ if } c_i^s > 0 \text{ and } f_i^s \leqslant D.$$

4. A similar argument (to 3.) can be applied to buying asset i ($i = 1, \ldots, N$). If we buy y_i^b of asset i we incur the transaction cost $f_i^b + c_i^b y_i^b$ and this must be less that the limit on transaction cost D (i.e. $f_i^b + c_i^b y_i^b \leqslant D$). We can update the upper limit U_i^b on the amount of asset i that can be bought using

   $$U_i^b = 0 \text{ if } f_i^b > D,$$

   $$U_i^b = \min[U_i^b, (D - f_i^b)/c_i^b] \text{ if } c_i^b > 0 \text{ and } f_i^b \leqslant D.$$

3.5 Data Sets

We tested the performance of our transaction cost problem using publicly available test problems relating to six major market indices, available from OR-Library (Beasley, [6]).

Five of our market indices were the Hang Seng (Hong Kong), DAX 100 (Germany), FTSE 100 (UK), S&P 100 (USA) and the Nikkei 225 (Japan), as taken from http://people.brunel.ac.uk/∼mastjjb/jeb/orlib/portinfo.html. The remaining market index was the S&P 500 (USA), as taken from http://people.brunel.ac.uk/∼mastjjb/jeb/orlib/indtrackinfo.html. The size of our six test problems ranged from $N = 31$ (Hang Seng) to $N = 457$ (S&P 500). We used $V^{new} = 0$, $l_i = 0$, $u_i = 1$ $(i = 1, ..., N)$ and $K = 10$. The values for X_i, P_i, L_i^s, L_i^b, U_i^s, U_i^b, f_i^s, f_i^b, c_i^s and c_i^b for all $i = 1, \ldots, N$ are found on the CD accompanying this thesis.

Our transaction cost models were implemented using AMPL and its associated script language. The solver we used was CPLEX (version 11.0). The system runs under Windows NT and in our computational work we used an Intel Core2 pc with a 2.40 GHz processor and 3.24 GB RAM.

3.6 Transaction Cost Frontiers

One aspect of transaction cost optimization that appears to have received no attention in the literature is the fact that in the presence of constraints of the type we have considered above the frontier produced is markedly different from the UEF. In this Section we answer several questions:

1. *Is the transaction cost efficient frontier discontinuous?*
2. *Are fixed transaction costs (in themselves) sufficient to make the transaction cost efficient frontier discontinuous even when there are no cardinality restrictions?*

Here we note, the Figures represented in this Section (and throughout this Chapter) usually contain the trading portfolio frontier (the set of points that correspond to the least risk portfolios at all feasible return levels) for the market index in dark blue, the original portfolio in red, the trading portfolio frontier for the original portfolio in green and the efficient frontier for the original portfolio in pink.

3.6.1 Comparing Transaction Cost to the Unconstrained Problem

Before we are able to answer any of the questions raised above, we make a comment on comparing the transaction cost problem to the unconstrained problem. For the unconstrained case risk and return are calculated using the standard Markowitz equations, where the proportions used add to one (i.e. $\sum_{i=1}^{N} w_i = 1$, equation (2.1)). In Section 3.2.3 we state that in general the proportions do not sum to one (i.e. $\sum_{i=1}^{N} w_i < 1$). As a result, in order to compare the new portfolio ($x_i,\ i = 1, \ldots, N$) that results from the transaction cost optimization problem (e.g. graphically) we need to calculate its risk and return in an analogous way.

Hence, for the purpose of comparing with the unconstrained case the risk is

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\sigma_{ij}\frac{P_iP_jx_ix_j}{(\sum_{k=1}^{N}P_kx_k)^2} \tag{3.58}$$

and the return is

$$\sum_{i=1}^{N}\mu_i\frac{P_ix_i}{\sum_{k=1}^{N}P_kx_k}. \tag{3.59}$$

Here $[P_ix_i/\sum_{k=1}^{N}P_kx_k]$ has been used to represent the proportion of the invested portfolio (which is not the same as the original money available as transaction cost has been incurred) invested in asset i ($i = 1, \ldots, N$).

3.6.2 Fixed and Variable Transaction Cost Efficient Frontier

To answer the question, *Is the transaction cost efficient frontier (TCEF) discontinuous?*, we consider creating the efficient frontier based on the transaction cost model: minimise equation (3.12) subject to equations (3.13)-(3.24). We do this using all $N = 31$ assets from the Hang Seng market index, $V^{new} = 0$, $l_i = 0$ ($i = 1, \ldots, N$), $u_i = 1$ ($i = 1, \ldots, N$), $D = 13210$ and in the Appendix (Table 7.1) we give the values for P_i, X_i, L_i^s, L_i^b, U_i^s, U_i^b, f_i^s, f_i^b, c_i^s and c_i^b for all $i = 1, \ldots, N$.

In Figure 3.1, we show the trading portfolio frontier (for the Hang Seng market index), TCEF (for the original portfolio) and the original portfolio. The TCEF is based upon 1000 equally spaced return levels from minimum average return to maximum average return. The Figure shows that the TCEF is discontinuous.

3.6.3 Four Asset Example

In order to answer our second question above: *Are fixed transaction costs (in themselves, i.e. a G_i with $f_i^s > 0$, $f_i^b > 0$ and $c_i^s = c_i^b = 0$) sufficient to make the transaction cost efficient frontier discontinuous even when there are no cardinality restrictions?*, we consider the small four asset example shown in Table 3.1, drawn from the FTSE data set (Chang *et al.* [13] also used this data set to illustrate the discontinuities in the cardinality constrained efficient frontier).

Asset	Return	Standard	Correlation Matrix			
	(weekly)	Deviation	1	2	3	4
1	0.004798	0.046351	1	0.118368	0.143822	0.252213
2	0.000659	0.030586		1	0.164589	0.099763
3	0.003174	0.030474			1	0.083122
4	0.001377	0.035770				1

Table 3.1: A Four Asset Example

Consider the transaction cost model: minimise equation (3.12) subject to equations (3.13)-(3.24), in which there are no variable costs (so $c_i^s = c_i^b = 0$ resulting in $G_i = f_i^s \alpha_i^s + f_i^b \alpha_i^b$) and we examine 1000 equally spaced return levels from R_{min} (minimum average return) to R_{max} (maximum average return). We let $X_i = 100$ ($i = 1, 2, 3, 4$), the limit on transaction cost to be two (i.e. $D = 2$), then for every $i = 1, 2, 3, 4$, $P_i = 10$, $L_i^s = L_i^b = 0$, $U_i^s = U_i^b = \infty$ and $f_i^s = f_i^b = 1$. We illustrate this for the following three cases:

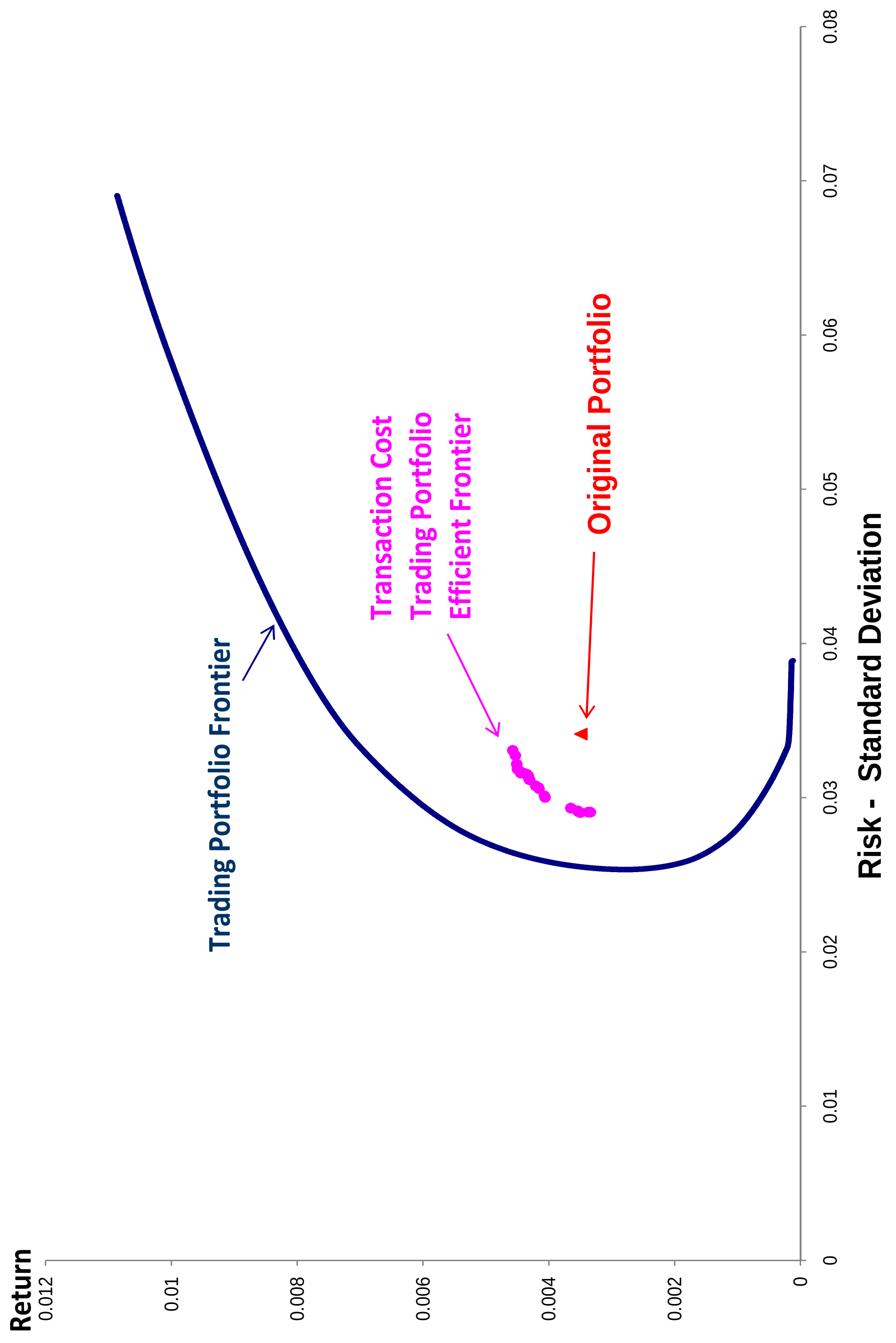

Figure 3.1: An example of the TCEF for the Hang Seng Market Index

Case 1: Zero Cash Investment

In this scenario, there are no exogenous cash injections, (i.e. $V^{new} = 0$). With no new money available to invest in the portfolio the monetary balance constraint, equation (3.8), becomes $\sum_{i=1}^{N} P_i x_i = \sum_{i=1}^{N} P_i X_i - \sum_{i=1}^{N} G_i$. This would then imply that the monetary value of our new portfolio is decreased by the transaction cost payment. In other words, equation (3.8) assumes that when cash is not available for a new portfolio (x_i, $i = 1, \ldots, N$) the cost of transacting is taken from our existing portfolio (X_i, $i = 1, \ldots, N$).

In Figure 3.2, we graphically present the trading portfolio frontier and in Figure 3.3 the efficient frontier produced from our original portfolio as a result of no cash inflows or outflows. The Figures show that fixed costs in themselves are sufficient to produce discontinuous trading portfolio and efficient frontiers even when there are no explicit cardinality restrictions for the case $V^{new} = 0$.

Case 2: Positive Cash Investment

A positive cash investment means there is new money to invest in the portfolio, (i.e. $V^{new} > 0$). Figure 3.4 shows the trading portfolio frontier while Figure 3.5 depicts the efficient frontier produced from our original portfolio when $V^{new} = 1.1 \sum_{i=1}^{N} P_i X_i$ (i.e. the original portfolio's value is increased by 10%). With cash available to invest, the current portfolio's (X_i, $i = 1, \ldots, N$) outlook is more favourable. For instance, when money added to the portfolio is greater than or equal to the transaction cost, (i.e. $V^{new} \geqslant \sum_{i=1}^{N} G_i$), the current portfolio value can be maintained or increased in the new portfolio (x_i, $i = 1, \ldots, N$). In essence, the case $V^{new} > 0$ can offset transaction cost.

In Figure 3.4 and Figure 3.5, we can see the trading portfolio and efficient frontiers produced from our original portfolio as a result of a positive cash investment are discontinuous. Therefore fixed costs in themselves are sufficient to produce discontinuous frontiers even when there are no cardinality restrictions for the case $V^{new} > 0$.

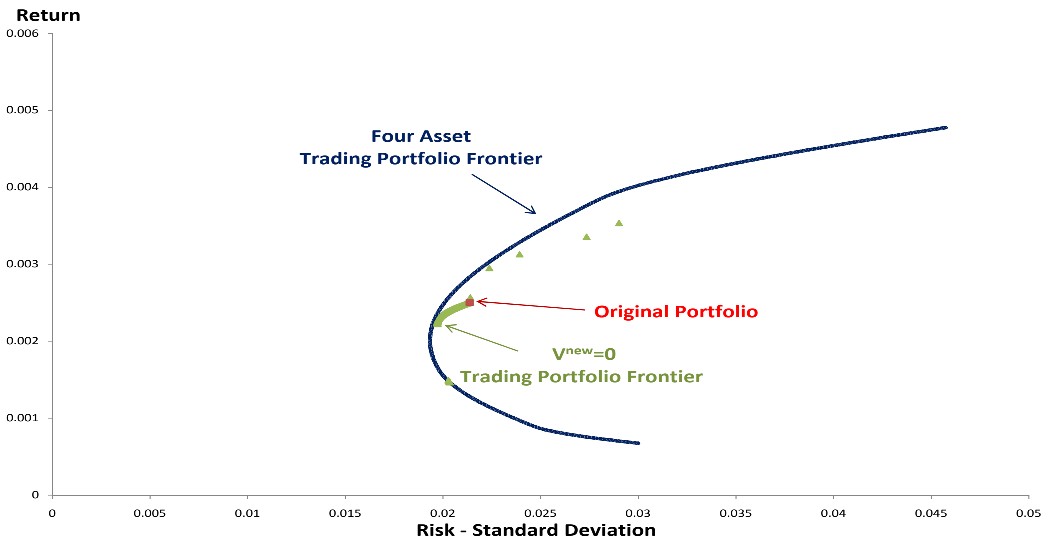

Figure 3.2: A Four Asset Example Trading Portfolio Frontier for $V^{new} = 0$

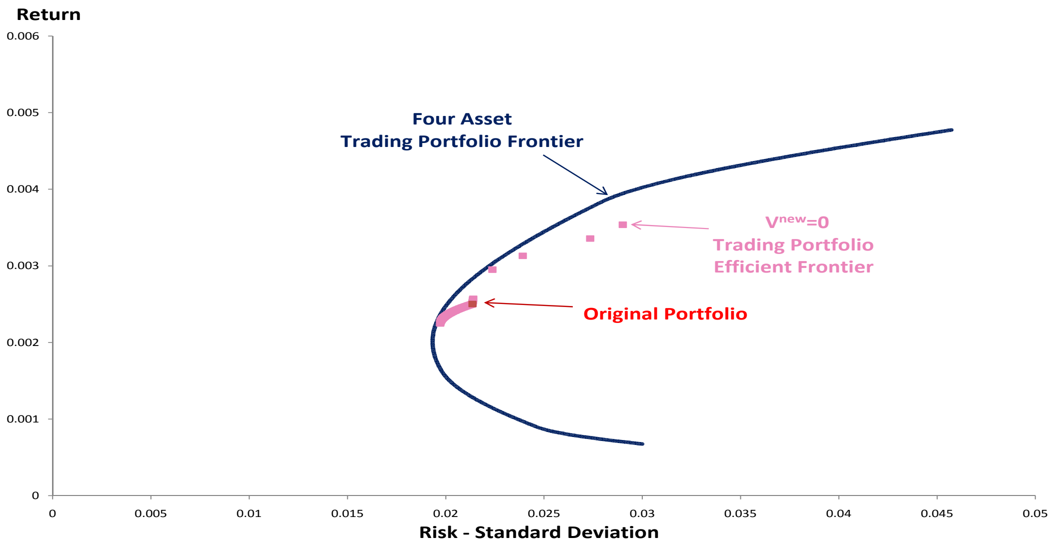

Figure 3.3: A Four Asset Example Trading Portfolio Efficient Frontier for $V^{new} = 0$

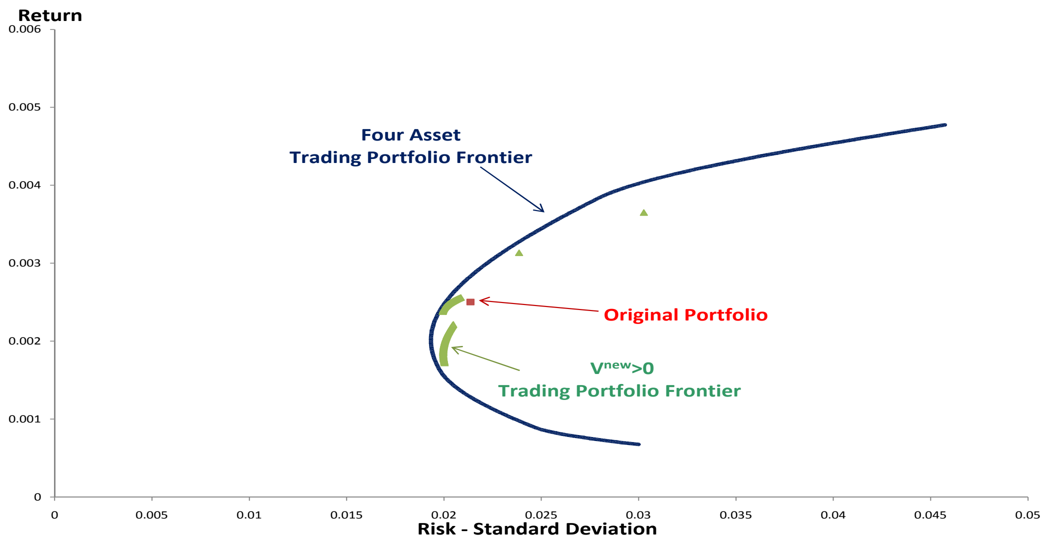

Figure 3.4: A Four Asset Example Trading Portfolio Frontier for $V^{new} > 0$

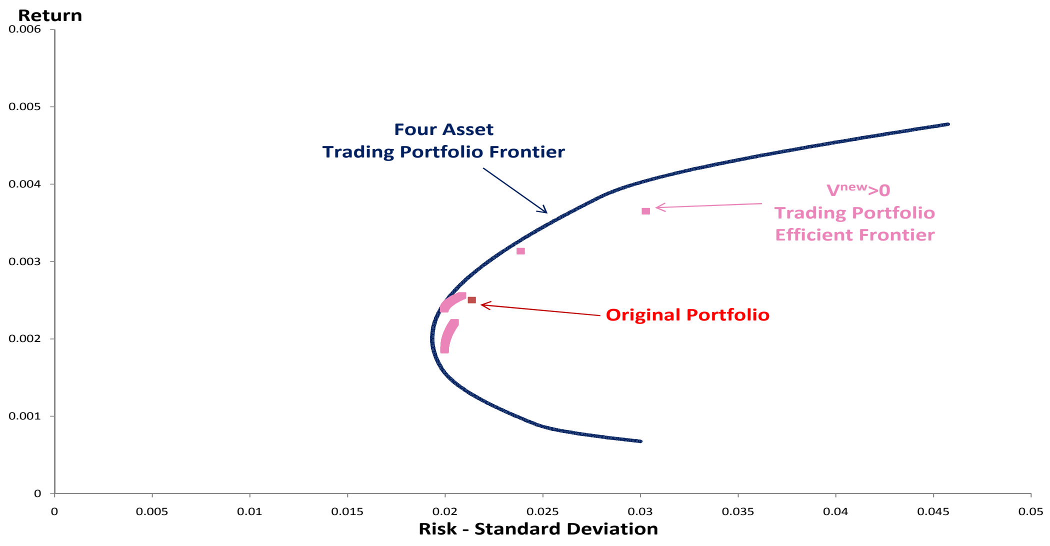

Figure 3.5: A Four Asset Example Trading Portfolio Efficient Frontier for $V^{new} > 0$

Case 3: Negative Cash Investment

No investor wants to be in a position where their current portfolio is losing value. In recent times the world markets have been volatile. For instance, after starting the year 2008 with 6,500 points, the FTSE 100 index of top UK companies had fallen by 3,000 points by the end of October. This situation causes money to be withdrawn from the portfolio, (i.e. $V^{new} < 0$, where 0 represents the starting position of the portfolio). With the portfolio value suffering a reduction and the loss of monetary value from transacting, the investor wishes to make certain that any transaction would result in a more efficient portfolio.

Figure 3.6 and Figure 3.7 depict the trading portfolio and efficient frontiers (respectively) produced from our original portfolio when $V^{new} = 0.9\sum_{i=1}^{N} P_i X_i$. The original portfolio experiences a 10% drop in value and rebalancing takes place. The Figures show that fixed costs in themselves are sufficient to produce discontinuous trading portfolio and efficient frontiers even when there are no cardinality restrictions for the case $V^{new} < 0$.

Implications of The Three Cash Investment Scenarios

1. In each case (i.e. $V^{new} < 0$, $V^{new} > 0$ and $V^{new} = 0$) the trading portfolio frontier produced is a discontinuous curve.
2. All three cases of V^{new} (i.e. $V^{new} < 0$, $V^{new} > 0$ and $V^{new} = 0$) produce a fixed transaction cost efficient frontier (FTCEF). As is the case with the trading portfolio frontier, the FTCEF is also discontinuous. Hence, when faced with only fixed transaction cost, there will be returns which no rational investor could consider.
3. Figure 3.8 graphically depicts all three cash investment strategies i.e. $V^{new} < 0$ (purple), $V^{new} > 0$ (green) and $V^{new} = 0$ (blue) trading portfolio frontiers. It demonstrates that there are return levels which are most beneficial to the investor because of different V^{new} representations. For instance
   (a) A portfolio at a return level A in Figure 3.8 would be better off if money is taken out of the portfolio (so $V^{new} < 0$).
   (b) A portfolio at a return level B in Figure 3.8 would be better off if money is placed into the portfolio (so $V^{new} > 0$).

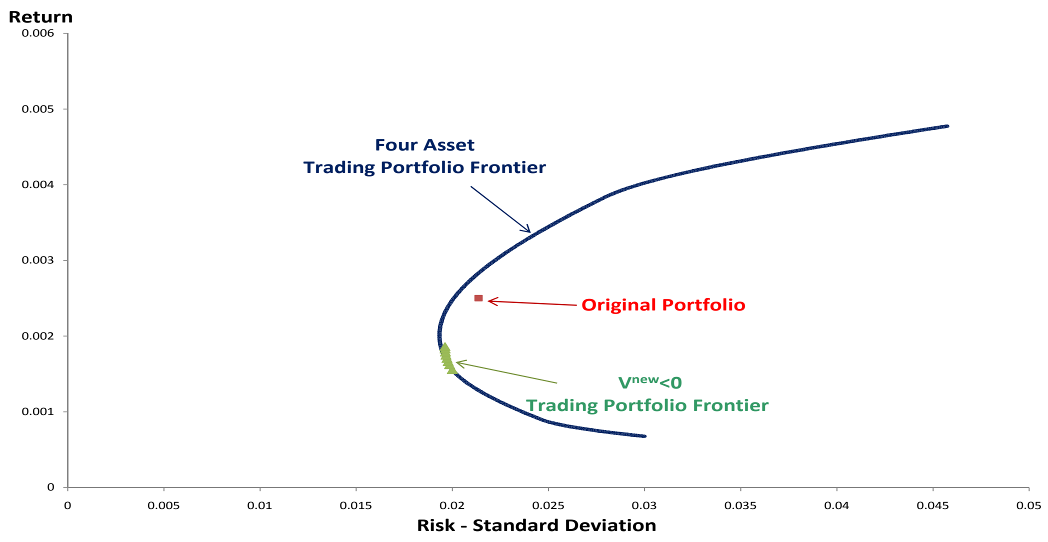

Figure 3.6: A Four Asset Example Trading Portfolio Frontier for $V^{new} < 0$

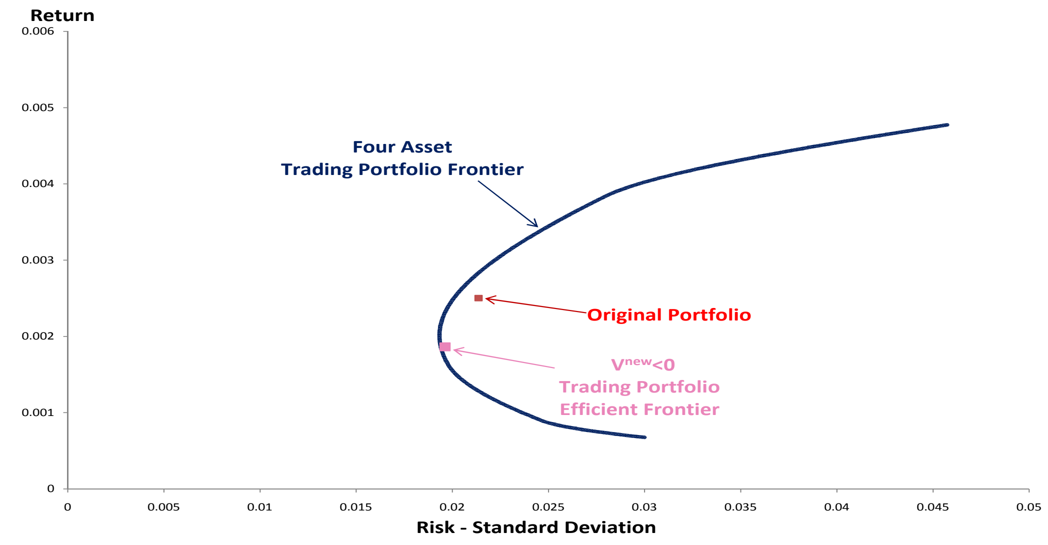

Figure 3.7: A Four Asset Example Trading Portfolio Efficient Frontier for $V^{new} < 0$

(c) A portfolio at a return level C in Figure 3.8 could be achieved if the present portfolio is rebalanced with no exogenous cash investments (so $V^{new} = 0$).

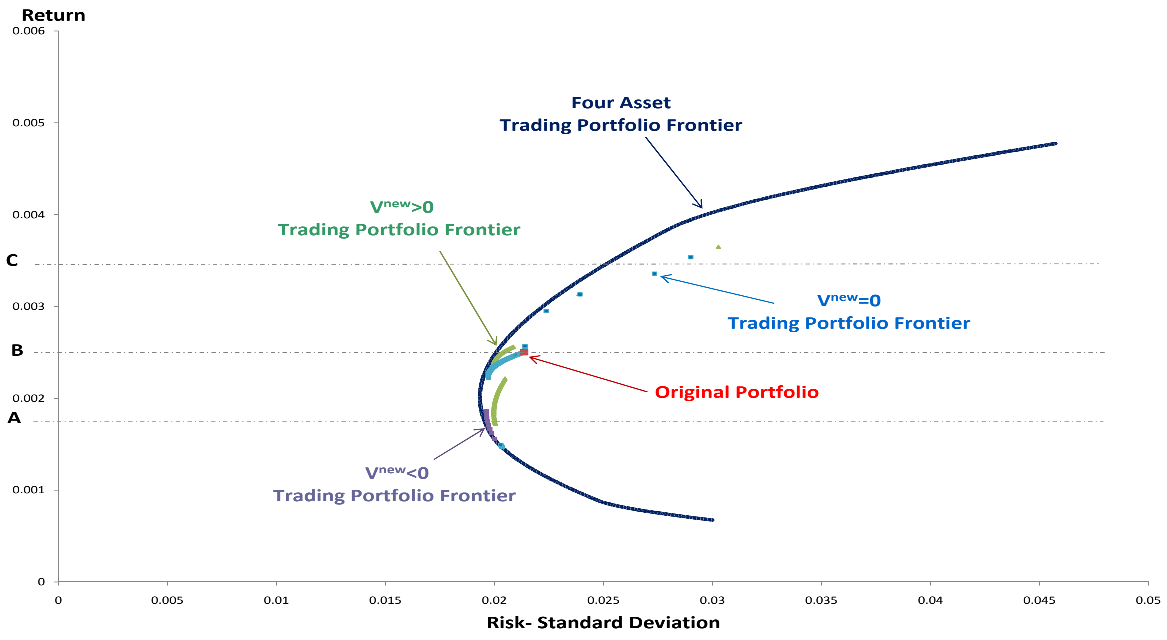

Figure 3.8: A Four Asset Example with $V^{new} = 0$, $V^{new} > 0$ and $V^{new} < 0$

3.7 Non-Cardinality Constrained Transaction Cost Frontiers

In this Section, we present the non-cardinality constrained transaction cost trading portfolio frontier. By this we mean a trading portfolio frontier from an original portfolio (containing 10 randomly generated assets) in which any asset can be included to create a more efficient portfolio. These frontiers have not been seen before in the literature; researchers have focused their work on a single portfolio (for example minimum variance with transaction cost) not an efficient frontier. For these frontiers, we use $V^{new} = 0$, $l_i = 0$ and $u_i = 1$ based on the transaction cost model: minimise equation (3.12) subject to equations (3.13)-(3.24) and (3.30)-(3.34). We ran the Hang Seng, DAX 100, FTSE 100, S&P 100, the Nikkei 225 and S&P 500 Market Indices' data set for 1000 equally spaced return levels from R_{min} to R_{max}.

In Table 3.5 we state the market indices, the number of assets, the computational times in seconds taken to create the non-cardinality constrained transaction cost trading portfolio frontier and the average time (in seconds) per return level. From Table 3.5 we see for instance that the FTSE 100 having $N = 89$ assets took a total of 1273 seconds to calculate which is an average computational time per return level of 1.273 seconds.

Market Index	N	Total Computational Time (seconds)	Average time (seconds) per return level
Hang Seng	31	70	0.070
DAX 100	85	271	0.271
FTSE 100	89	1273	1.273
S&P 100	98	1309	1.309
Nikkei 225	225	1490	1.490
S&P 500	468	6016	6.016

Table 3.2: Non-Cardinality Constrained Transaction Cost Trading Portfolio Frontier Times

In Figures 3.9 to 3.20 we illustrate the original portfolio with the non-cardinality constrained transaction cost trading portfolio frontier and non-cardinality constrained transaction cost efficient frontier (respectively) for the Hang Seng, DAX 100, FTSE 100, S&P 100, Nikkei 225 and S&P 500 market indices respectively. In Figures 3.19 and 3.20 the efficient frontier for the S&P 500 market index is shown rather than the trading portfolio frontier because of numeric issues in AMPL (due to the size of the problem the solver was unable to solve the problem, which resulted in AMPL crashing). Figures 3.9 to 3.20 show that the trading portfolio frontier and the efficient frontiers to be discontinuous and in some cases contains clusters of portfolios. Here and elsewhere in this thesis, we give the first two figures as full page figures to aid the reader, other figures are smaller for reasons of space.

In Table 3.3 we present for each of our data sets: the mean, median, maximum and minimum percentage errors for the non-cardinality constrained efficient frontier (which is different from the UEF). From this Table we see that the FTSE 100 market index has the smallest mean and median percentage error (1.4661% and 1.3628% respectively). Each of the market indices had minimum percentage errors below 6% with the S&P 100 having a minimum percentage error as low as 0.0188%. The highest maximum percentage error came from the S&P 500 being 26.4100%.

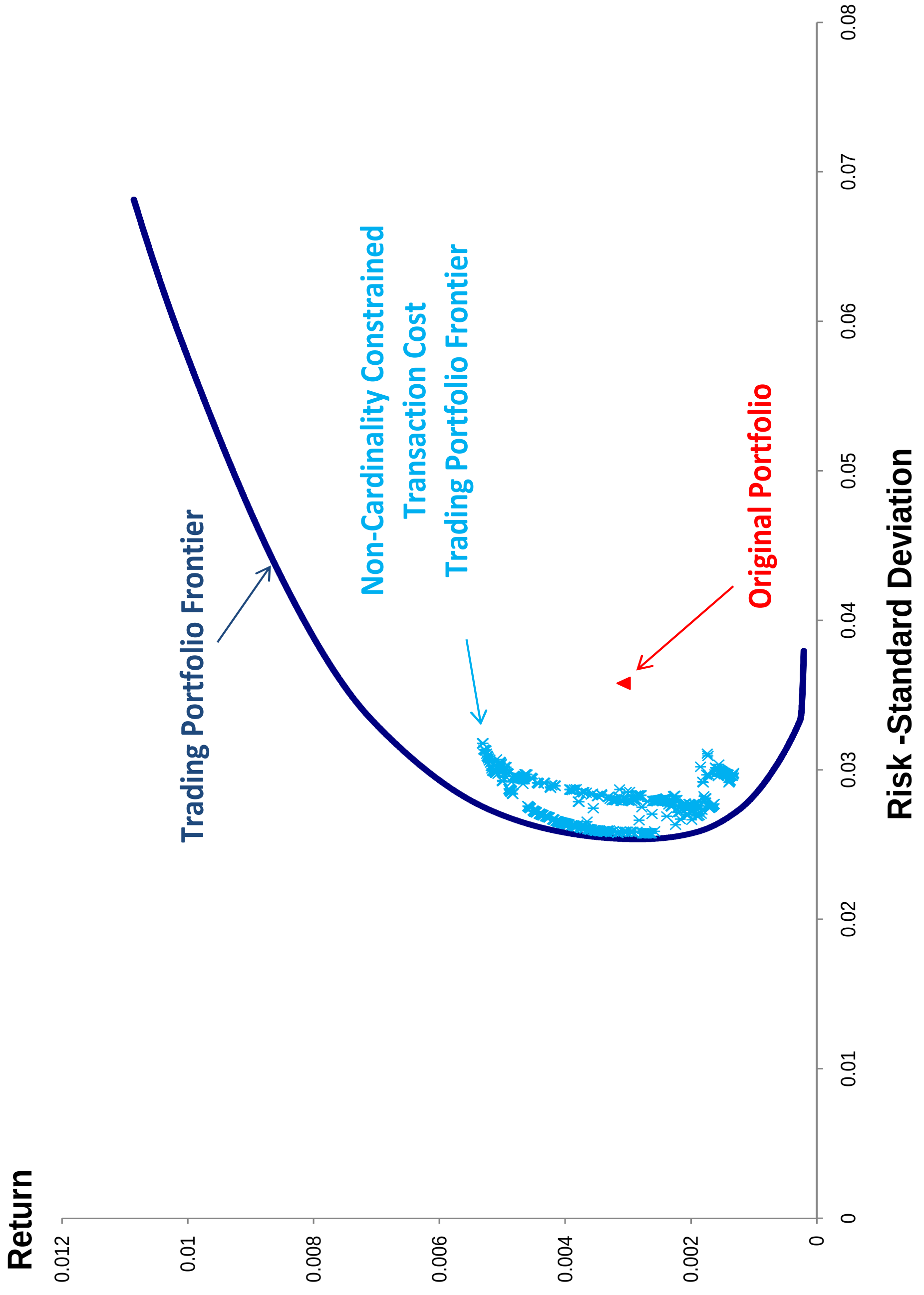

Figure 3.9: Hang Seng Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

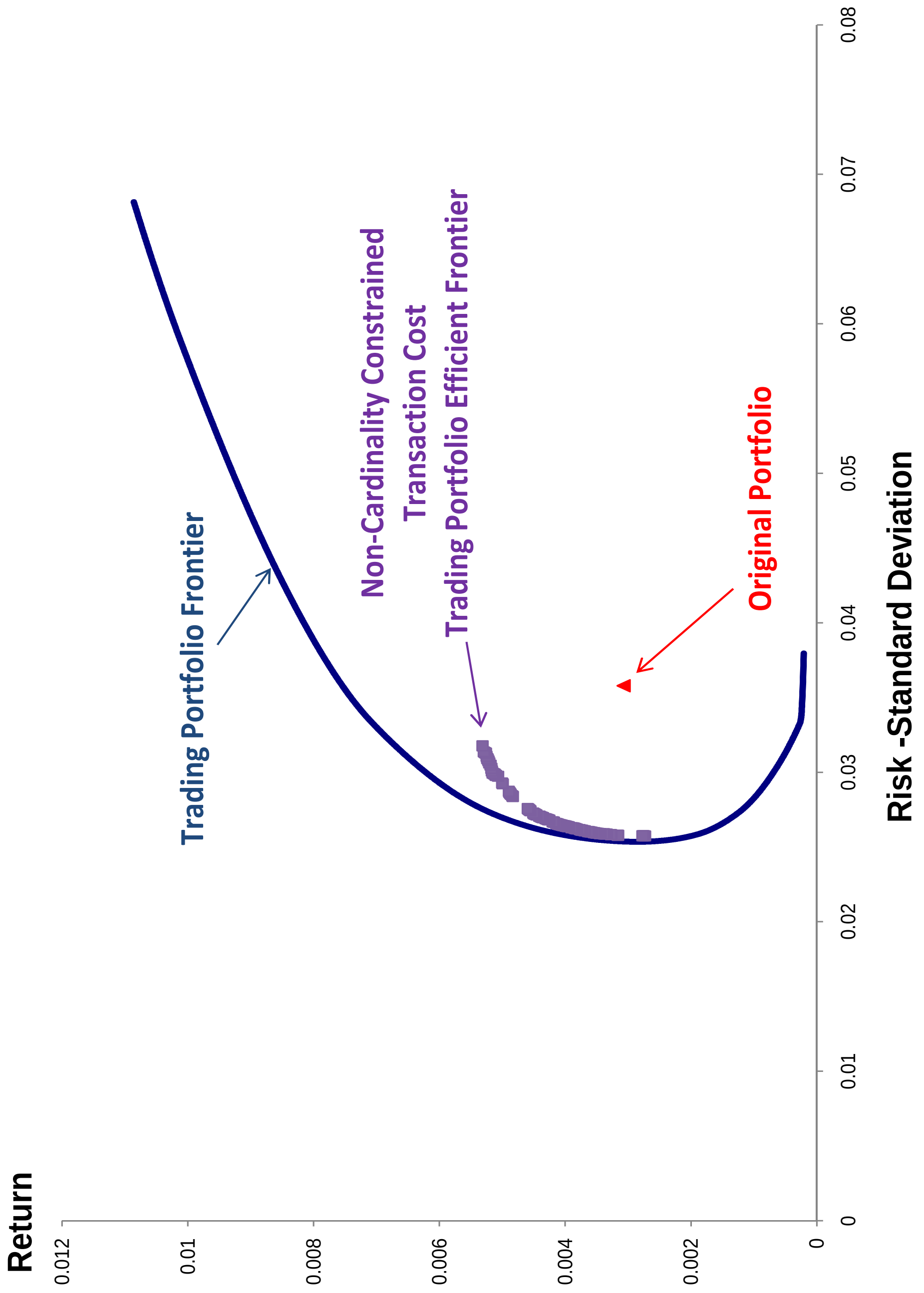

Figure 3.10: Hang Seng Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

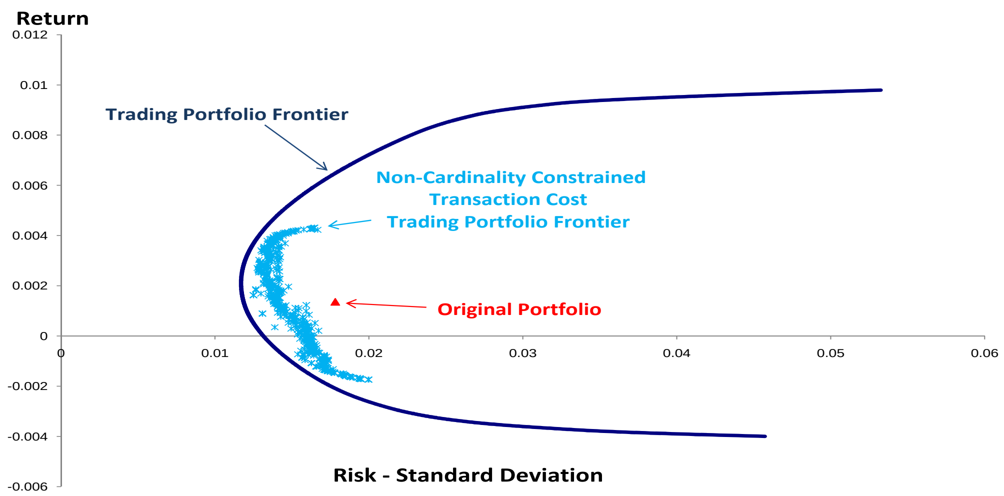

Figure 3.11: DAX 100 Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

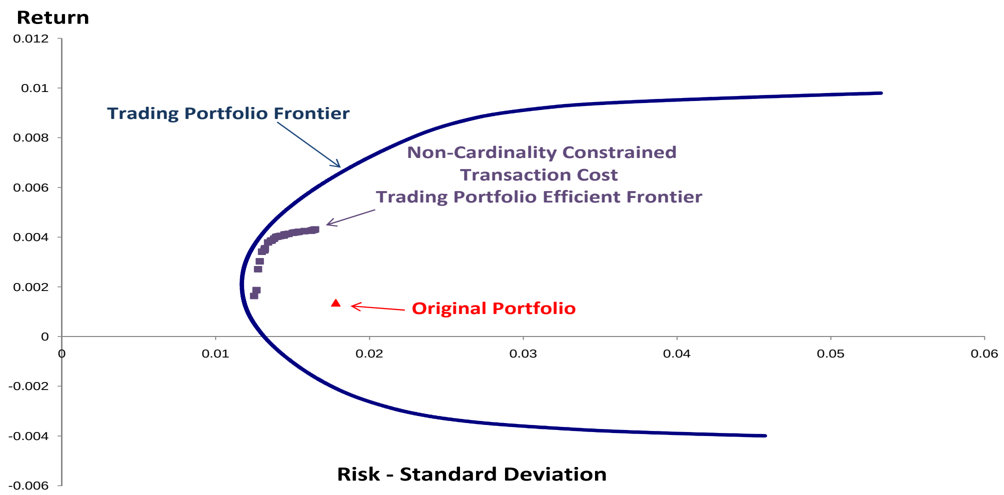

Figure 3.12: DAX 100 Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

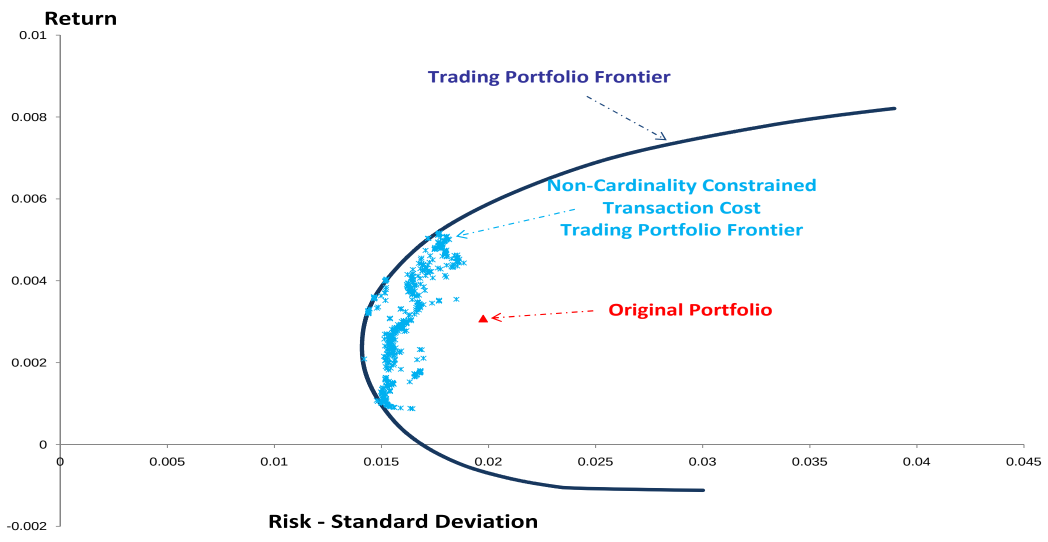

Figure 3.13: FTSE 100 Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

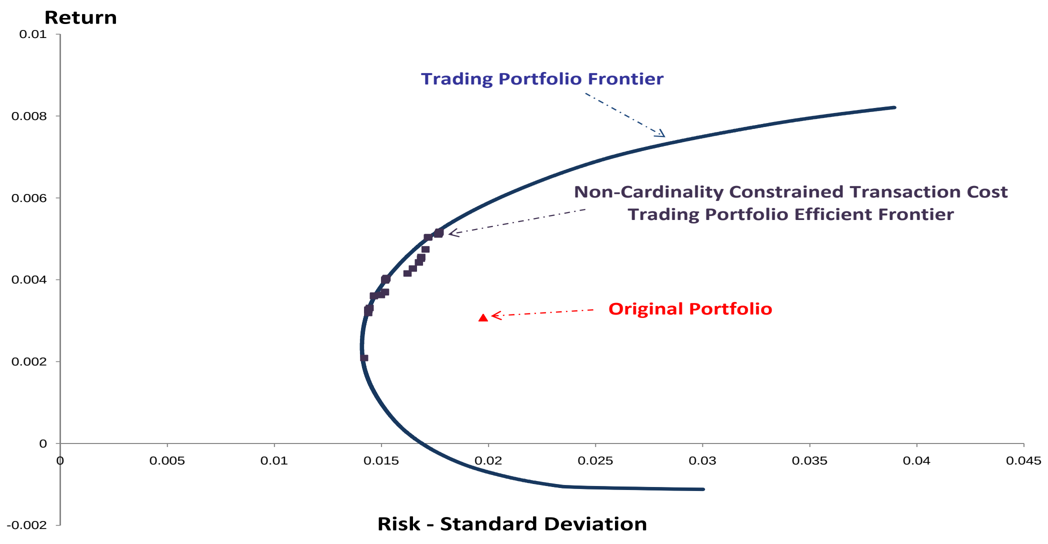

Figure 3.14: FTSE 100 Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

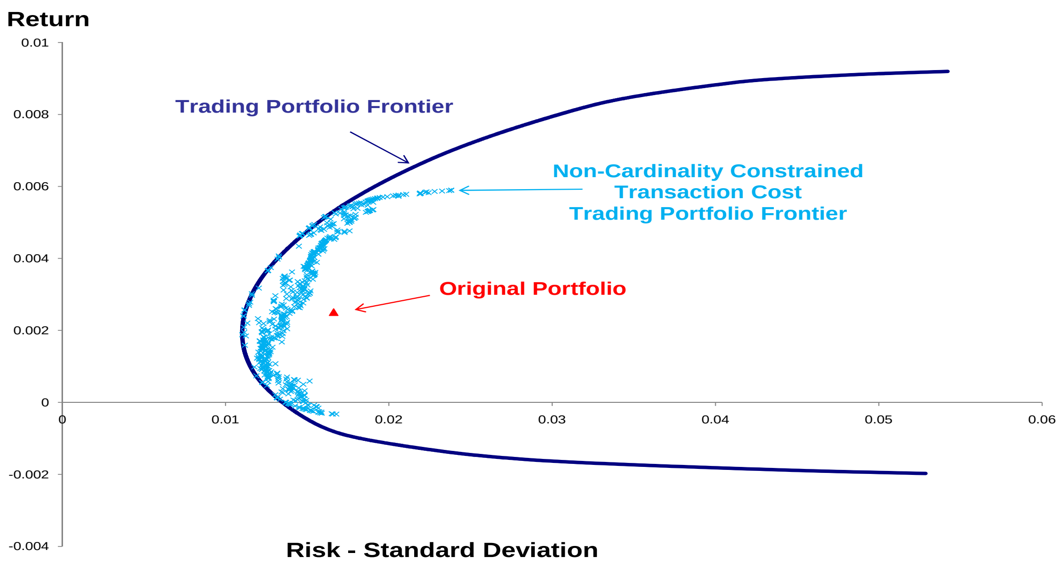

Figure 3.15: S&P 100 Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

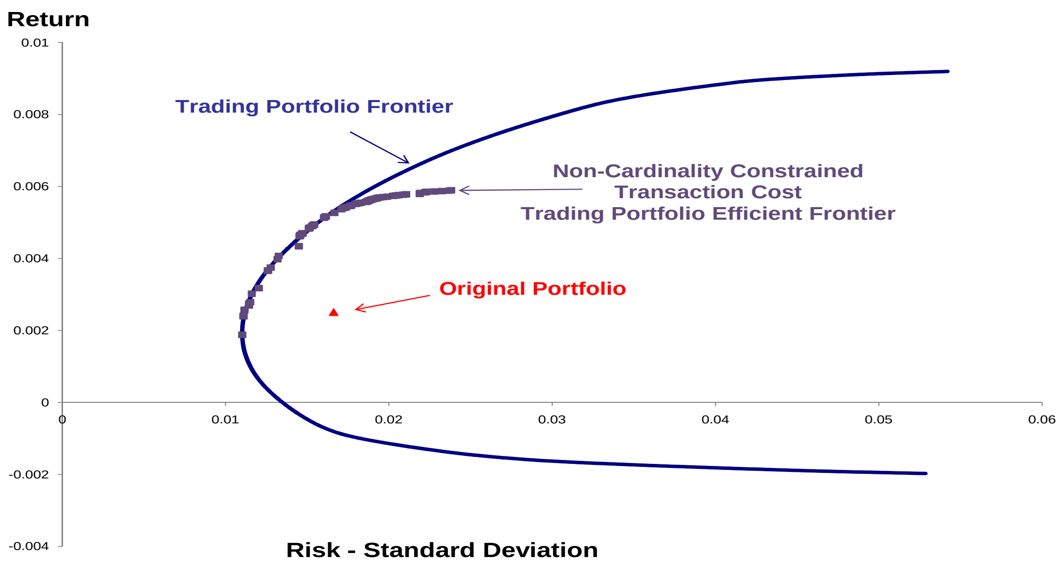

Figure 3.16: S&P 100 Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

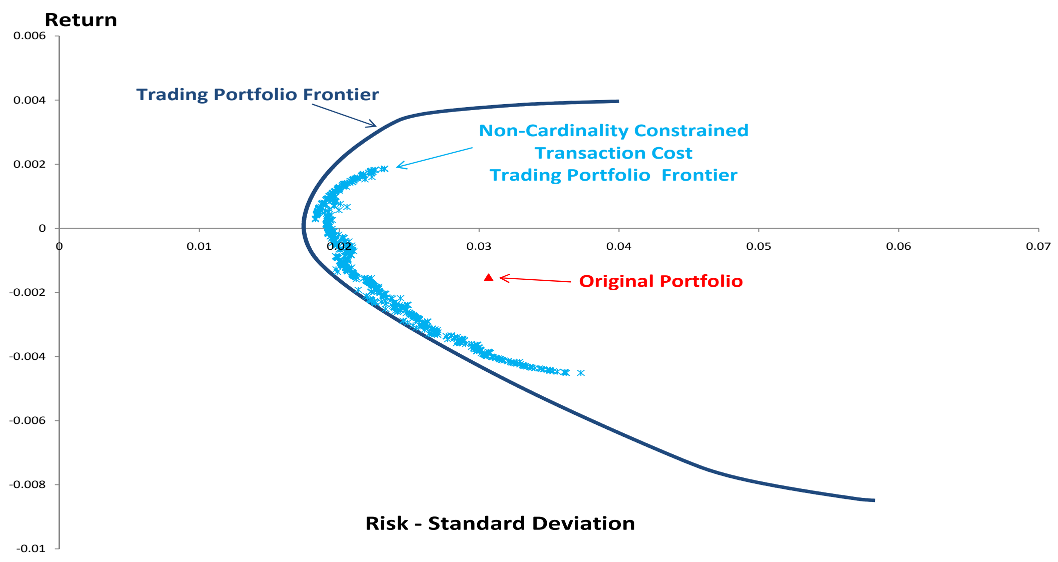

Figure 3.17: Nikkei 225 Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

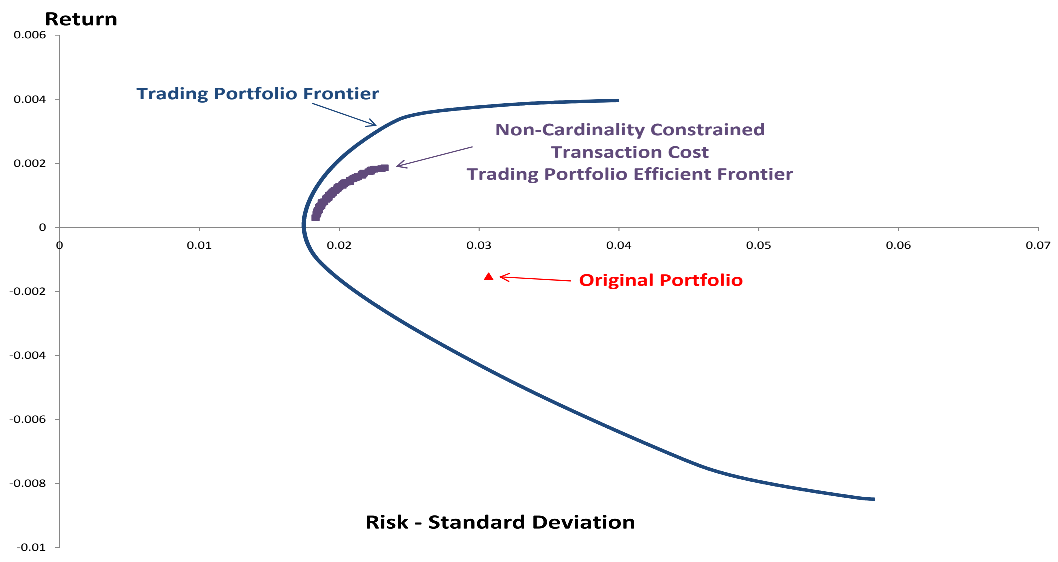

Figure 3.18: Nikkei 225 Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

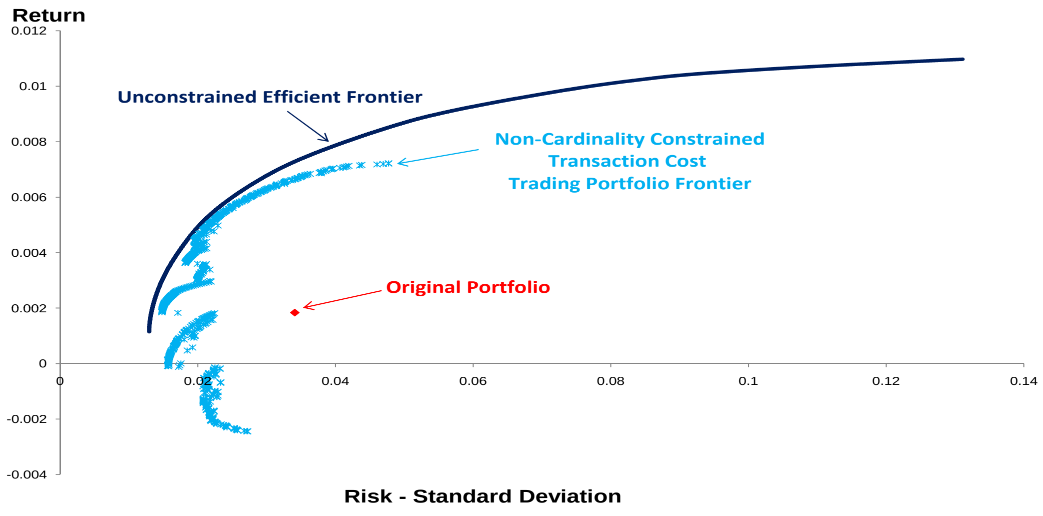

Figure 3.19: S&P 500 Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

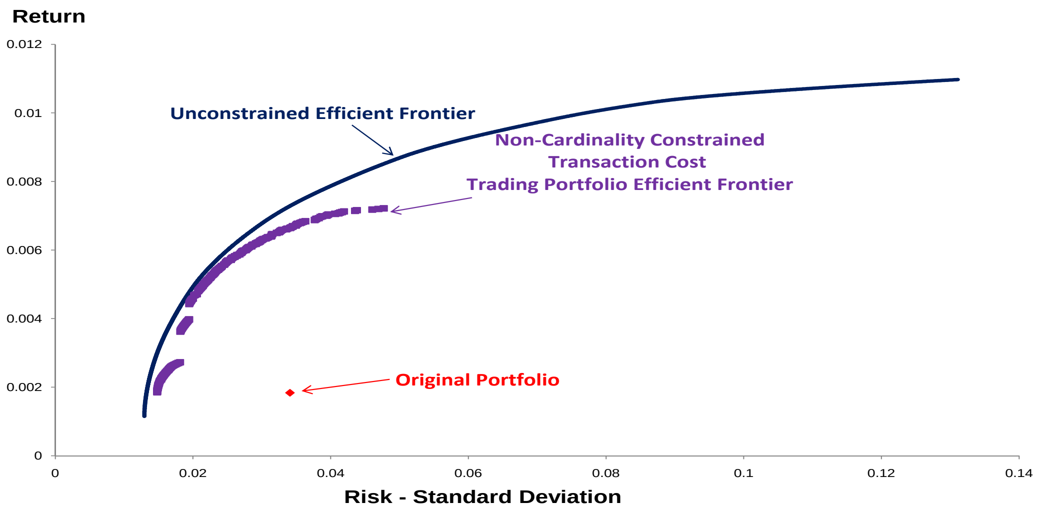

Figure 3.20: S&P 500 Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

Percentage Error	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	S&P 500 $N = 457$	Average all problems
Mean	3.9342	11.4965	1.4661	4.0098	8.7509	9.5623	6.5366
Median	2.2752	8.7332	1.3628	2.1712	7.5653	8.1039	5.0353
Minimum	0.1071	3.8460	0.2902	0.0188	0.1716	5.2874	
Maximum	14.8037	23.6433	3.8941	15.7078	19.5221	26.4100	

Table 3.3: Percentage Error Results for the Transaction Cost Non-Cardinality Constrained Efficient Frontier

Percentage Error	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	Average all problems
Mean	10.9868	11.4983	5.4564	11.1342	17.5140	11.3161
Median	9.8534	11.2493	4.7768	11.5564	16.8255	10.8523
Minimum	3.0922	8.7803	0.1874	6.8437	12.5286	
Maximum	19.5885	14.6783	11.5384	16.0530	22.9826	

Table 3.4: Percentage Error Results for the Transaction Cost Cardinality Constrained Efficient Frontier

3.8 Cardinality Constrained Transaction Cost Frontiers

In this Section, we present the cardinality constrained transaction cost trading portfolio frontier to answer the question: *What does the optimal solution for the cardinality constrained transaction cost optimisation model look like?*. To create these frontiers we began with the same randomly generated portfolio used in the non-cardianlity constrained transaction cost frontiers section (Section 3.7), using $K = 10$, $V^{new} = 0$, $l_i = 0$ and $u_i = 1$ based on the transaction cost model: minimise equation (3.35) subject to equations (3.36)-(3.46) and equations (3.50)-(3.57). We ran the Hang Seng, DAX 100, FTSE 100, S&P 100 and the Nikkei 225 Market Indices' data set for 50 equally spaced return levels from R_{min} to R_{max}. As was the case with the non-cardinality constrained problem, these frontiers have not been seen before in the literature.

Market Index	N	Total Computational Time (seconds)	Average time (seconds) per return level
Hang Seng	31	72	1.44
DAX 100	85	348	6.96
FTSE 100	89	3426	68.52
S&P 100	98	127746	2554.92
Nikkei 225	225	566537	11330.74

Table 3.5: Cardinality Constrained Transaction Cost Trading Portfolio Frontier Times

In Table 3.5 we state the market indices, the number of assets, the computational times in seconds taken to create the cardinality constrained transaction cost trading portfolio frontier and the average time (in seconds) per return level. Therefore, the Nikkei 225 which contains 225 assets took a total of 566537 seconds (almost 7 days) which gives an average computational time per return level of 11330.74 seconds (a little more than 3 hours).

If we compare the average time per return levels in Tables 3.2 and 3.5, we see that the non-cardinality constrained case is calculated more quickly per return level. For instance, the S&P 100 takes 1.309 seconds per return level in the non-cardinality case while it takes 2554.92 seconds per return level in the the cardinality constrained case which is approximately 1952 times longer.

Table 3.4 shows the mean, median, maximum and minimum percentage errors for the cardinality constrained efficient frontier for the Hang Seng, DAX 100, FTSE 100, S&P 100 and the Nikkei 225 Market Indices' data set. If we are to consider a market index in the Table, say the DAX 100, we see that the mean, median, minimum and maximum percentage error was 11.4983%, 11.2493%, 8.7803% and 14.6783% respectively.

Considering both Tables 3.3 and 3.4 we note that all percentage errors (mean, median, minimum and maximum) are increased when one considers cardinality constraints. This would be expected because the non-cardinality constrained efficient frontier places no restrictions on the number of assets in a portfolio when moving from the considered original portfolio.

In Figures 3.21 to 3.30 we illustrate the original portfolio, cardinality constrained transaction cost trading portfolio frontier then the cardinality constrained transaction cost efficient frontier along with the trading portfolio frontier for the Hang Seng, DAX 100, FTSE 100, S&P 100 and Nikkei 225 market indices respectively. Each Figure shows that the frontiers (portfolio and efficient) are discontinuous.

3.9 Summary

This Chapter presented optimal solutions for transaction cost model. We first presented our formulation of the problem which extended the Markowitz model to include fixed and variable cost. In that Section we included an explanation of the benefits of our formulation.

In Section 3.3 we presented the cardinality extension to our formulation in which we placed restrictions on the assets in the portfolio, as well as those assets bought, sold or traded. We went on to give more equations that would be necessary for revising an existing portfolio. Then we highlighted the benefits of our cardinality constrained transaction cost formulation.

In Section 3.4 we presented some computational considerations which can contribute to better computational times. In Section 3.5 we gave the data sets for our test problems. In Section 3.6 we spoke of the transaction cost frontiers. We showed not only that the transaction cost efficient frontier was discontinuous but, fixed transaction costs (in themselves) were suf-

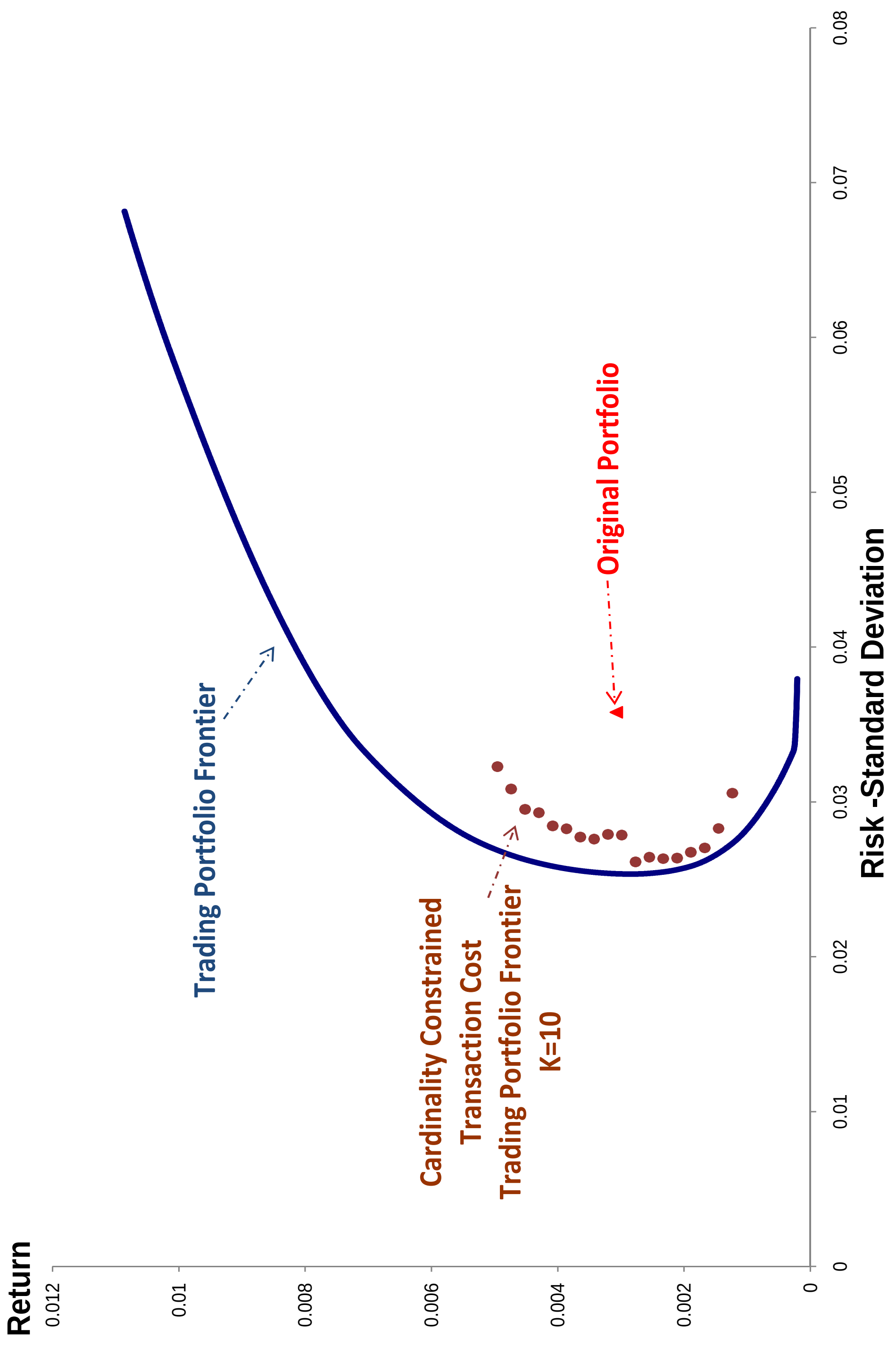

Figure 3.21: Hang Seng Transaction Cost Cardinality Constrained Trading Portfolio Frontier

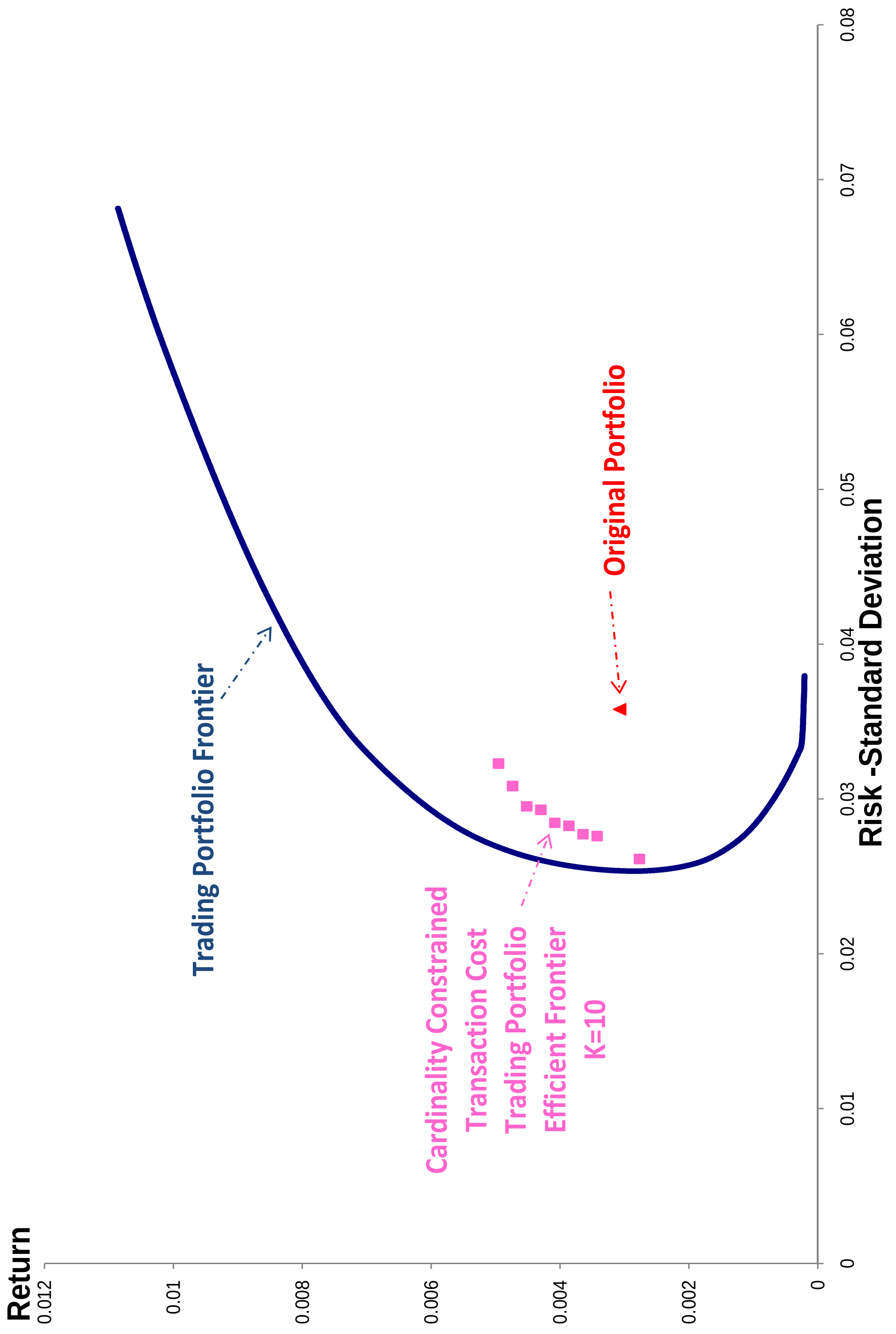

Figure 3.22: Hang Seng Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

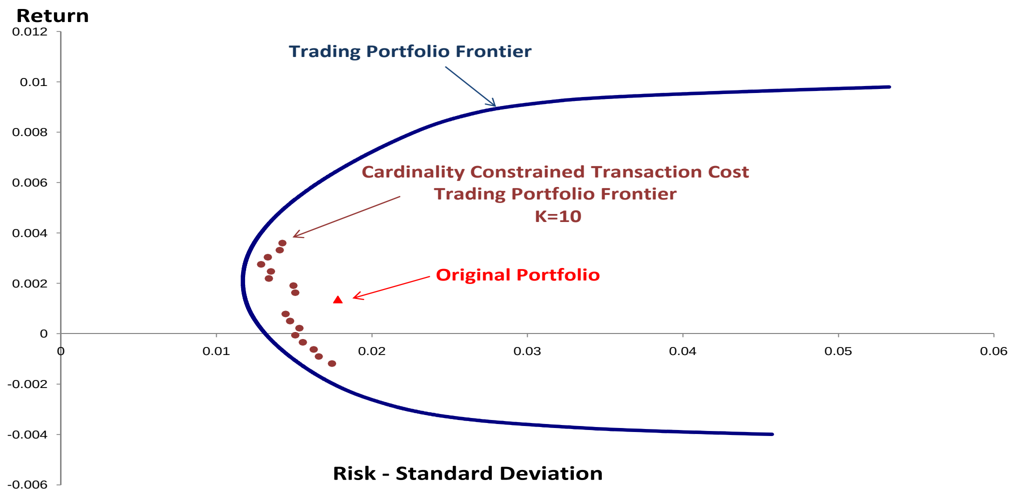

Figure 3.23: DAX 100 Transaction Cost Cardinality Constrained Trading Portfolio Frontier

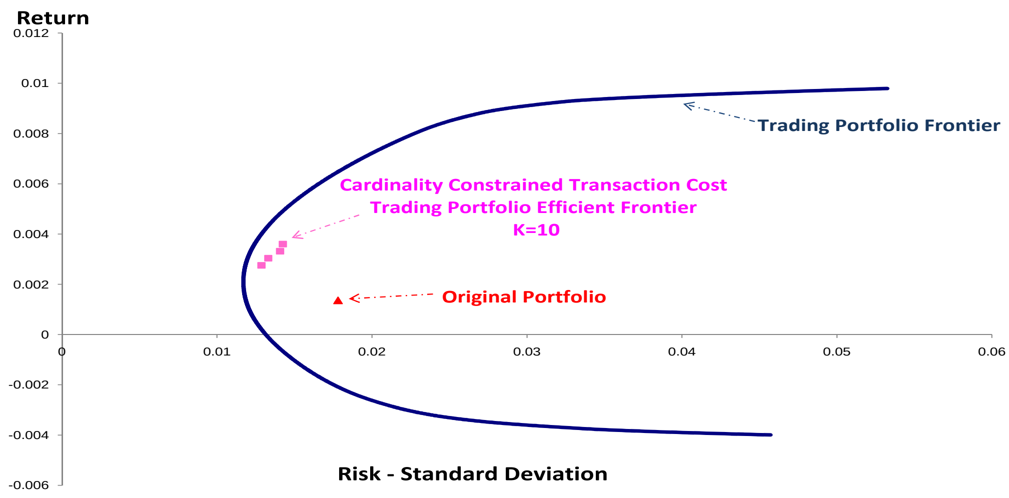

Figure 3.24: DAX 100 Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

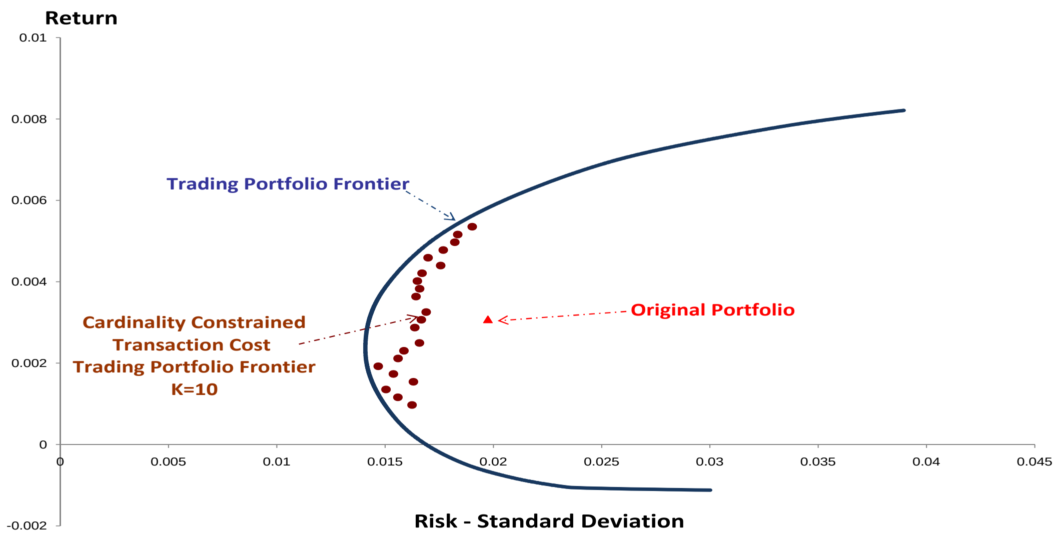

Figure 3.25: FTSE 100 Transaction Cost Cardinality Constrained Trading Portfolio Frontier

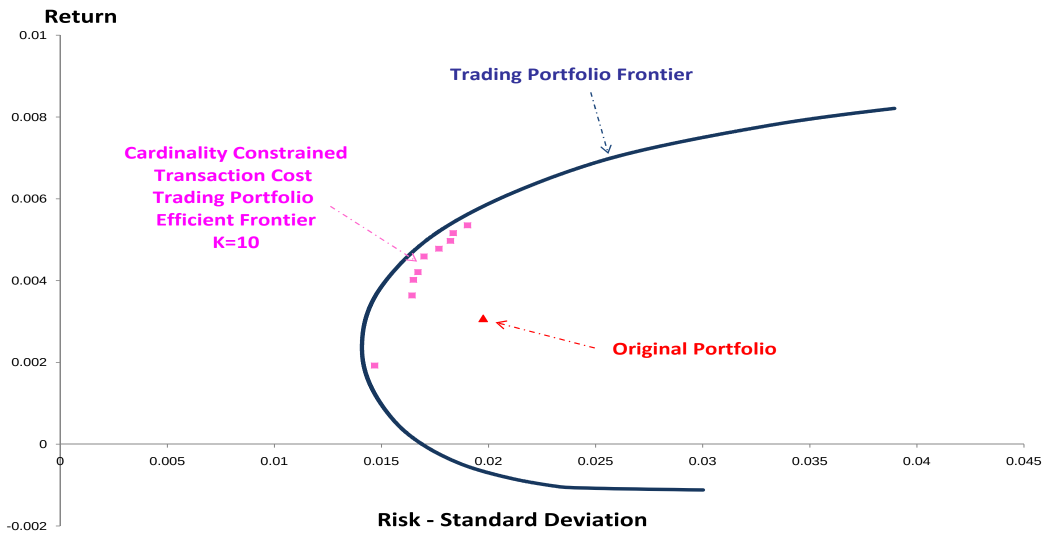

Figure 3.26: FTSE 100 Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

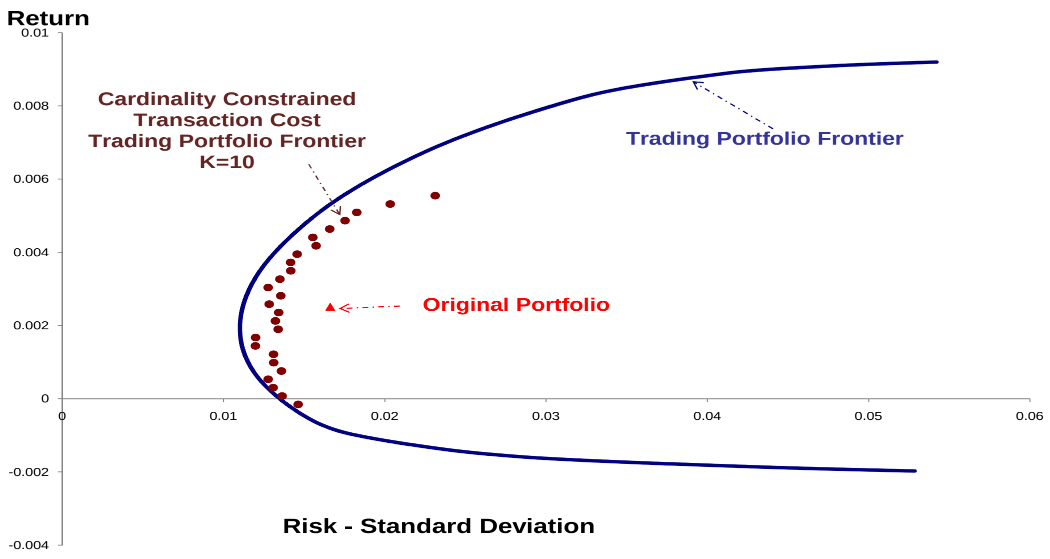

Figure 3.27: S&P 100 Transaction Cost Cardinality Constrained Trading Portfolio Frontier

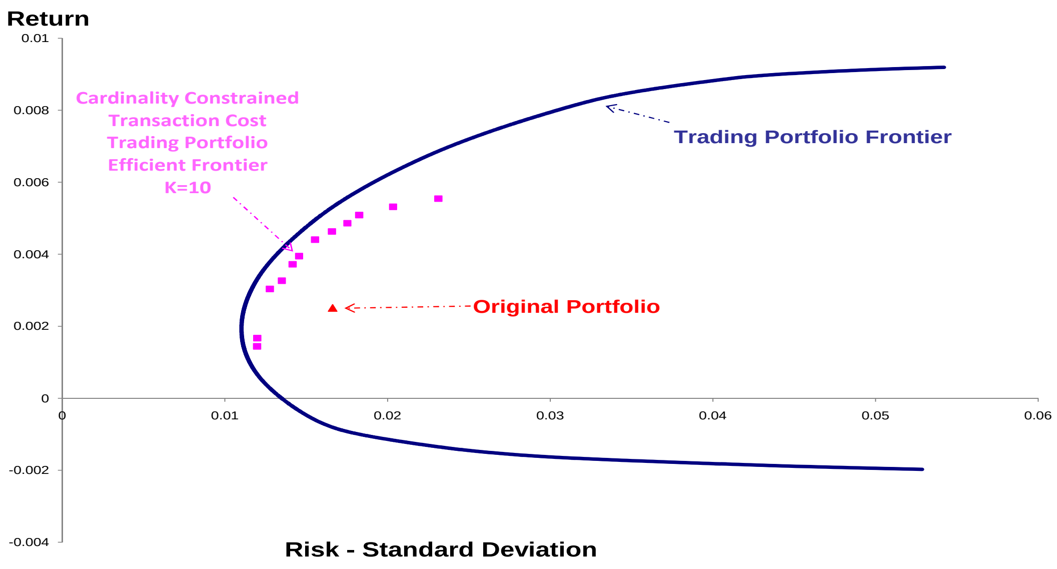

Figure 3.28: S&P 100 Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

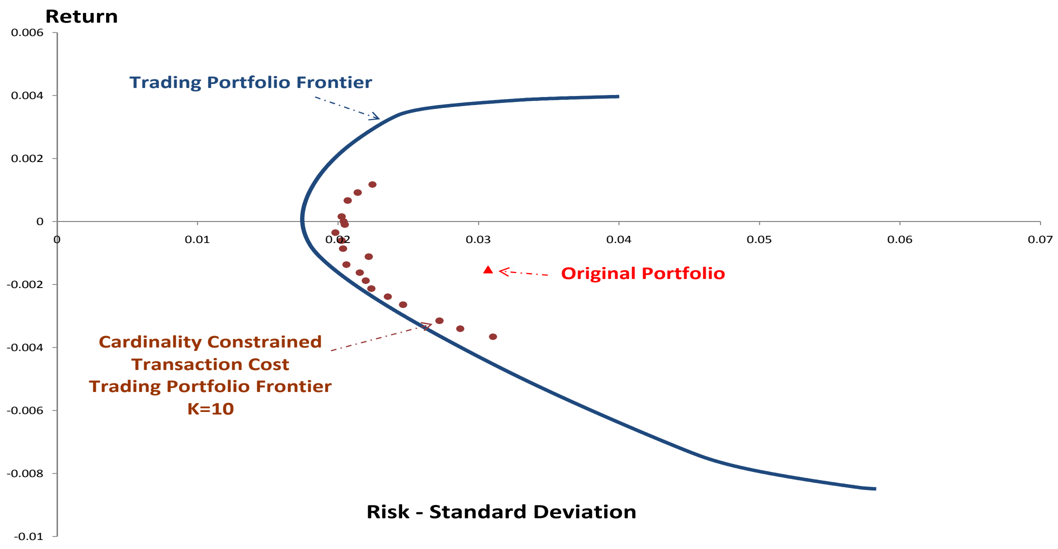

Figure 3.29: Nikkei 225 Transaction Cost Cardinality Constrained Trading Portfolio Frontier

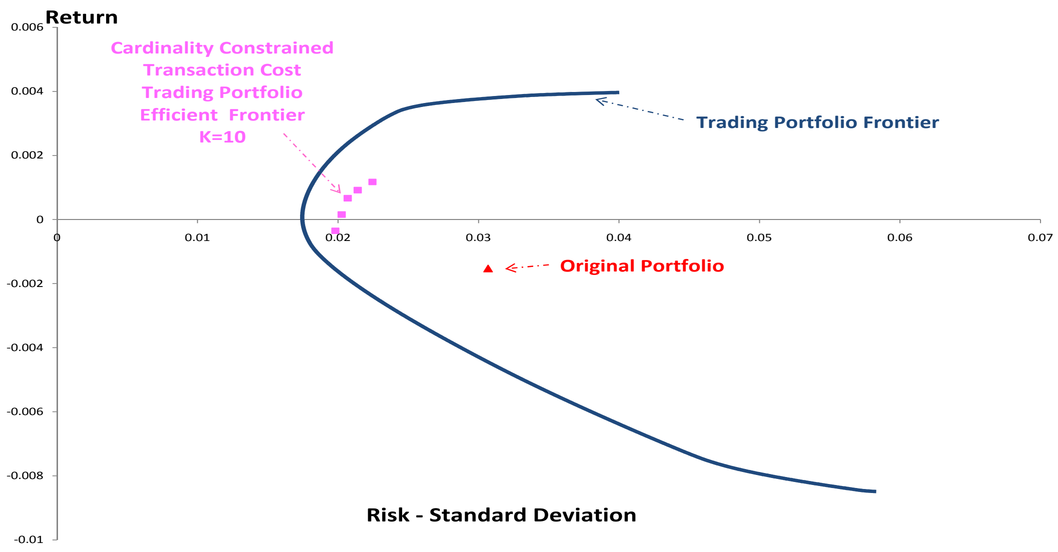

Figure 3.30: Nikkei 225 Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

ficient to make the efficient frontier discontinuous even when there are no explicit cardinality restrictions. We presented a four asset example that showed positive, negative and zero cash investments causes discontinuities in the trading portfolio and efficient frontiers.

In Sections 3.7 and 3.8 we considered the non-cardinality and cardinality constrained transaction cost frontiers. In these two sections we showed that, given an original portfolio, each market index has a unique trading portfolio and efficient frontier. The non-cardinality constrained trading portfolio frontier (unlike the trading portfolio frontier of the market index) is not a smooth parabola and it could contain clusters. The non-cardinality constrained trading portfolio frontier is capable of being solved in a quicker time frame and the non-cardinality constrained has a smaller percentage error from its unconstrained efficient frontier than the transaction cost cardinality constrained case.

Chapter 4

Heuristic Algorithms for the CCEF

4.1 Introduction

As billions of dollars (pounds sterling) are invested in markets around the world, investors must not only consider maximising their expected return, but also minimising the volatility that results from expected fluctuations in the value of their investment portfolios. Increasingly, portfolio managers are seeking more robust asset selection (portfolio formation) strategies to create desirable portfolios (ones that potentially gives a good tradeoff between investment risk and return) for their investors. The Markowitz MV model introduced in Section 2.2 serves as a major guideline for financial portfolio optimisation. However, in order to incorporate real market situations, it has become necessary to introduce discrete constraints such as buy-in threshold and a cardinality constraint. These two discrete constraints allows the investor to specify the limits on the proportions and the number of assets in their portfolios. In this Chapter, we present our heuristic algorithms for determining the CCEF.

This Chapter is organised as follows. In Section 4.2, we consider the optimisation problem (denoted the subset optimisation problem) that underlies each of our heuristic algorithms. In Section 4.3, our implementation of the heuristic algorithms (of GA, TS and SA) for finding the CCEF is presented. This is followed by Section 4.4 where we present the computational results for our heuristic algorithms and finally we conclude the Chapter in Section 4.5 with a summary.

4.2 Subset Optimisation

In this thesis, the heuristic algorithms we present make use of subset optimisation. This is an optimisation model that allows us to specify subsets of assets for which we know their status (either in or out of the chosen portfolio). We accomplish this by introducing S_{in} as a subset of assets that must be included in the chosen portfolio, and S_{out} as the subset of assets that must be excluded from the chosen portfolio. S_{in} and S_{out} have no assets in common, thus $S_{in} \cap S_{out} = \emptyset$. Given these subsets we optimise for any remaining assets to determine if they are to be included in, or excluded from, the chosen portfolio. For every asset i $(i = 1, \ldots, N)$ in the portfolio the proportion invested in that asset i is decided by the solver through the subset optimisation process.

Early computational experience indicated that attempting to find a portfolio with precisely K assets and precise return R was relatively time-consuming, even if the number of assets from which we were choosing was small. Consequently, in the subset optimisation problem (equations (4.1)-(4.8), below) we relax the desired return constraint (i.e. equation (2.2), $\sum_{i=1}^{N} \mu_i x_i = R$) such that desired return is allowed to be in a specific range as opposed to the return being precisely specified. As a result, we are content with a portfolio bounded by a lower limit on return, R_L, and an upper limit on return, R_U.

The notation used in the subset optimisation problem is common with that of the previous two chapters, therefore we have have chosen not to give them again because of space.

The subset optimisation problem that we solve is:

$$\text{Minimise} \sum_{i=1}^{N} \sum_{j=1}^{N} \sigma_{ij} w_i w_j \tag{4.1}$$

subject to

$$R_L \leqslant \sum_{i=1}^{N} \mu_i w_i \leqslant R_U \tag{4.2}$$

$$\sum_{i=1}^{N} w_i = 1 \tag{4.3}$$

$$l_i\delta_i \leqslant w_i \leqslant u_i\delta_i \qquad i = 1, \ldots, N \tag{4.4}$$

$$\sum_{i=1}^{N} \delta_i = K \tag{4.5}$$

$$\sum_{i \in S_{in}} \delta_i = \min[|S_{in}|, K] \tag{4.6}$$

$$\delta_i = 0 \qquad \forall i \in S_{out} \tag{4.7}$$

$$\delta_i = 0 \,\text{or}\, 1 \qquad i = 1, \ldots, N \tag{4.8}$$

$$w_i \geq 0 \qquad i = 1, \ldots, N \tag{4.9}$$

Equations (4.1), (4.3)-(4.5) and (4.9) are as in the cardinality constrained model (equations (2.1)-(2.4), and equations (2.21)-(2.23)). Equation (4.2) constrains the chosen portfolio's expected return to be within the desired return range, $[R_L,R_U]$. Equation (4.6) forces all assets in S_{in} into the portfolio if $|S_{in}| \leqslant K$, and chooses K assets from S_{in} if $|S_{in}| > K$. Equation (4.7) ensures that assets in S_{out} are not placed in the chosen portfolio while, equation (4.8) declares that the binary variables are zero or one.

This subset optimisation problem (equations (4.1)-(4.9)) like the cardinality constrained model (equations (2.1)-(2.4), and equations (2.21)-(2.23)) is a QMIP, and provided that the number of assets for which we have to make a decision as to whether they are to be included in or excluded from the chosen portfolio is small (i.e. $N - |S_{in} \cup S_{out}|$ is small) it can be solved relatively quickly to proven optimality.

For simplicity of notation in the heuristic algorithms we present in Section 4.3 we refer to the subset optimisation problem (equations (4.1)-(4.9)) as $\mathrm{F}(S_{in}, S_{out})$. In the computational results reported later in Section 4.4 we use both $[R_L = 0.9R, R_U = 1.1R]$, i.e. a portfolio within ten percent of the desired return level, and $[R_L = -\infty, R_U = +\infty]$, where we disregard desired return.

4.3 Heuristic Algorithms

In this Section, we present our implementation of the three heuristic algorithms based upon genetic algorithm (Section 4.3.1), tabu search (Section 4.3.3) and simulated annealing (Section 4.3.4) that we have developed for finding the CCEF. Also included in this Section is an illustrative example as to how the GA heuristic algorithm works (Section 4.3.2).

Here we note that one of the potential practical advantages of our heuristic algorithm implementations is that any additional (user specified) constraints on the composition of the chosen portfolio can be included in the subset optimisation problem. Such constraints might include, for example:

- class or sector constraints which specify minimum or maximum exposure to certain sectors (sets of assets),
- lot size constraints which specify that the amount invested, and in any asset must be an integer multiplier of a known constant.

The heuristic algorithms outlined below are applicable, without significant change, to problems of these types.

4.3.1 A Genetic Algorithm Heuristic

In our GA we use a population, P, of fixed size $|P| = 100$, i.e we have 100 portfolios. Given the desired return of R, each member of the initial population is generated by randomly choosing $\max[2K, 20]$ assets to be in S_{in}, with the other $[1, ..., N] - S_{in}$ assets being in S_{out} and then solving the subset optimisation problem $\mathrm{F}(S_{in}, S_{out})$. In order to try and ensure that the subset optimisation problem is feasible in making a random choice of assets, we include in S_{in} some assets i $(i = 1, ..., N)$ that have expected return greater than or equal to the desired return, i.e. $\mu_i \geq R$, and some assets i $(i = 1, ..., N)$ that have expected return less than or equal to the desired return, i.e. $\mu_i \leq R$.

In our GA we use parent sets. We first select two parents sets, Q_1 and Q_2 (each of fixed size q, in our computational results presented later in Section 4.4 we use $q = 5$). We create the parent sets by sorting the members of the population into increasing risk (variance) order. Take the first $2q$ portfolios in this ordered list and assign the first portfolio to Q_1, the second to Q_2, the third to Q_1, etc in an alternate fashion. These two sets collectively contain the $2q$ fittest members of the population (i.e. they represent the lowest risk portfolios).

In order to produce children (offspring) we consider all pairs of portfolios, one portfolio from Q_1, the other from Q_2, hence in the crossover operator we produce q^2 parent portfolio pairs in total. For each parent portfolio pair a single child is produced using crossover. In our crossover procedure if an asset is present in both of the parent portfolios it is present in the child (and therefore in S_{in}); if it is absent from both of the parent portfolios it is absent in the child (and so in S_{out}); if it is present in one of the parent portfolios (absent in the other) then its presence (or absence) in the child will be decided as a result of the subset optimisation process.

Mutation is standard within GAs and introduces a degree of stochastic variation. We employ it to alter offspring portfolios with a very low probability. Offspring portfolios are subject to mutation with a probability of m_p. Our GA mutates a child by randomly selecting one asset in the child portfolio and replacing it by a random asset not present in the child portfolio. This process is applied in generation g^* to each child portfolio. In our computational results presented later in Section 4.4 we ran our GA heuristic algorithm for four generations, with mutation occurring in the third generation (i.e. $g^* = 3$) with $m_p = 0.03$.

Each child (for which the sets S_{in} and S_{out} have been decided after crossover and mutation) is optimised by solving $\mathrm{F}(S_{in}, S_{out})$. Note here that we cannot guarantee that we get a feasible solution when we solve this subset optimisation problem, i.e. it is possible that there is no feasible child given the choice that has been made of S_{in} and S_{out} via crossover and mutation. In the event that the offspring solution is infeasible it is disregarded.

In our GA to generate a new population we combine the $|P|$ members of the current population with the set of feasible children, sort the portfolios in this combined set into increasing risk (variance) order and take the first $|P|$ members of this ordered list to constitute the new population for the next generation. At the end of the GA process the $|P|$ portfolios in the final population contribute to the cardinality constrained efficient frontier (though note here that we do eliminate at this stage any portfolios that are dominated by others in the final population). Our GA heuristic algorithm is given in psuedocode in Algorithm 1.

R_{min}	be the minimum expected return for all assets, thus $R_{min} = \min[\mu_i \| 1, ..., N]$
R_{max}	be the maximum expected return for all assets, thus $R_{max} = \max[\mu_i \| 1, ..., N]$
O_{xy}	be the offspring of $x \in Q_1$ and $y \in Q_2$, and consists of
O^*	be the set of feasible offspring
P^*	be the set of feasible offspring and current generation members
G	the number of iterations

```
begin
for $R := R_{\min}, ..., R_{\max}$ do                /examine $R$ values equally spaced in $[R_{\min}, R_{\max}]$/
$Q_1$, $Q_2$, $O^*$, $P^* := \emptyset$
initialise $P := \{P_1, ..., P_{100}\}$              /random initialisation, $S_{in} = \max[2K, 20]$ assets/
determine $S_{out} := [1, ..., N] - S_{in}\ \forall p \in P$
solve $F(S_{in}, S_{out})\ \forall p \in P$ and sort by variance     /subset optimisation/
    for $g := 1, ..., G$ do                          /G iterations in all/
    select $Q_1, Q_2 \in P$ by selection criteria
        for $x \in Q_1$ and $y \in Q_2$ do           /crossover to produce offspring/
        $S_{in} := \{i \in (1, ..., N) | i \in x \cap y\}$
        $S_{out} := \{i \in (1, ..., N) | i \notin x \cup y\}$
            if $g := g^*$ then                       /mutation/
                for $i \in S_{in}$ and $j \in S_{out}$ do
                $S_{in} := S_{in} \cup [j] - [i]$
                $S_{out} := S_{out} \cup [i] - [j]$
                end for
            end if
        solve $F(S_{in}, S_{out})$                   /subset optimisation/
            if $F(S_{in}, S_{out})$ is feasible then /evaluate solution/
                $O^* := O^* \cup O_{xy}$             /collects feasible offsprings/
            end if
        end for
        $P^* := P \cup O^*$ and sort by variance     /combine offspring with current population/
        $P :=$ first $|P|$ in $P^*$                  /new population/
    end for
end for
end
```

Algorithm 1: GA heuristic algorithm psuedocode

4.3.2 A Genetic Algorithm Heuristic Example

To illustrate how our GA heuristic algorithm minimises risk in a portfolio, we created a ten asset problem. For illustrative purposes we use different parameter values than those given in Section 4.3.1. In this example, we use ten assets (i.e. $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$), an initial population of size 7 with each portfolio containing six assets using our GA heuristic algorithm selection criteria. Those seven portfolios undergo the subset optimisation process where they are reduced to the defined cardinality of $K = 3$, and we determine the portfolio risk (in terms of variance) and portfolio return for the specific R_{min} and R_{max} values.

In Table 4.1, we display our selection results for $R_{min} = 0.003953$ and $R_{max} = 0.004376$, along with the parent set selections (using $q = 2$). If we consider portfolio P_2 in Table 4.1, it has a random selection of $2K$ assets $\{1, 3, 5, 7, 8, 9\}$. Those randomly selected assets undergo subset optimisation which results in the portfolio $\{1, 5, 7\}$ having a portfolio return of 0.004011 and portfolio risk (variance) of 0.021050. P_2 has been placed in the parent set Q_1 because it has one of the four lowest portfolio risk values, while a portfolio such as P_4 has not been chosen for a parent set because it has one of the highest portfolio risk (variance) values.

Portfolio	Random Generation	Portfolio Assets (after subset optimisation)	Portfolio Return	Portfolio Risk (variance)	Parent Set
P_1	{1,2,4,7,9,10}	{2,7,10}	0.003936	0.023011	
P_2	{1,3,5,7,8,9}	{1,5,7}	0.004011	0.021050	Q_1
P_3	{2,3,5,7,9,10}	{3,5,9}	0.003953	0.021174	Q_2
P_4	{3,4,6,7,8,10}	{4,6,8}	0.003953	0.022154	
P_5	{2,4,5,6,8,10}	{5,6,10}	0.004169	0.022908	
P_6	{1,2,4,7,8,9}	{4,5,10}	0.003953	0.019497	Q_1
P_7	{2,3,5,8,9,10}	{3,8,10}	0.003987	0.020258	Q_2

Table 4.1: GA Heuristic Algorithm Selection Example

Once the selection process is completed the parental portfolios are subject to crossover and mutation. In Figure 4.1 we illustrate this process for the parent portfolios of P_6 and P_7 given in Table 4.1. Figure 4.1 shows that once the parental sets have been decided S_{in} and S_{out} are determined. In this case, S_{in}=$\{10\}$ because asset 10 is in both portfolios P_6 and

P_7, while S_{out}={1,2,6,7,9} because these assets are not present in either portfolios P_6 or P_7. Subset optimisation is next. This process results in an offspring portfolio {3,4,10} that has return 0.003953 and risk (variance) 0.018247.

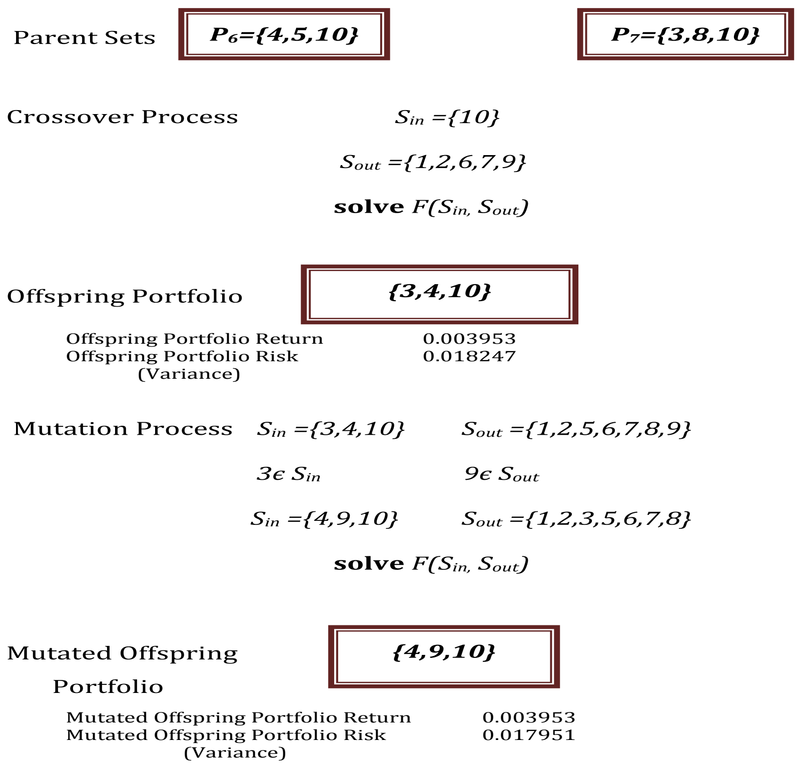

Figure 4.1: GA Heuristic Algorithm Crossover and Mutation Example

In the mutation process S_{in} and S_{out} are once again determined. S_{in} being all assets in the offspring portfolio and S_{out} representing those assets not present in the offspring portfolio. An asset is randomly chosen from the offspring (in this case 3) and an asset is randomly chosen from S_{out} (in this case 9). Those assets are then interchanged (i.e. 3 is placed in S_{out} while 9 is placed in S_{in}), then subset optimisation takes place to obtain the new portfolio risk (variance) and return levels of the mutated child (consisting of $\{4, 9, 10\}$). Note here that although the assets in the portfolios have not changed, subset optimisation also determines optimal proportions to be invested in each asset. The mutated child (being a feasible solution) would then be added to the original population. The seven portfolios with the lowest risk (variance) would be chosen to form the next generation. In this example, when the child portfolio produced by portfolios P_6 and P_7, $O_{P_6P_7}$, is added to the population of all portfolios, P_1 has the highest portfolio risk (variance), and would therefore be removed from the population. The process would then be continued until the termination criteria have been met.

4.3.3 A Tabu Search Heuristic Algorithm

In our TS heuristic algorithm, given the desired return of R, we first generate $|P| = 100$ different portfolios, as for our GA, and then select the portfolio with the lowest risk (variance) as the initial starting solution. Let S_{in} be the set of assets in this initial solution.

In our approach we have a candidate list C of assets that can be considered for inclusion in the current solution, and a tabu list T of assets that cannot be considered. Initialise C with the $N/3$ assets with the highest return (excluding assets in S_{in}). Initialise T with the assets in $[1, ..., N] - S_{in} \cup C$.

In our TS heuristic algorithm, we at each iteration, randomly select an asset i in the current portfolio and replace it by a randomly selected asset j in the candidate list C. Then we solve the subset optimisation problem $\mathrm{F}(S_{in}, S_{out})$ with $S_{out} = [1, ..., N] - S_{in}$. If the resulting portfolio from this optimisation is better (of lower risk) than the current solution then it replaces the current solution and asset i is added to the tabu list T. However, if the resulting portfolio from this optimisation is not better than the current solution then asset j is added to the tabu list T. The candidate list is then updated by adding assets from the tabu list that are no longer tabu, i.e. those assets who have served their tabu tenure are placed in the candidate list. We terminate our TS heuristic algorithm after a fixed number of iterations and use a tabu tenure of 7. Our TS heuristic algorithm is given in psuedocode in Algorithm 2.

S^*	be the initial solution
S^*_c	be the current solution
S^*_p	be the resulting portfolio from the subset optimisation

```
begin
for $R := R_{\min}, ..., R_{\max}$ do                          /examine $R$ values equally spaced in $[R_{\min}, R_{\max}]$/
initialise $P := \{P_1, ..., P_{100}\}$                        /random initialisation, $S_{in} = \max[2K, 20]$ assets/
determine $S_{out} := [1, ..., N] - S_{in}\ \forall p \in P$
solve $\mathrm{F}(S_{in}, S_{out})\ \forall p \in P$           /subset optimisation/
$S^* := \{p \in P |\ \min \sigma^2_p\}$                        /initial solution/
initialise $C$:={the $N/3$ assets with highest
            return excluding assets in $S^*$}
initialise $T := \{[1, ..., N] - S^* \cup C\}$
    for $g := 1, ..., G$ do                                    /G iterations in all/
    $S^*_c := S^*$
    randomly select $i \in S^*_c$ and $j \in C$
    $S_{in} := S^*_c \cup [j] - [i]$                           /neighbourhood solution/
    $S_{out} := [1, ..., N] - S_{in}$
    solve $\mathrm{F}(S_{in}, S_{out})$                        /subset optimisation/
        if $\mathrm{F}(S_{in}, S_{out})$ is feasible then      /evaluate solution/
            if $\sigma^2_{S^*_p} < \sigma^2_{S^*_c}$ then
                $S^* := S^*_p$                                 /move/
                $T := T \cup [i]$                              /update tabu list/
            else
                $S^* := S^*_c$                                 /no move/
                $T := T \cup [j]$                              /update tabu list/
            end if
        end if
        if $\mathrm{F}(S_{in}, S_{out})$ is infeasible then    /evaluate solution/
            $T := T \cup [j]$                                  /update tabu list/
        end if
        check $T$ and update $C$                               /determine assets who have served
                                                                   tabu tenure and place in $C$/

    end for
    end for
    end
```

Algorithm 2: TS heuristic algorithm psuedocode

4.3.4 A Simulated Annealing Heuristic Algorithm

In our SA heuristic algorithm, given the desired return of R, we generate an initial starting solution (a set S_{in} of assets in the portfolio) in the same manner as in our TS heuristic algorithm above.

At each iteration we randomly select an asset i in the current solution S_{in} and swap it with a randomly selected asset j not in the current solution (therefore $j \notin S_{in}$). Then we solve the subset optimisation problem F(S_{in}, S_{out}) with $S_{out} = [1, ..., N] - S_{in}$. If the portfolio resulting from this optimisation is better (of lower risk) than the current solution then it replaces the current solution. If it is worse than the current solution then it is accepted (so replacing the current solution) with probability $e^{-(difference\ in\ solution\ risk\ values)/(current\ temperature)}$. The current temperature is reduced by a constant (cooling) factor at each iteration.

We terminate our SA heuristic algorithm after a fixed number of iterations. In the computational results given in Section 4.4 we use a cooling factor of 0.95 and an initial temperature derived from the objective function value of the initial starting solution. Our SA heuristic algorithm is given in psuedocode in Algorithm 3.

$Temp$	be the current temperature
α	be the cooling factor

```
begin
for $R := R_{\min}, ..., R_{\max}$ do                          /examine $R$ values equally spaced in $[R_{\min}, R_{\max}]$/
initialise $P := \{P_1, ..., P_{100}\}$                         /random initialisation, $S_{in} = \max[2K, 20]$ assets/
determine $S_{out} := [1, ..., N] - S_{in}\ \forall p \in P$
solve $\mathrm{F}(S_{in}, S_{out})\ \forall p \in P$                        /subset optimisation/
$S^* := \{p \in P | \min \sigma_p^2\}$                          /initial solution/
$Temp := \min[\sigma_p^2 | p \in P]/10$                        /initialise SA parameters/
$\alpha := 0.95$
  for $g := 1, ..., G$ do                                       /G iterations in all/
  $S_c^* := S^*$
  randomly select $i \in S_c^*$ and $j \notin S_c^*$
  $S_{in} := S_c^* \cup [j] - [i]$
  $S_{out} := [1, ..., N] - S_{in}$
  solve $\mathrm{F}(S_{in}, S_{out})$                                 /subset optimisation/
    if $\mathrm{F}(S_{in}, S_{out})$ is feasible then                 /evaluate solution/
      if $\sigma^2_{S_p^*} < \sigma^2_{S_c^*}$ then $S^* := S_p^*$                  /move/
      else
        $r$:= a random number from $[0, 1]$
        if $r > exp^{-(\sigma^2_{S_c^*} - \sigma^2_{S_p^*})/Temp}$ then         /criteria for accepting worse portfolio/
          $S^* := S_p^*$
        end if
      end if
    $Temp := \alpha Temp$                                        /update temperature/
  end for
  end for
  end
```

Algorithm 3: SA heuristic algorithm psuedocode

4.4 Computational Results

In this Section we present the data sets (Section 4.4.1), CPLEX results (Section 4.4.2), percentage deviation calculations (Section 4.4.3), the heuristic algorithms' parameter values (Section 4.4.4), and the heuristic algorithms' results for the CCEF (Section 4.4.5).

4.4.1 Data Sets

We tested the performance of our GA, TS and SA heuristic algorithms for finding the cardinality constrained efficient frontier using publicly available test problems relating to seven major market indices, available from the OR-Library (Beasley, [6]).

Five of our market indices were the Hang Seng (Hong Kong), DAX 100 (Germany), FTSE 100 (UK), S&P 100 (USA) and the Nikkei 225 (Japan), as taken from http://people.brunel.ac.uk/∼mastjjb/jeb/orlib/portinfo.html. All of these problems were previously considered by Chang *et al.* [13]. The remaining two market indices were the S&P 500 (USA) and Russell 2000 (USA), as taken from http://people.brunel.ac.uk/∼mastjjb/jeb/orlib/indtrackinfo.html. The size of our seven test problems ranged from $N = 31$ (Hang Seng) to $N = 1318$ (Russell 2000). We used $l_i = 0.01$, $u_i = 1$ $(i = 1, ..., N)$ and $K = 10$.

As we are interested in the cardinality constrained efficient frontier our results below are for tracing out this frontier using 50 equally spaced desired return levels R between the return level associated with the minimum variance unconstrained portfolio R_{min} and the return level associated with the maximum asset return $R_{max} = \max[\mu_i|i = 1, ..., N]$.

Our heuristic algorithms were implemented using AMPL and its associated script language. The solver we used was CPLEX (version 11.0). The system runs under Windows NT and in our computational work we used an Intel Core2 pc with a 2.40 GHz processor and 3.24 GB RAM.

4.4.2 CPLEX Results

Before using the heuristic algorithm approaches presented above to solve the CCEF we investigated using CPLEX to test how effectively it could determine CCEFs. Potentially, for example, should CPLEX be able to optimally solve for the CCEF, i.e. to optimally solve the CCEF QMIP (minimise equation (2.1) subject to equations (2.2)-(2.4), and equations (2.22)-(2.23)), there may be no need for any heuristic algorithm approaches.

We considered two cases for the cardinality constraint: one in which equality is considered (thus $\sum_{i=1}^{N} \delta_i = K$ as in equation (2.23)) and the other where the equality in equation (2.23) is replaced by inequality, hence $\sum_{i=1}^{N} \delta_i \leq K$.

We tested CPLEX (version 11.0) on one of the smaller test problems (DAX 100, N=85 assets) and the results are shown in Table 4.2. As mentioned above in Section 4.4.1 these results are for 50 equally spaced return levels. Therefore, for example, in this Table we have that for the DAX with $K = 5$ and equality in terms of the number of chosen assets to trace out the CCEF at these 50 return levels required 58336 seconds (over 16 hours).

	K=2	K=3	K=4	K=5
Equality case (precisely K assets in the portfolio)	62	527	6984	58336
Inequality case ($\leqslant K$ assets in the portfolio)	19	50	106	138

Table 4.2: Computation time (seconds) for the DAX CCEF using CPLEX

It is clear from Table 4.2 that the inequality case (for the DAX at least) is computationally far easier than the equality case. We also attempted to solve the largest test problem (Russell 2000, N=1318 assets) for the same set of eight cases ($K = 2, 3, 4, 5$ and equality/inequality) as shown in Table 4.2. CPLEX was unable to solve even a single one of these eight cases (not even $K = 2$, inequality) within a time limit of 7200 seconds (2 hours).

Based on Table 4.2 and our work with Russell 2000, N=1318 assets we would conclude that solving the CCEF QMIP (for the equality case) using CPLEX is not a computationally effective approach. As such we are justified in adopting heuristic algorithm approaches to the problem. Note here that these results for CPLEX accord with other results presented in the literature (Shaw *et al.*, [64]; Bertsimas and Shioda, [8]), albeit those results relate to an earlier version of CPLEX.

4.4.3 Percentage Deviation Calculations

In general for measuring the quality of a heuristic algorithm, one would like to measure the deviation of the heuristic algorithm solution from the optimal solution. However for the CCEF, as the results in Table 4.2 illustrate, the optimal frontier is typically unknown. As such in measuring the quality of the results produced by our heuristic algorithms we adopt the same approach as used previously by Chang *et al.* [13]. This involves calculating the percentage deviation of points on the heuristically calculated CCEF from the unconstrained efficient frontier (which can be easily calculated using QP). This method as presented by Chang *et al.* [13] is set out below.

Let (x_i, y_i) be the discrete (x-coordinate: standard deviation, y-coordinate: return) values on our UEF. For a portfolio with (x^*, y^*) let j correspond to $y_j = \min[y_i | y_i \geqslant y^*]$ and k correspond to $y_k = \max[y_i | y_i \leqslant y^*]$ (i.e. y_j and y_k are the closest y-coordinates bracketing y^*). Simple geometry enables us to say that the value x^{**} associated with the x-direction linearly interpolated point on the UEF with $y = y^*$ (i.e. looking horizontally) is $x^{**} = x_k + (x_j - x_k)[(y^* - y_k)/(y_j - y_k)]$. A convenient percentage deviation error measure for this direction is then $|100(x^* - x^{**})/x^{**}|$ (note here that no value is calculated if either j or k do not exist).

To derive an expression for linear interpolation in the y-direction: let j correspond to $x_j = \min[x_i | x_i \geqslant x^*]$ and k correspond to $x_k = \max[x_i | x_i \geqslant x^*]$ (i.e. x_j and x_k are the closest x-coordinates bracketing x^*). Then y^{**} associated with the y-direction linearly interpolated point on the UEF with $x = x^*$ (i.e. looking vertically) is $y^{**} = y_k + (y_j - y_k)[(x^* - x_k)/(x_j - x_k)]$. A convenient percentage deviation error measure for this direction is then $|100(y^* - y^{**})/y^{**}|$ (note here that no value is calculated if either j or k do not exist).

4.4.4 Heuristic Algorithms' Parameter Values

In creating heuristic algorithms for genetic algorithm, tabu search and simulated annealing an important step is the assignment of parameter values (e.g. for population size, tabu tenure, and cooling factor) which pose a trade-off between efficiency and effectiveness. In this Section, we answer the questions of, *Why and How exactly did we decide upon the parameter values?*

We investigated parameter values on the smallest data set (Hang Seng, $N = 31$ assets) and the results (for mean and median percentage errors as well as the computation time in seconds) are given in Table 4.3 (for genetic algorithm), Table 4.4 (for tabu search) and, Table 4.5 (for simulated annealing). As mentioned earlier in Section 4.4.1 these results are for 50 equally spaced desired return levels.

Genetic Algorithm Heuristic Algorithm Parameter Values

Within the framework of the GA heuristic algorithm we had to decide parameters that related to the operators of selection (population size parameter), crossover (parental set size parameter), and mutation (mutation probability parameter).

We began with the selection operator by varying population sizes. Alander [4] suggested that a population size around 50 to 200 is suitable for most problems, thus we tested $|P| = 50, 100$ and 150. The best mean percentage error was given when $|P| = 50$, which is a small population and would not allow sufficient room to explore the search space effectively especially when considering we would be incorporating larger market indices (such as the S&P 500 with 457 assets or the Russell 2000 having 1318 assets). The difference between the mean percentage error of $|P| = 50$ and $|P| = 100$ is only 0.0005% with $|P| = 100$ offering more opportunities to explore the entire search space. The best median percentage error occurred when $|P| = 100$, suggesting that there were many low values and the higher mean percentage error was the result of a few high percentage error values. $|P| = 150$ was the most computationally expensive, offering both the highest mean and median percentage errors.

Using all this information, we decided to use a population size of 100.

With the population size determined (i.e. we are working in a sequential fashion to the GA heuristic algorithm process to decide parameter values) we next had to determine the crossover operator parameter value of parental set size. Hence, we tried $q = 3, 5$ and 7. The smallest q value of 3 gave the best mean percentage error and times of 0.6959% and 47 seconds, respectively. This mean percentage error had a difference of 0.1541% with the second best mean percentage error given by $q = 5$. The best median percentage error occurred when $q = 5$, once again suggesting that there were many low values and the higher mean percentage error was the result of a few high percentage error values. The most computationally time consuming parental set was $q = 7$ which also offered the highest median percentage error of 0.7457% and the second highest mean percentage error of 0.7668%. These results suggested to us that it would be better to use $q = 5$ which had the better median percentage error and gave more chances for crossover than $q = 3$ in the population.

Our final decision for the GA heuristic algorithm was the assignment of the mutation probability for the population. As we stated earlier in Section 2.5.1, mutation is a secondary operator used with low probability. As a result we tried m_p values of 0.01, 0.03 and 0.05. The mutation probability having the best mean percentage error was 0.01, which was only 0.0303% more than the mean percentage error than $m_p = 0.03$. The best median percentage error was 0.5873% which came from both $m_p = 0.03$ and $m_p = 0.05$. But, $m_p = 0.05$ posted the worst values on time and mean percentage error. In light of this, we decided that we would use $m_p = 0.03$ which appeared to best keep the benefits of crossover of the three mutation probabilities.

In Table 4.3 the parameters values relating to population size, parental set sizes and mutation probabilities are all given.

Tabu Search Heuristic Algorithm Parameter Values

With the population size already having been determined from the GA heuristic algorithm we turn our attention to the TS heuristic algorithm and focus on the parameter relating to tabu

Percentage error	Population Sizes			Parental Sets			Mutation Rates		
& time	$\lvert P\rvert = 50$	$\lvert P\rvert = 100$	$\lvert P\rvert = 150$	$q = 3$	$q = 5$	$q = 7$	$m_p = 0.01$	$m_p = 0.03$	$m_p = 0.05$
Mean	0.8496	0.8501	0.9100	0.6959	0.8501	0.7668	0.8197	0.8501	0.9089
Median	0.5989	0.5873	0.6105	0.6104	0.5873	0.7457	0.6103	0.5873	0.5873
Time (seconds)	64	76	112	47	76	124	67	76	101

Table 4.3: GA Heuristic Algorithm Parameter Test Values

tenure. For this parameter we tested the three values of 5, 7 and 10.

The tabu tenure of 5 had the best mean percentage error value, while the best median percentage error occurred with a tabu tenure of 7. The tabu tenure of 10 produced the highest computational time as well as mean and median percentage error. We decided to use a tabu tenure of 7 because of two main reasons. The first is because despite the fact that the tabu tenure of 5 has attractive results with the Hang Seng data, a small tabu tenure can cause cycling within the search space. Then, secondly our chosen tabu tenure would be in keeping with Glover and Languana [29] who suggested a minimum tabu tenure of 7. The tabu tenure of 10, although not very large we felt may cause appealing moves to be forbidden and lead to the exploration of lower quality solutions. In Table 4.4 the tabu tenure test parameter values are all given.

Percentage error	Tabu Tenure		
& time	5	7	10
Mean	0.7645	0.8234	1.1529
Median	0.4173	0.3949	0.5169
Time (seconds)	69	76	84

Table 4.4: TS Heuristic Algorithm Parameter Test Values

Simulated Annealing Heuristic Algorithm Parameter Values

For the SA heuristic algorithm the parameter we needed to decide is the constant factor α which relates to the cooling schedule. The typical range for this value is between 0.90 and 0.99. For this parameter we tested three α values of 0.90, 0.95 and 0.975. After examining the results we decided to use $\alpha = 0.95$ because it gave the best mean and median percentage error in a reasonable time.

In Table 4.4 the cooling schedule test parameter values are all given.

Percentage error	Constant Factor		
& time	$\alpha = 0.90$	$\alpha = 0.95$	$\alpha = 0.975$
Mean	1.5806	1.0589	1.0913
Median	1.5791	0.5355	0.9094
Time (seconds)	67	76	86

Table 4.5: SA Heuristic Algorithm Parameter Test Values

4.4.5 Heuristic Algorithms' Results for the CCEF

We have divided this Section into two parts: trade-off curves and tabular heuristic algorithms' results for the CCEF. In the first Section we consider only the DAX 100 Market Index trade-off curves. Then in the second Section we consider all of the Market Indices mentioned in Section 4.4.1. Within each Section we show how our results compare to those of Chang *et al*. [13]. The computational results reported (as mentioned above) examine 50 equally spaced return levels.

Trade-off Curves

In creating portfolios, the decision maker is faced with a different CCEF trade-off curve for each value of K and thus must consider the tradeoff between risk, return and the number of assets in the portfolio when deciding which portfolio to adopt. In this Section, we graphically depict the UEF, the tradeoff curves (for K = 2, 3, 4, 5) that we were able to obtain for our GA heuristic algorithm and those tradeoff curves Chang *et al* [13] obtained for the same K values using their pooled heuristic algorithms (GA, TS and SA) results from the DAX 100 Market Index. In both Chang *et al*. [13] and our work we use $l_i = 0.01$ and $u_i = 1$.

Figure 4.2 illustrates the DAX trade-off curves for $K = 2$ and $K = 4$, while Figure 4.3 illustrates the DAX trade–off curves for $K = 3$ and $K = 5$. If we consider $K = 5$ in Figure 4.3, we can see portfolios at return levels which appeared to be discontinuities in the Chang *et al.* [13] results. The same is true also for all K values. Pictorially it is seen that in all cases of K, our results gives the investor more risk-return choices and better quality solutions to use for making a decision.

Tabular Heuristic Algorithms' Results for the CCEF

With regard to the number of iterations, which is the termination criteria for both our TS and SA heuristic algorithms, we used 100 iterations at each return level for the TS heuristic algorithm and 50 iterations at each return level for the SA heuristic algorithm. Our GA heuristic algorithm was repeated for 4 generations (as was stated earlier).

In Table 4.6 we show for each of our data sets and each of our heuristic algorithms: the mean, median, maximum and minimum percentage errors as well as the computation time in seconds. Also presented in Table 4.6 are the mean and median percentage errors and computation time for the five smaller test problems as given in Chang *et al.* [13] using their GA, SA and TS heuristic algorithms.

In Figure 4.4 we display the undominated points obtained for each of our heuristic algorithms for the Russell 2000 Index Market. The figure shows that the GA and TS heuristic algorithms were able to produce many portfolios for estimating the UEF while, the SA heuristic algorithm produce few portfolios and did not have the same quality of solution as the GA and TS heuristic algorithms.

Considering our GA, TS and SA heuristic algorithms as presented in this thesis, labeled (Woodside-Oriakhi in Table 4.6), it seems reasonable to conclude from the values presented at the foot of Table 4.6 (in the row Average, all problems) that SA heuristic algorithm is not competitive with GA and TS heuristic algorithms. Our TS heuristic algorithm (on average) gives better quality results than our GA heuristic algorithm but at the expense of more computation time. For example, over all problems, the mean error for our TS heuristic algorithm is only 0.8512% in an average of 351 seconds, compared to a mean error of 1.3163% in an average of 125 seconds for our GA heuristic algorithm. However for all heuristic algorithms the computation time is not excessive, the largest computation time (given by the SA heuristic

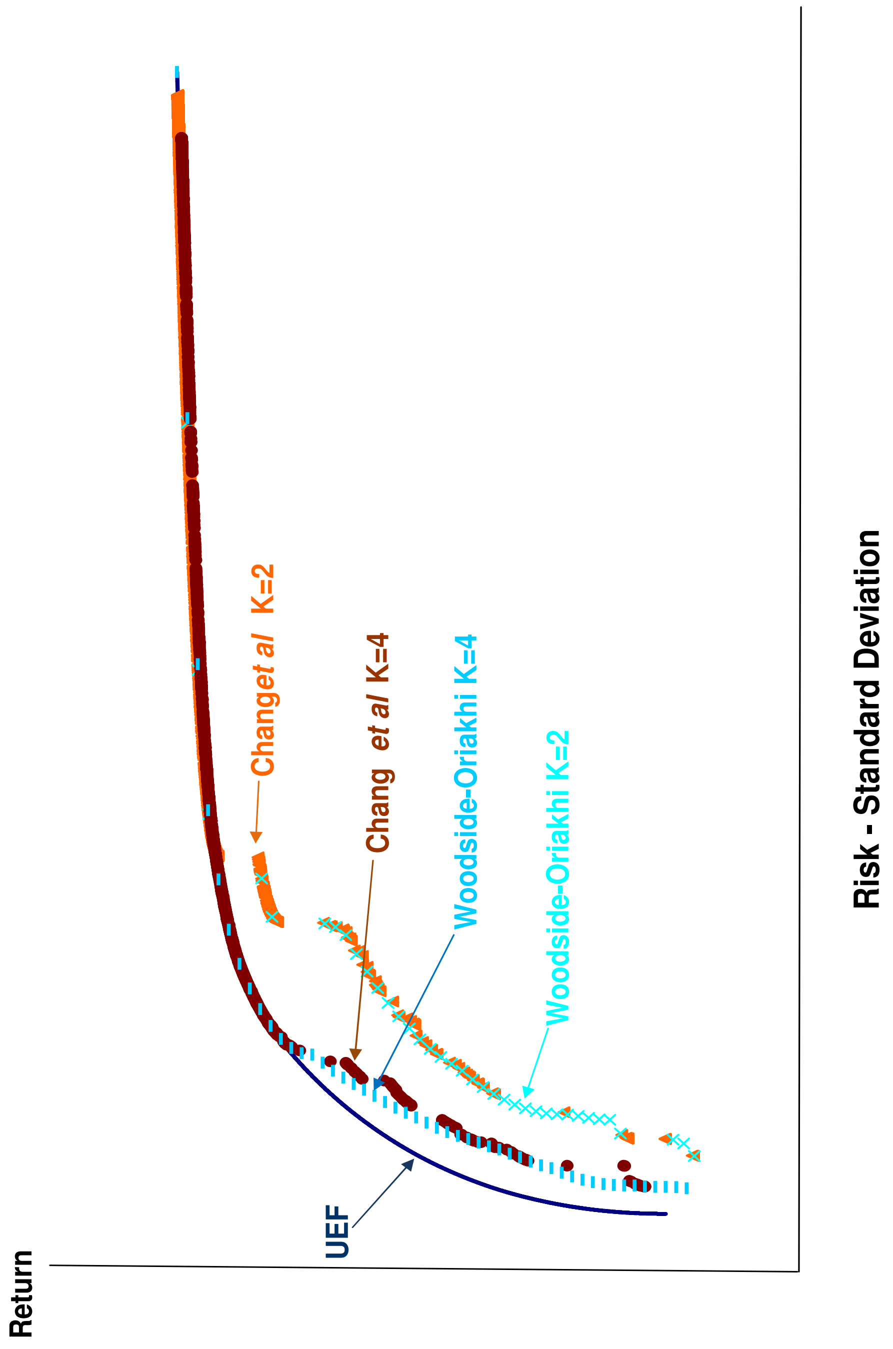

Figure 4.2: DAX 100 Trade-off Curves for $K = 2$ and $K = 4$

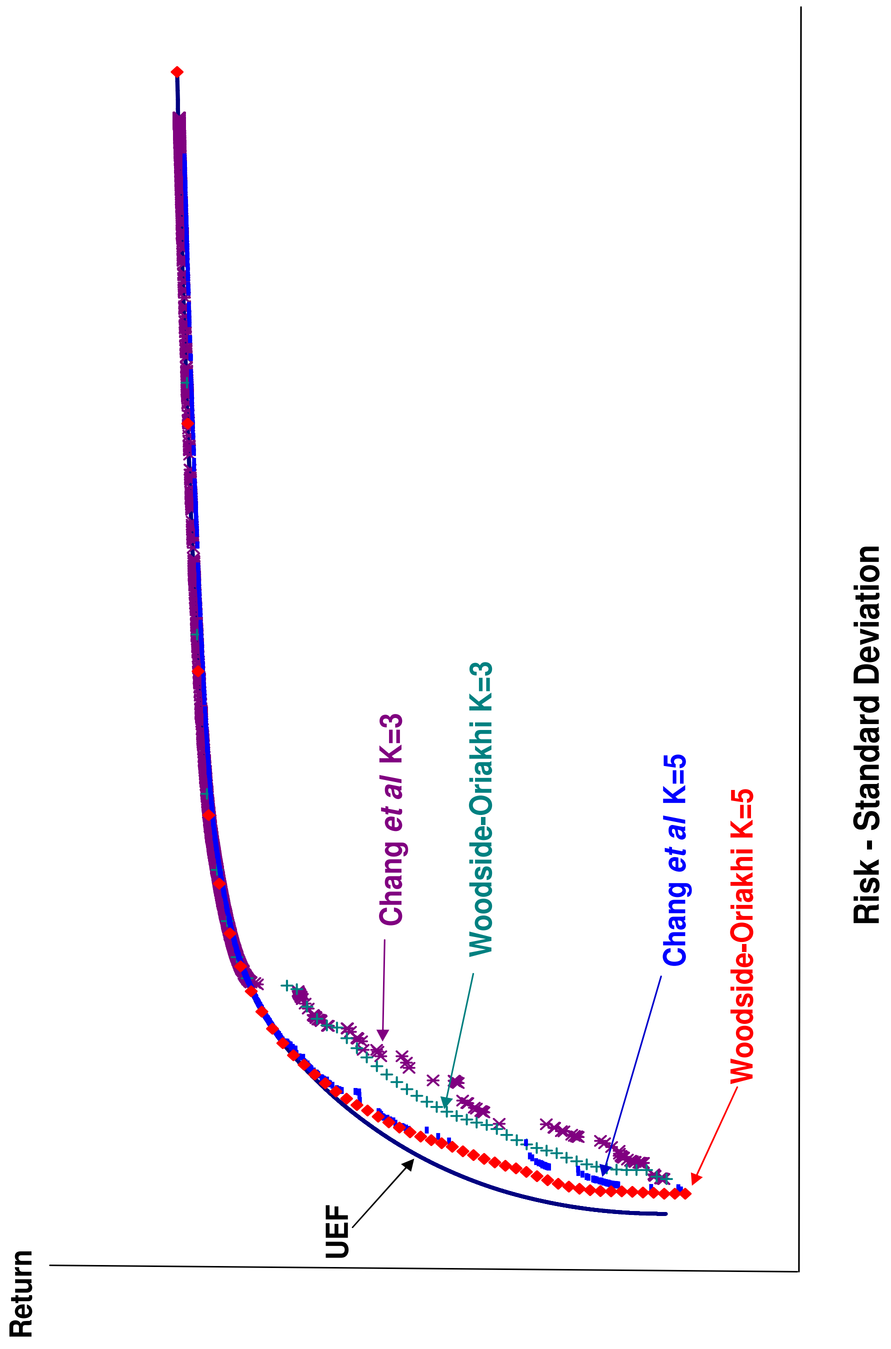

Figure 4.3: DAX 100 Trade-off Curves for $K = 3$ and $K = 5$

Index	N	Percentage error & time	Genetic Algorithm		Tabu Search		Simulated Annealing	
			Chang *et al.*	Woodside-Oriakhi	Chang *et al.*	Woodside-Oriakhi	Chang *et al.*	Woodside-Oriakhi
Hang Seng	31	Mean	0.9457	0.8501	0.9908	0.8234	0.9892	1.0589
		Median	1.1819	0.5873	1.1992	0.3949	1.2082	0.5355
		Minimum		0.0036		0.0068		0.0349
		Maximum		2.9034		4.6096		4.6397
		Time (seconds)	172	76	74	85	79	99
DAX 100	85	Mean	1.9515	0.7740	3.0635	0.7190	2.4299	1.0267
		Median	2.1262	0.2400	2.5383	0.4298	2.4675	0.8682
		Minimum		0.0000		0.0149		0.0278
		Maximum		4.6811		2.7770		4.4123
		Time (seconds)	544	74	199	113	210	293
FTSE 100	89	Mean	0.8784	0.1620	1.3908	0.3930	1.1341	0.8952
		Median	0.5938	0.0820	0.6361	0.2061	0.7137	0.3944
		Minimum		0.0000		0.0019		0.0230
		Maximum		0.7210		3.4570		10.2029
		Time (seconds)	573	95	246	232	215	286
S&P 100	98	Mean	1.7157	0.2922	3.1678	1.0358	2.6970	3.0952
		Median	1.1447	0.1809	1.1487	1.0248	1.1288	2.1064
		Minimum		0.0007		0.0407		0.8658
		Maximum		1.6295		3.0061		8.6652
		Time (seconds)	638	100	225	222	242	371
Nikkei 225	225	Mean	0.6431	0.3353	0.8981	0.7838	0.6370	1.1193
		Median	0.6062	0.3040	0.5914	0.6526	0.6292	0.6877
		Minimum		0.0180		0.0085		0.0113
		Maximum		1.0557		2.6082		3.9678
		Time (seconds)	1964	104	545	414	553	604
Average, Chang problems		Mean	1.2269	0.4827	1.9022	0.7510	1.5774	1.4391
		Median	1.1306	0.2788	1.2227	0.5416	1.2295	0.9184
		Minimum		0.0045		0.0146		0.1926
		Maximum		2.1981		3.2916		6.3776
		Time (seconds)	778	90	258	213	260	331
S&P 500	457	Mean		2.0205		1.4689		5.2502
		Median		0.1899		1.1047		4.5142
		Minimum		0.0114		0.0335		0.1552
		Maximum		21.1701		5.1203		13.9470
		Time (seconds)		187		660		719
Russell 2000	1318	Mean		4.7797		0.7345		4.1102
		Median		0.0940		0.2700		3.8136
		Minimum		0.0001		0.0097		0.0001
		Maximum		58.7478		3.8205		8.5477
		Time (seconds)		239		729		868
Average, all problems		Mean		1.3163		0.8512		2.3651
		Median		0.2397		0.5833		1.8457
		Minimum		0.0048		0.0166		0.1597
		Maximum		12.9869		3.6284		7.7689
		Time (seconds)		125		351		463

Table 4.6: Heuristic Algorithms' Results for the CCEF

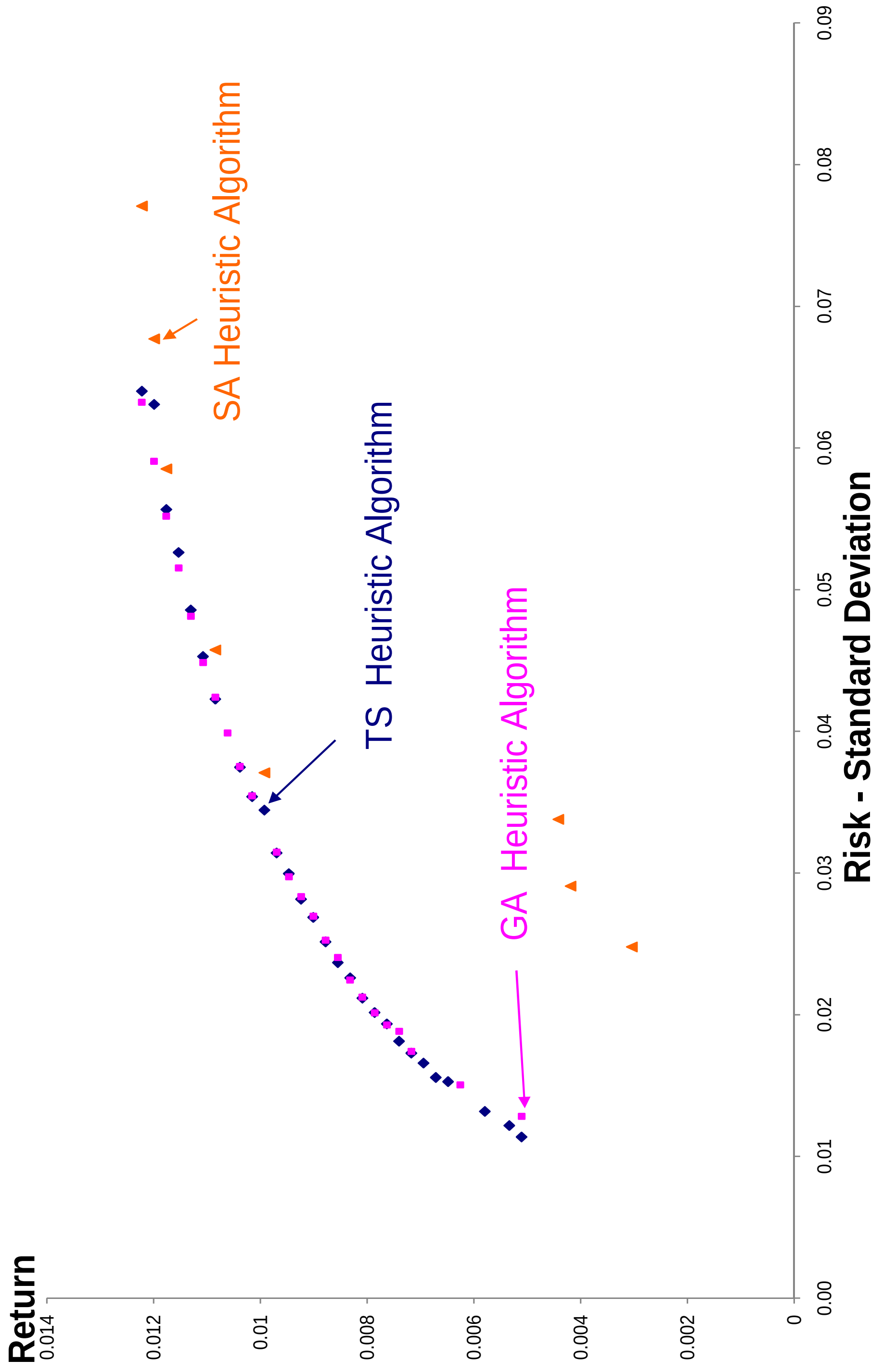

Figure 4.4: Russell 2000 GA, TS and SA Heuristic Algorithms' Results

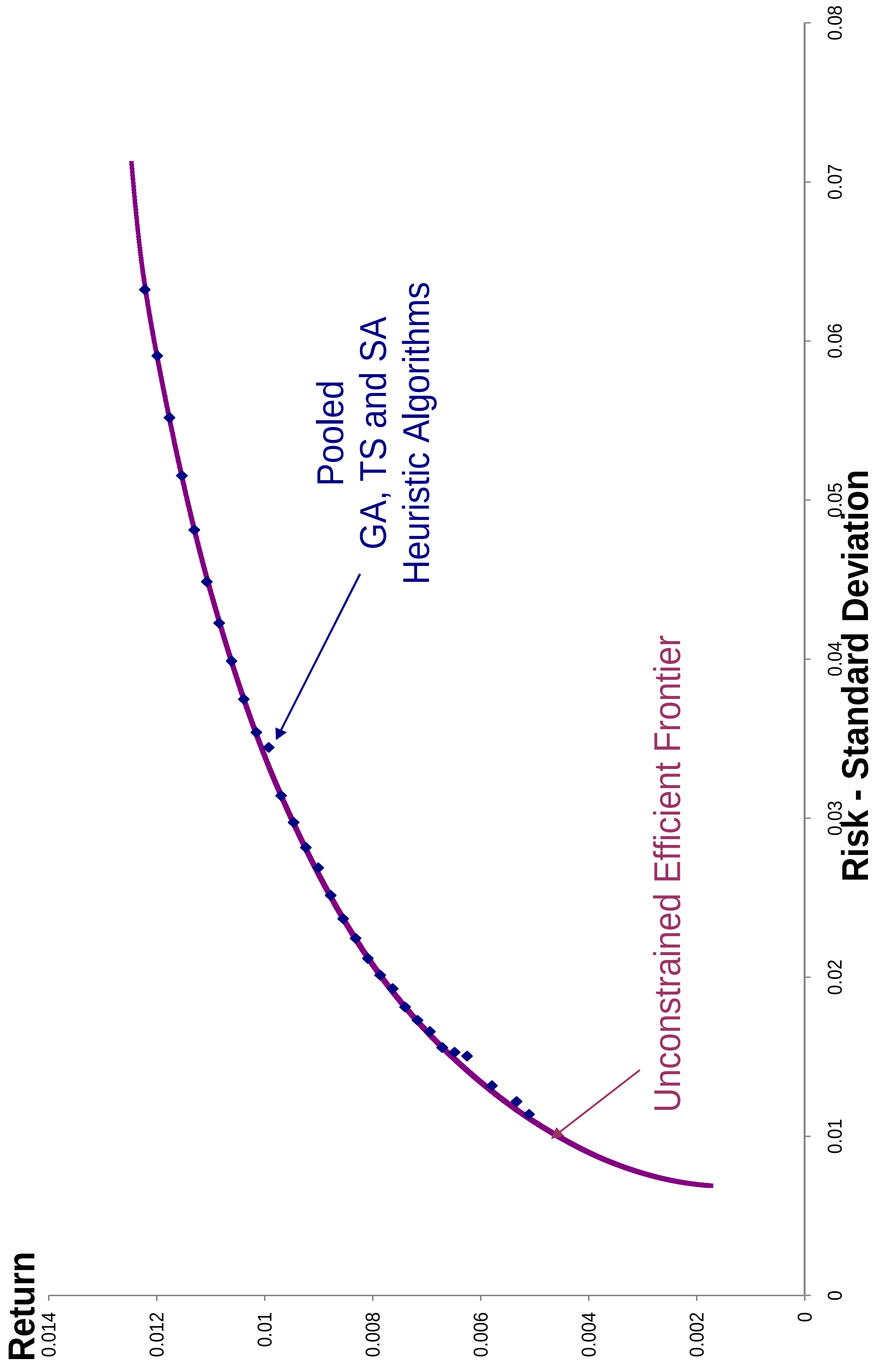

Figure 4.5: Russell 2000 pooled GA, TS and SA Heuristic Algorithms' Results

algorithm for the Russell 2000) seen in Table 4.6 being 868 seconds, approximately 15 minutes.

Comparing, for the five smaller test problems, our results with the results for the GA, SA and TS heuristic algorithms of Chang *et al.* [13] it seems reasonable to conclude from the averages over these five test problems that our GA and TS heuristic algorithms give solutions of significantly better quality than the GA and TS heuristic algorithms of Chang *et al.*. For example, over all five smaller test problems: our GA heuristic algorithm has a mean error of 0.4827%, the GA heuristic algorithm of Chang *et al.* [13] a mean error of 1.2269%; our TS heuristic algorithm has a mean error of 0.7510%, the TS heuristic algorithm of Chang *et al.* [13] a mean error of 1.9022%; our SA heuristic algorithm has a mean error of 1.4391%, the SA heuristic algorithm of Chang *et al.* [13] a mean error of 1.5774%.

Moreover for these five test problems it seems that our GA heuristic algorithm outperforms our TS heuristic algorithm, both with respect to solution quality and computation time. Our GA heuristic algorithm has a mean error of 0.4827% in an average of 90 seconds, our TS heuristic algorithm has a mean error of 0.7510% in an average of 213 seconds.

With regard to computation time the times given for the work of Chang *et al.* [13] relate to different hardware than we have used. Utilising Dongarra [21] it is possible to make an ***approximate*** estimate of the relative speed of the hardware involved. On this basis the computation times for the work of Chang *et al.* [13] as shown in Table 4.6 should be divided by a factor of 70 to be comparable with the hardware we have used. As such we can conclude that for these smaller test problems our GA heuristic algorithm takes longer, but gives better quality results in a reasonable time (less than two minutes), than any of the heuristic algorithms of Chang *et al.* [13].

As we have a number of results from different heuristic algorithm approaches we can pool results, i.e. combine together the efficient portfolios from each of the heuristic algorithms and eliminate any portfolios that are dominated. In Table 4.7 we show the pooled results as given in Chang *et al.* [13] and present the pooled results for our three heuristic algorithms.

Comparing the averages at the foot of Table 4.7 (in the row Average, all problems) it seems clear that there is little advantage to including results from our SA heuristic algorithm in

Index	N	Percentage error & time	Pooled heuristics				
			Chang *et al.*	Woodside-Oriakhi			
			GA,TS,SA	GA,TS,SA	GA,TS	GA,SA	TS,SA
Hang Seng	31	Mean	0.9332	0.4265	0.4098	0.6404	0.6036
		Median	1.1899	0.1839	0.1948	0.3669	0.3745
		Time (seconds)	325	260	161	175	184
DAX 100	85	Mean	2.1927	0.6539	0.4696	0.7055	0.7070
		Median	2.4626	0.2194	0.2073	0.2275	0.4247
		Time (seconds)	953	480	187	367	406
FTSE 100	89	Mean	0.7790	0.4418	0.2690	0.1598	0.5284
		Median	0.5960	0.1074	0.0851	0.0935	0.2061
		Time (seconds)	1034	613	327	381	518
S&P 100	98	Mean	1.3106	0.6748	0.5109	0.6172	1.0944
		Median	1.0686	0.2395	0.2756	0.2712	1.0495
		Time (seconds)	1105	693	322	471	593
Nikkei 225	225	Mean	0.5690	0.7307	0.7214	0.3870	0.9119
		Median	0.5844	0.3223	0.3223	0.2785	0.5481
		Time (seconds)	3062	1122	518	708	1018
Average, Chang problems		Mean	1.1569	0.5855	0.4760	0.5020	0.7691
		Median	1.1803	0.2145	0.2170	0.2475	0.5206
		Time (seconds)	1296	634	303	420	544
S&P 500	457	Mean		0.8385	0.7549	1.1319	1.5500
		Median		0.2861	0.2685	0.2181	1.1581
		Time (seconds)		1566	847	906	1379
Russell 2000	1318	Mean		0.7192	0.7192	4.7797	0.8355
		Median		0.1040	0.1040	0.0940	0.2890
		Time (seconds)		1836	968	1107	1597
Average, all problems		Mean		0.6408	0.5506	1.2031	0.8901
		Median		0.2089	0.2082	0.2214	0.5786
		Time (seconds)		939	476	588	814

Table 4.7: Pooled Heuristic Algorithms' Results for the CCEF

pooling, rather the best pooled results come from pooling our GA and TS heuristic algorithms. Comparing pooled results for GA and TS heuristic algorithms with the individual results for GA and TS as presented at the foot of Table 4.6 it seems clear that the quality of results are improved considerably by pooling. For example, the mean error from pooling our GA and TS heuristic algorithms is 0.5506%, our GA and TS heuristic algorithms individually have mean errors of 1.3163% and 0.8512% respectively. In Figure 4.5 we display the UEF and the undominated pooled heuristic algorithms results for the Russell 2000 Index Market. From Table 4.7, we notice that the pooled results of the GA, TS and SA heuristic algorithms is the same as the pooled results of GA and TS heuristic algorithms for the Russell 2000 indicating that all the SA heuristic algorithm results were dominated by those of either the GA or the TS heuristic algorithms. This example shows there was no advantage in including the SA heuristic algorithm.

For the five smaller test problems the pooled results from our GA and TS heuristic algorithms are of better quality than the pooled results for all three of the Chang *et al.* [13] heuristic algorithms, the mean error from pooling our GA and TS heuristic algorithms for these test problems is 0.4760%, the mean error from the pooled Chang *et al.* [13] heuristic algorithms is 1.1569%.

4.5 Summary

In this Chapter we presented heuristic algorithms for finding the cardinality constrained efficient frontier. We began in Section 4.2 by presenting an optimisation model that allowed us to specify subsets of assets for which we knew whether they were to be included in or excluded from the chosen portfolio. We called this model the subset optimisation problem which underlies each of our heuristics.

Then, in Section 4.3, we outlined our heuristic algorithm implementation of genetic algorithm, tabu search and simulated annealing for finding the CCEF. This was followed by Section 4.4 where we gave computational results for our heuristic algorithms on test problems considered previously in the literature, as well as on two larger test problems involving 457 and 1318 assets.

Within the computational results we justified our use of heuristic algorithms for finding the CCEF and explained how and why we used our chosen parameter values. We then went on to highlight how our results for the DAX 100 Market Index trade-off curves and those for finding the CCEF compared to those of Chang *et al.* [13]. We showed that in both cases our results gives more risk-return choices and a better quality solution than previous heuristic algorithms presented in the literature, albeit at the expense of more computation time. However, in all cases, our computation times were reasonable and were never more than fifteen minutes on a modern computer, even for the largest problem.

Chapter 5

Transaction Cost: Heuristic Algorithms

5.1 Introduction

In Chapter 3 we presented optimal solutions for the transaction cost optimisation model then, in Chapter 4 we applied heuristic algorithms to the cardinality constrained efficient frontier. This Chapter builds upon that work by applying the heuristic algorithms of Chapter 4 to the transaction cost models of Chapter 3 to rebalance an existing portfolio. In other words, in this Chapter we use heuristic algorithms to create the portfolio (and efficient) frontier for the non-cardinality and cardinality constrained transaction cost optimisation models.

The remainder of this Chapter is organized as follows. In Section 5.2 we give the subset optimisation problem. This is followed by the heuristic algorithms and data sets in Sections 5.3 and 5.4 respectively. The non-cardinality constrained transaction cost heuristic algorithms' results are provided in Section 5.5 while, the cardinality constrained transaction cost heuristic algorithms' results are made available in Section 5.6. The Chapter is concluded with a summary in Section 5.7.

5.2 Subset Optimisation

As was stated in Chapter 4, in this thesis the heuristic algorithms make use of subset optimisation, that allows us to specify subsets of assets for which we know their status (either in or out of the chosen portfolio). Each of the transaction cost models (non-cardinality and cardinality constrained) has their own subset optimisation model. Below we state the subset optimisation for the cardinality constrained transaction cost problem and in the discussion that follows state the difference which occurs with the two models.

The notation is the same as was adopted in previous chapters, so they are not repeated here.

$$\text{Minimise} \sum_{i=1}^{N}\sum_{j=1}^{N}\sigma_{ij}w_iw_j \tag{5.1}$$

subject to

$$R_L \leqslant \frac{\sum_{i=1}^{N}\mu_iP_ix_i}{\sum_{k=1}^{N}P_kx_k} \leqslant R_U, \ \text{(when linearised)}, \tag{5.2}$$

$$L_i^s\alpha_i^s \leqslant y_i^s \leqslant U_i^s\alpha_i^s, \qquad i = 1,\ldots,N, \tag{5.3}$$

$$L_i^b\alpha_i^b \leqslant y_i^b \leqslant U_i^b\alpha_i^b, \qquad i = 1,\ldots,N, \tag{5.4}$$

$$\alpha_i^s + \alpha_i^b \leqslant 1, \qquad i = 1,\ldots,N, \tag{5.5}$$

$$x_i = X_i + y_i^b - y_i^s, \qquad i = 1,\ldots,N, \tag{5.6}$$

$$G_i = c_i^sy_i^s + c_i^by_i^b + f_i^s\alpha_i^s + f_i^b\alpha_i^b, \qquad i = 1,\ldots,N, \tag{5.7}$$

$$\sum_{i=1}^{N}G_i \leqslant D, \tag{5.8}$$

$$\sum_{i=1}^{N}P_ix_i = \sum_{i=1}^{N}P_iX_i + V^{new} - \sum_{i=1}^{N}G_i, \tag{5.9}$$

$$w_i = \frac{P_ix_i}{\sum_{k=1}^{N}P_kX_k + V^{new}}, \qquad i = 1,\ldots,N, \tag{5.10}$$

$$l_i \leqslant \frac{P_ix_i}{\sum_{k=1}^{N}P_kx_k}, \leqslant u_i, \ \text{(when linearised)}, \qquad i = 1,\ldots,N, \tag{5.11}$$

$$\sum_{i=1}^{N} \delta_i = K, \tag{5.12}$$

$$\sum_{i=1}^{N} \alpha_i^s \leqslant K^S, \tag{5.13}$$

$$\sum_{i=1}^{N} \alpha_i^b \leqslant K^B, \tag{5.14}$$

$$\sum_{i=1}^{N} (\alpha_i^s + \alpha_i^b) \leqslant K^T, \tag{5.15}$$

$$\sum_{i \in S_{in}} \delta_i = \min[|S_{in}|, K], \tag{5.16}$$

$$\delta_i = 0, \qquad \forall i \in S_{out}, \tag{5.17}$$

$$\delta_i \;=\; 0 \,\text{or}\, 1, \qquad i = 1, \ldots, N, \tag{5.18}$$

$$w_i \leqslant \delta_i, \qquad i = 1, \ldots, N, \tag{5.19}$$

$$x_i \geqslant L_i^b \delta_i \text{ if} X_i = 0, \qquad i = 1, \ldots, N, \tag{5.20}$$

$$y_i^b \geqslant L_i^b \delta_i \text{ if} X_i = 0, \qquad i = 1, \ldots, N, \tag{5.21}$$

$$\alpha_i^b \geqslant \delta_i \text{ if} X_i = 0, \qquad i = 1, \ldots, N, \tag{5.22}$$

$$y_i^s \geqslant X_i(1 - \delta_i) \text{ if} X_i > 0, \qquad i = 1, \ldots, N, \tag{5.23}$$

$$\alpha_i^s \geqslant 1 - \delta_i \text{ if} X_i > 0, \qquad i = 1, \ldots, N, \tag{5.24}$$

$$w_i, x_i, y_i^s, y_i^b, G_i \geq 0, \qquad i = 1, \ldots, N, \tag{5.25}$$

$$\alpha_i^s,\, \alpha_i^b \in [0, 1], \qquad i = 1, \ldots, N. \tag{5.26}$$

Equations (5.1), (5.3)-(5.12) and (5.19)-(5.26) are as in the cardinality constrained transaction cost model (minimise equation (3.35) subject to equations (3.36)-(3.46) and equations (3.50)-(3.55)). Equation (5.2) constrains the chosen portfolio's expected return to be within the desired return range, $[R_L, R_U]$. Equation (5.16) forces all assets in S_{in} into the portfolio if $|S_{in}| \leqslant K$ and chooses K assets from S_{in} if $|S_{in}| > K$. Equation (5.17) ensures that assets in S_{out} are not placed in the chosen portfolio while, equation (5.18) declares that the binary decision variables are zero or one.

For the non-cardinality constrained transaction cost subset optimisation problem we remove the cardinality constraint (equation (5.12)) and replace equation (5.16) with

$$\sum_{i \in S_{in}} \delta_i = |S_{in}| \tag{5.27}$$

This equation forces all assets in S_{in} to be in our portfolio. Equations (5.1)-(5.11) and (5.17)-(5.24) remain the same for the non-cardinality constrained transaction cost subset optimisation problem.

The non-cardinality and cardinality constrained transaction cost subset optimisation problem (equations (5.1)-(5.11) and (5.17)-(5.27) and equations (5.1)-(5.24) respectively) like the cardinality constrained model (equations (2.1)-(2.4) and equations (2.21)-(2.23)) and the subset optimisation problem for finding the CCEF (equations (4.1)-(4.9)) are all QMIP. They can be solved relatively quickly to proven optimality, provided that the number of assets for which we have to make a decision as to whether they are to be included in or excluded from the chosen portfolio is small (i.e. $N - |S_{in} \cup S_{out}|$ is small).

5.3 Heuristic Algorithms

The three heuristic algorithms based upon genetic algorithm, tabu search and simulated annealing are the same as was represented in Chapter 4 (sections 4.3.1, 4.3.3 and 4.3.4 that we have developed for finding the CCEF). **This is owing to the fact that our structure in Chapter 4 is so general it can be applied to the transaction cost model. The only change required is to the subset optimisation problem.**

With regard to the number of iterations, we use the same criteria as before (in Chapter 4 Section 4.4.5). That is we use 100 iterations at each return level for the TS heuristic algorithm, 50 iterations at each return level for the SA heuristic algorithm and our GA heuristic algorithm was repeated for 4 generations.

5.4 Data Sets

We tested the performance of our GA, TS and SA heuristic algorithms for finding the non-cardinality and cardinality constrained transaction cost trading portfolio frontier using publicly available test problems relating to six major market indices, available from OR-Library (Beasley, [6]).

The market indices remain the same from Chapters 3 and 4. Subsequently, five of our market indices were the Hang Seng (Hong Kong), DAX 100 (Germany), FTSE 100 (UK), S&P 100 (USA) and the Nikkei 225 (Japan), as taken from
http://people.brunel.ac.uk/∼mastjjb/jeb/orlib/portinfo.html. The remaining market index was the S&P 500 (USA), as taken from
http://people.brunel.ac.uk/∼mastjjb/jeb/orlib/indtrackinfo.html. The size of our six test problems ranged from $N = 31$ (Hang Seng) to $N = 457$ (S&P 500). We used $V^{new} = 0$, $l_i = 0$, $u_i = 1$ $(i = 1, ..., N)$ and $K = 10$. The original portfolios (X_i) are the same as those used in Chapter 3 for each market index. The values for X_i, P_i, L_i^s, L_i^b, U_i^s, U_i^b, f_i^s, f_i^b, c_i^s and c_i^b for all $i = 1, \ldots, N$ are found on the CD accompanying this thesis.

As we are interested in the (non-cardinality and cardinality constrained) transaction cost portfolio (efficient) frontier our results below are for tracing out this frontier using 50 equally spaced desired return levels R between the return level associated with the minimum variance unconstrained portfolio R_{min} and the return level associated with the maximum asset return $R_{max} = \max[\mu_i | i = 1, ..., N]$.

As was the case earlier (in Chapters 3 and 4), our heuristic algorithms were implemented using AMPL and its associated script language. The solver we used was CPLEX (version 11.0). The system runs under Windows NT and in our computational work we used an Intel Core2 pc with a 2.40 GHz processor and 3.24 GB RAM.

5.5 Non-Cardinality Constrained Transaction Cost Heuristic Algorithms' Results

In this Section, we provide the GA (Section 5.5.1), TS (Section 5.5.2), SA (Section 5.5.3) and pooled (Section 5.5.4) heuristic algorithms' results for the non-cardinality constrained transaction cost trading portfolio frontier.

Within each of the subsections, we make available the total computational time of the heuristic algorithm(s), the percentage error of the undominated points produced by the heuristic algorithm(s) from the unconstrained efficient frontier of the market index and the difference in percentage error of the heuristic algorithm(s) efficient frontier and that produced by the non-cardinality constrained efficient frontier (given in Chapter 3, Section 3.7).

Also included in each subsection are graphical illustrations of some of the results. All the graphics include the trading portfolio frontier (dark blue) of the particular market index and the original portfolio (red). Some diagrams contain the non-cardinality constrained trading portfolio frontier of the original portfolio (blue stars) along with the trading portfolio frontier produced by the heuristic algorithm (purple crosses) while the other illustrations will contain the non-cardinality constrained efficient frontier of the original portfolio (purple squares) along with the efficient frontier produced by the heuristic algorithm (blue crosses). Here we note that some of the crosses, squares and stars have been made larger to aid the reader in being able to see them.

5.5.1 Genetic Algorithm

Table 5.1 shows the computational times for the GA heuristic algorithm non-cardinality constrained transaction cost model. In the Table, we state the market indices, the number of assets, the total computational times in seconds taken to create the GA heuristic algorithm non-cardinality constrained transaction cost trading portfolio frontier and the average time (in seconds) per return level for each of the GA heuristic algorithm points. Also, included in Table 5.1 are the total computational time (in seconds) and average time (in seconds) per return level taken to compute the optimal solutions (as found in Chapter 3 Table 3.5).

Considering the Hang Seng market index (in Table 5.1) we see that the $N = 31$ assets had a total computational time of 70 seconds and the average time per return level was 0.070 seconds for the optimal solution while for the GA heuristic algorithm the total computational time was 146 seconds and the average time per return level was 3 seconds. Considering the entire table, we note that each of the optimal solution times were quicker than those of the GA heuristic algorithm. Hence, when comparing the optimal solution to that of the GA heuristic algorithm the optimal solution provided the computational efficient way to solve this problem.

Market Index	Assets N	Optimal Solution		GA Heuristic Algorithm	
		Total Computational Time (seconds	Average time (seconds) per return level	Total Computational Time (seconds)	Average time (seconds) per return level
Hang Seng	31	70	0.070	146	3
DAX 100	85	271	0.271	374	7
FTSE 100	89	1273	1.273	551	11
S&P 100	98	1309	1.309	741	15
Nikkei 225	225	1490	1.490	1217	24
S&P 500	457	6016	6.016	5217	104
Average			2		27

Table 5.1: GA Heuristic Algorithm Computational Times for the Non-Cardinality Constrained Transaction Cost Model

Table 5.2 gives the mean, median, minimum and maximum percentage error results for the GA heuristic algorithm transaction cost non-cardinality constrained efficient frontier from the unconstrained efficient frontier for each of the six market indices. Taking a look at the S&P 100 we see that the mean percentage error is 16.2903%, the median percentage error is 15.4661% while the minimum and maximum percentage errors are 9.1917% and 31.7860% respectively.

The values in Table 5.2 are all over estimates because there exists the non-cardinality constrained efficient frontier between the UEF and the values obtained by the GA heuristic algorithm. Therefore, to derive an estimate of the percentage error of the heuristic algorithm we subtract the particular percentage error of the non-cardinality constrained efficient frontier from the equivalent percentage error of the efficient frontier of the heuristic algorithm. Hence,

Percentage Error	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	S&P 500 $N = 457$	Average all markets
Mean	6.3881	23.7485	16.8221	16.2903	14.9424	48.1775	21.0615
Median	4.4937	24.8043	17.7894	15.4661	16.3514	41.5044	20.0682
Minimum	1.7006	19.8863	9.4703	9.1917	10.9566	29.9343	
Maximum	12.7293	25.4993	20.4473	31.7860	17.5191	82.0831	

Table 5.2: GA Percentage Error Results for the Transaction Cost Non-Cardinality Constrained Efficient Frontier

Percentage Error Difference	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	S&P 500 $N = 457$	Average all markets
Mean	2.4539	12.2520	15.3560	12.2805	6.1914	38.6152	14.5249
Median	2.6621	16.0711	16.4266	13.2949	8.7861	33.4005	15.1069

Table 5.3: GA Percentage Error Difference for the Transaction Cost Non-Cardinality Constrained Frontiers

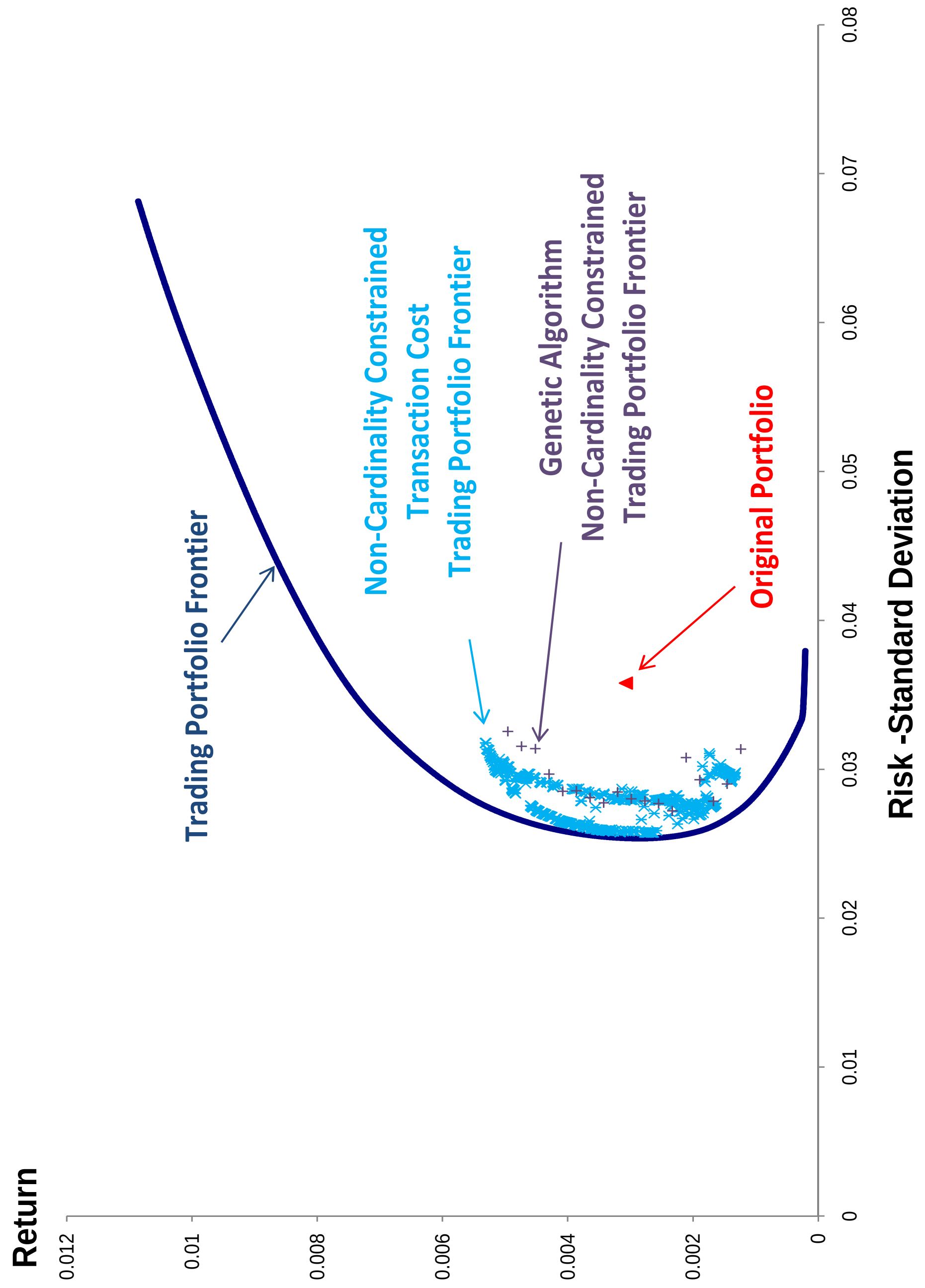

Figure 5.1: Hang Seng GA Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

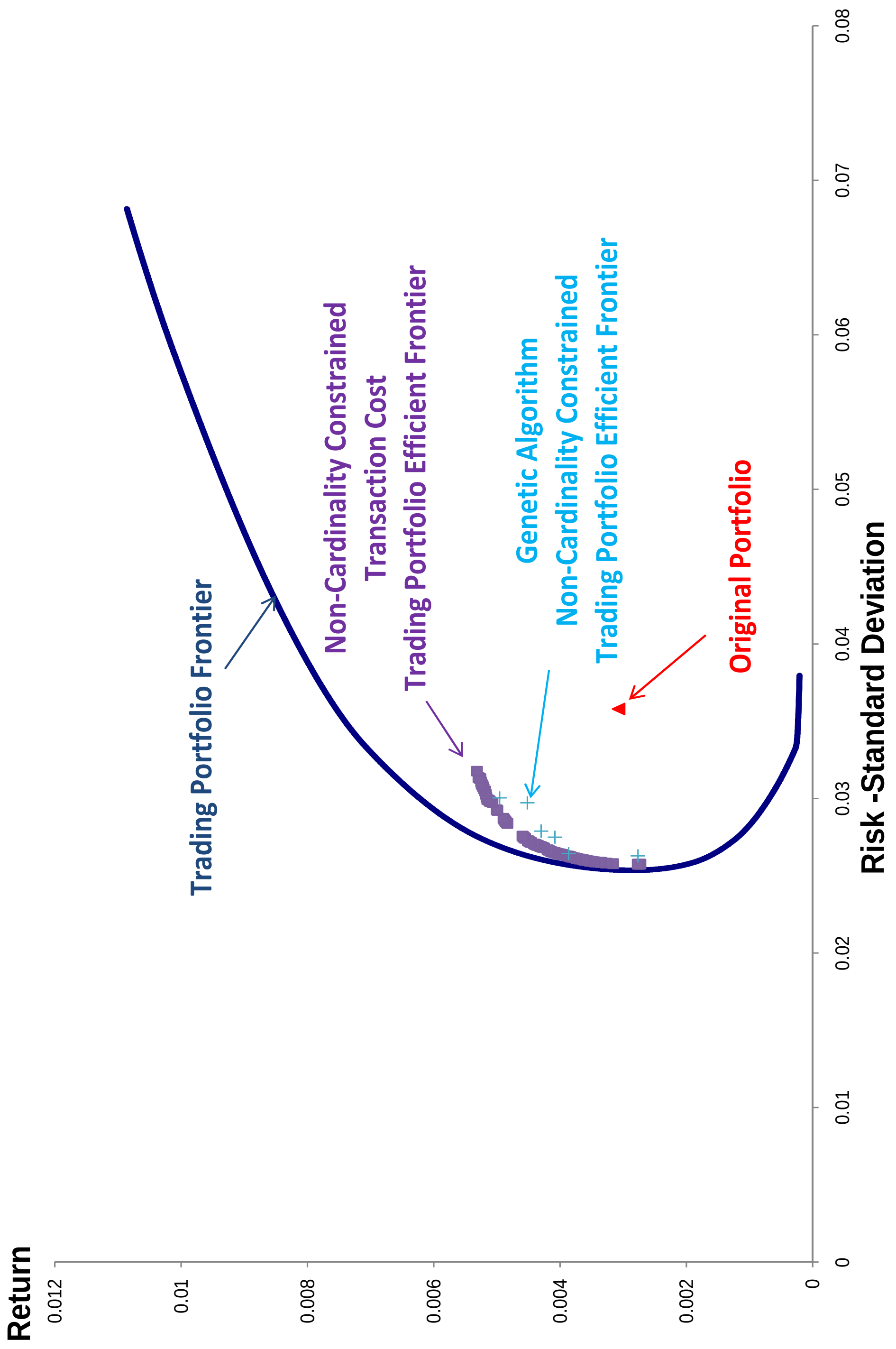

Figure 5.2: Hang Seng GA Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

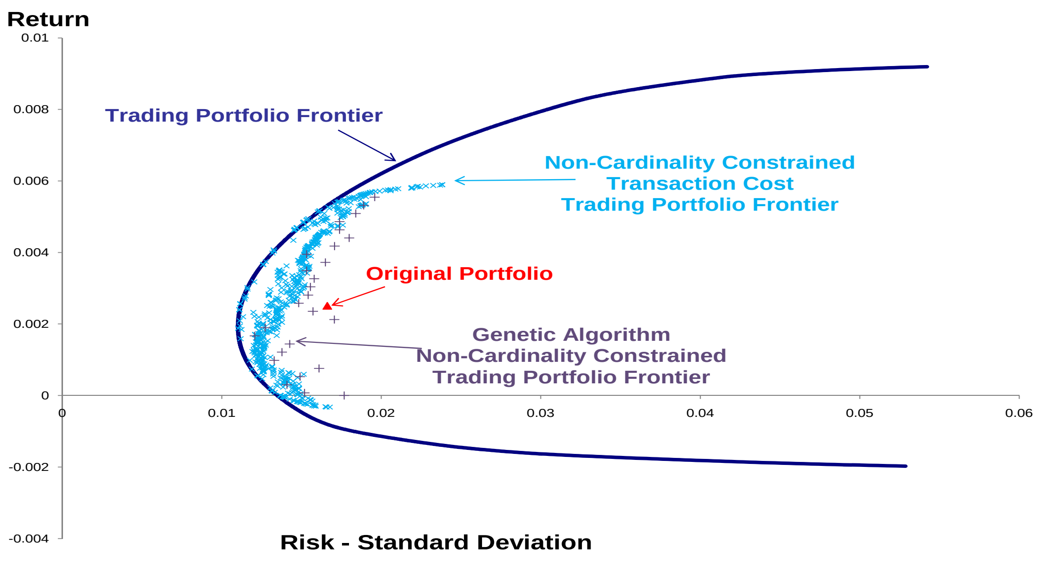

Figure 5.3: S&P 100 GA Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

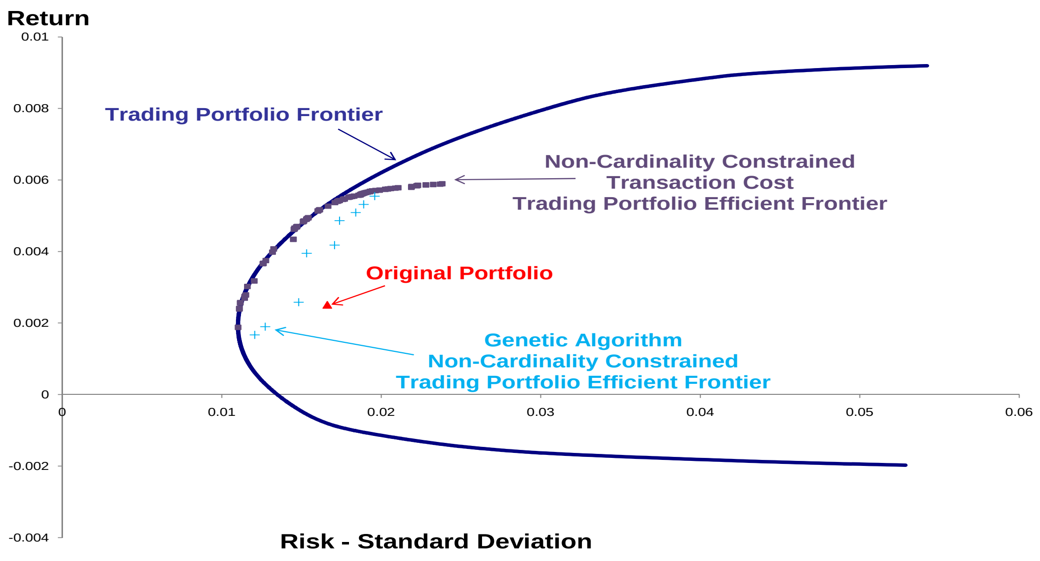

Figure 5.4: S&P 100 GA Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

if we are to once again consider the S&P 100, from Table 3.3 the mean percentage error for the non-cardinality constrained efficient frontier is given as 4.0098%. Thus, the percentage error difference is given by 16.2903% (the mean percentage error of the GA heuristic algorithm) minus 4.0098% (the mean percentage error of the non-cardinality constrained efficient frontier) which equals 12.2805% (the mean percentage error difference). In a similar fashion, from Table 3.3 the median percentage error difference of the S&P 100 is 2.1712% for the non-cardinality constrained efficient frontier. Taking 2.1712% from 15.4661% (the median percentage error of the GA heuristic algorithm) gives 13.2949% (the median percentage error difference).

In Table 5.3 we present the percentage error differences of each of the market indices made available in Table 3.3 (i.e. the Hang Seng, DAX 100, FTSE 100, S&P 100, Nikkei 225 and S&P 500). From the Table, the average mean percentage error difference for the non-cardinality constrained case of all six markets for the GA heuristic algorithm was 14.5249% while the average median percentage error difference (of all six markets) was 15.1069%.

Figures 5.1 to 5.4 graphically exhibit the GA heuristic algorithm non-cardinality constrained transaction cost trading portfolio frontier and then the efficient frontier for the Hang Seng and the S&P 100. In Figures 5.1 and 5.3 we show the non-cardinality constrained trading portfolio frontier and the trading portfolio frontier produced by the GA heuristic algorithm. In Figures 5.2 and 5.4 the non-cardinality constrained efficient frontier and the GA heuristic algorithm efficient frontier is shown. The Figures give a visual picture of the level of accuracy the GA heuristic algorithm achieved in estimating the trading portfolio and efficient frontier for the non-cardinality constrained transaction cost model. The graphics show that the majority of the GA heuristic algorithm portfolios were close to the particular non-cardinality constrained frontier.

5.5.2 Tabu Search

Table 5.4 provides the TS heuristic algorithm total computational times for the non-cardinality constrained transaction cost model. When considering how the TS heuristic algorithm performed against the optimal solution, we note that the optimal solution provides a quicker solution than the TS heuristic algorithm for the non-cardinality constrained problem.

Now considering the TS heuristic algorithm results, we see that with the exception of the Hang Seng, the average time per return level for each of the market indices was smaller than those of the GA heuristic algorithm non-cardinality constrained case. For instance, the average time per return level in the GA heuristic algorithm for the DAX 100 was 7 seconds, while the TS heuristic algorithm had an average time per return level of 5 seconds for the same market index. The S&P 500 had an average time per return level of 104 seconds for the GA heuristic algorithm and the TS heuristic algorithm average time per return level (for the S&P 500) was 74 seconds. Therefore, computationally for the non-cardinality constrained transaction cost problem the TS heuristic algorithm is quicker than that of the GA heuristic algorithm.

Market Index	Assets N	Optimal Solution		TS Heuristic Algorithm	
		Total Computational Time (seconds	Average time (seconds) per return level	Total Computational Time (seconds)	Average time (seconds) per return level
Hang Seng	31	70	0.070	188	4
DAX 100	85	271	0.271	258	5
FTSE 100	89	1273	1.273	391	8
S&P 100	98	1309	1.309	617	12
Nikkei 225	225	1490	1.490	1125	23
S&P 500	457	6016	6.016	3679	74
Average			2		21

Table 5.4: TS Heuristic Algorithm Computational Times for the Non-Cardinality Constrained Transaction Cost Model

The percentage errors and the percentage error differences for the TS heuristic algorithm are given in Tables 5.5 and 5.6 respectively. From Table 5.5 we note that for the TS heuristic algorithm the average mean percentage error (of all market indices) was 18.7697% and the average median percentage error (of all market indices) was 18.6186%. Both these values are less than the average mean and median percentage errors (of all market indices) of the GA heuristic algorithm for the non-cardinality constrained case (which was 21.0615% and 20.0682% respectively). Therefore, when considering only the GA and TS heuristic algorithms, the TS heuristic algorithm provided a closer approximation to the non-cardinality constrained efficient frontier than the GA heuristic algorithm.

Furthermore, in considering Table 5.5 we note that the TS heuristic algorithm of the FTSE 100, the Nikkei 225 and S&P 500 were the only three markets which produced better TS heuristic algorithm results than those produced by the GA heuristic algorithm. In the case of the FTSE 100 the mean and median percentage error for the GA heuristic algorithm was 16.8221% and 17.7894% (respectively) and in the case of the TS heuristic algorithm was 14.7862% and 15.6763% (respectively). Consequently, for these three markets (the FTSE 100, the Nikkei 225 and S&P 500) the TS heuristic algorithm produced better mean and median percentage errors differences (seen in Table 5.6) than the GA heuristic algorithm.

In Figures 5.5 to 5.8 we graphically present the TS heuristic algorithm results for the FTSE 100 and Nikkei 225. In the Figures, we show the market index trading portfolio frontier, the original portfolio, the non-cardinality constrained trading portfolio frontier (then it's efficient frontier) and the TS heuristic algorithm trading portfolio frontier (then it's efficient frontier). The diagrams which include the trading portfolio frontiers show that the TS heuristic produced a wide range of results; some were close to the non-cardinality constrained trading portfolio frontier while others appeared to be quite a distance away.

5.5.3 Simulated Annealing

Table 5.7 shows the SA heuristic algorithm computational times for the non-cardinality constrained transaction cost model. For every market index the SA heuristic algorithm was able to produce results in a faster time than both the GA and TS heuristic algorithms for the non-cardinality constrained case. In fact considering the average return time of all the indices, the GA heuristic algorithm was 1374 seconds, the TS heuristic algorithm was 1043 seconds and the SA heuristic algorithm was 431 seconds. Those times meant that the SA heuristic algorithm was 943 seconds faster than that of the GA heuristic algorithm and 612 seconds faster than that of the TS heuristic algorithm. Therefore, the SA heuristic algorithm had a better computational time of our three heuristic algorithms for the non-cardinality constrained transaction cost problem.

Despite the SA heuristic algorithm posting the best times of the three heuristic algorithms, Table 5.7 shows that on average the SA heuristic algorithm is almost 5 times slower that of

Percentage Error	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	S&P 500 $N = 457$	Average all markets
Mean	7.9987	36.9511	14.7862	19.5118	10.2698	23.1003	18.7697
Median	6.9831	36.9511	15.6763	17.0604	10.2698	24.7709	18.6186
Minimum	5.8707	26.1863	8.2855	2.2850	10.2698	13.6746	
Maximum	12.1578	47.7158	18.9944	37.9022	10.2698	30.8555	

Table 5.5: TS Percentage Error Results for the Transaction Cost Non-Cardinality Constrained Efficient Frontier

Percentage Error Difference	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	S&P 500 $N = 457$	Average all markets
Mean	4.0645	25.4546	13.3201	15.5020	1.5188	13.5380	12.2330
Median	4.4079	28.2179	14.3135	14.8892	2.7045	16.6670	13.5333

Table 5.6: TS Percentage Error Difference for the Transaction Cost Non-Cardinality Constrained Frontiers

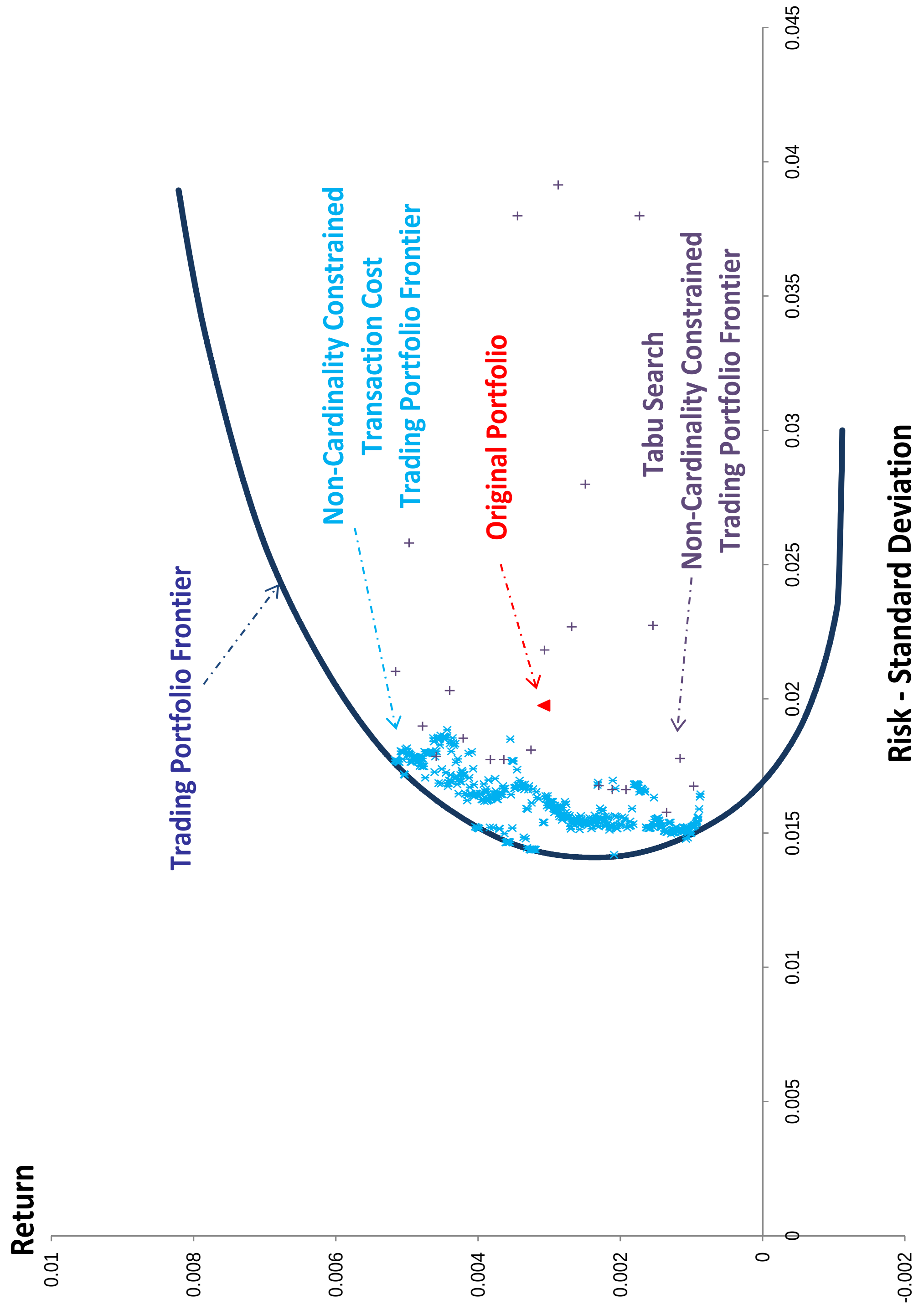

Figure 5.5: FTSE 100 TS Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

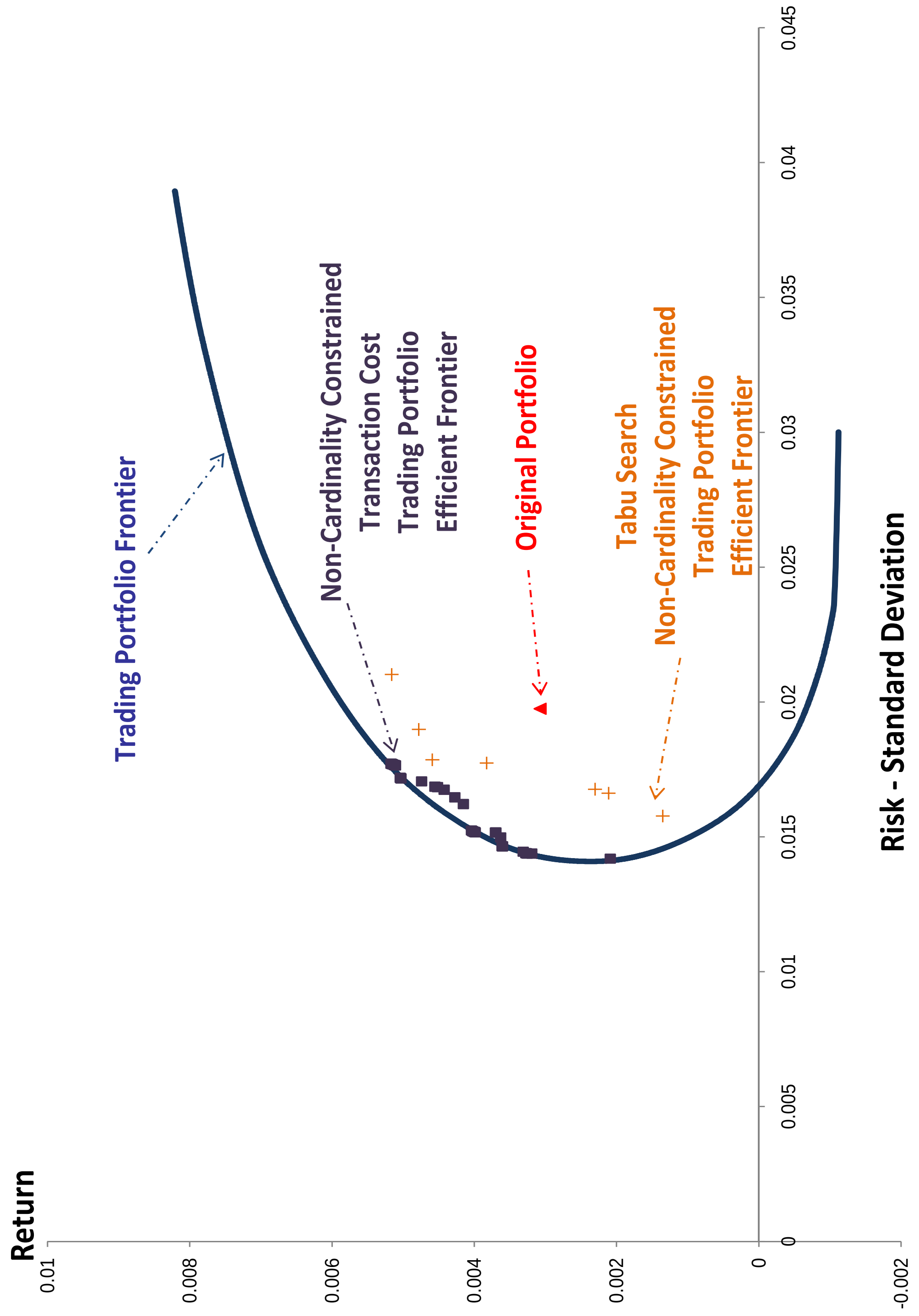

Figure 5.6: FTSE 100 TS Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

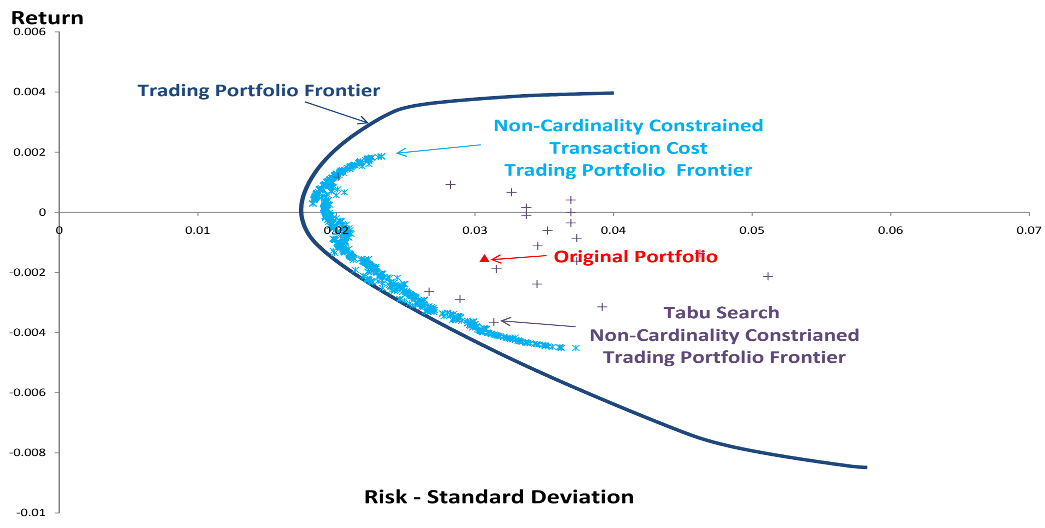

Figure 5.7: Nikkei 225 TS Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

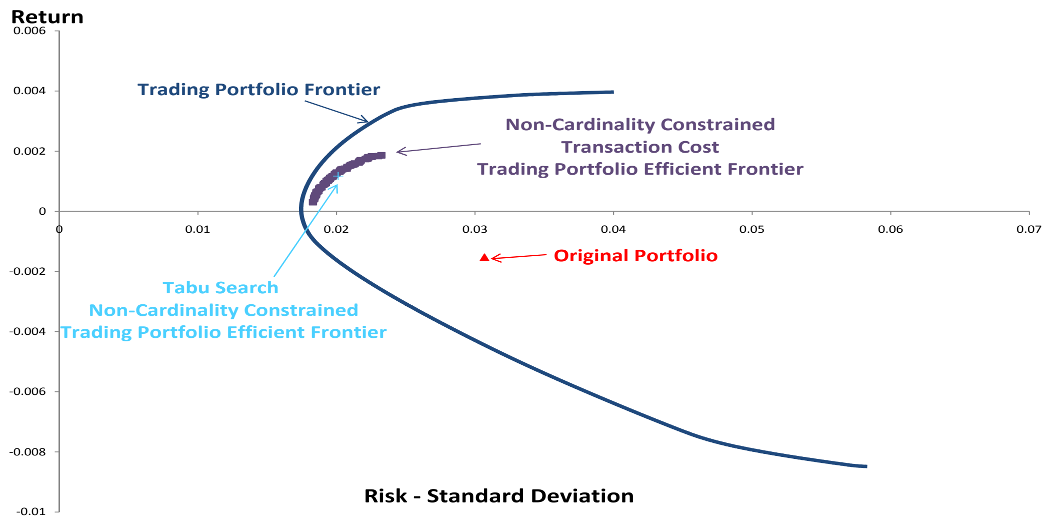

Figure 5.8: Nikkei 225 TS Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

Market Index	Assets N	Optimal Solution		SA Heuristic Algorithm	
		Total Computational Time (seconds	Average time (seconds) per return level	Total Computational Time (seconds)	Average time (seconds) per return level
Hang Seng	31	70	0.070	55	1
DAX 100	85	271	0.271	95	2
FTSE 100	89	1273	1.273	240	5
S&P 100	98	1309	1.309	355	7
Nikkei 225	225	1490	1.490	699	14
S&P 500	457	6016	6.016	1142	23
Average			2		9

Table 5.7: SA Heuristic Algorithm Computational Times for the Non-Cardinality Constrained Transaction Cost Model

the optimal solution. Hence, we can conclude that each of our heuristic algorithms were slower than the optimal solution for the non-cardinality constrained optimisation problem, i.e. CPLEX has a quicker solution time than our heuristic algorithms.

Turning our attention to percentage error results from the UEF of each of the market index for the SA heuristic algorithm, we present the percentage error results in Table 5.7. We note that the first five market indices' the SA heuristic algorithm mean and median percentage error results are greater than those produced by both those of the GA and TS heuristic algorithms. For instance, the SA heuristic algorithm for the DAX 100 has a mean percentage error of 193.2267% which compares to 23.7485% for the GA heuristic algorithm and 36.9511% for the TS heuristic algorithm. Maximum percentage error for the SA heuristic algorithm was as high as 326.8694% (given by the S&P 100) but, in both the GA and TS heuristic algorithms no maximum percentage error went above 48%.

Table 5.9 gives the percentage error differences for the SA heuristic algorithm. The average mean percentage error difference of all markets was 75.3167%. This overall average is more than 60% higher than that of the mean percentage error difference of all markets for the GA heuristic algorithm and greater than 63% higher than the mean percentage error difference of all markets for the TS heuristic algorithm. From these results we note that the SA heuristic algorithm was not as effective in calculating the non-cardinality constrained portfolio or efficient frontier as that of the GA and TS heuristic algorithms.

Percentage Error	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	S&P 500 $N = 457$	Average all markets
Mean	42.2449	193.2267	25.7778	138.4137	60.9065	30.5503	81.8533
Median	42.6443	193.2267	25.7581	99.1793	60.9065	30.8476	75.4271
Minimum	40.5471	193.2267	14.4630	76.8838	60.8534	16.6203	
Maximum	43.5434	193.2267	37.1320	326.8694	60.9597	43.8721	

Table 5.8: SA Percentage Error Results for the Transaction Cost Non-Cardinality Constrained Efficient Frontier

Percentage Error Difference	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	S&P 500 $N = 457$	Average all markets
Mean	38.3107	181.7302	24.3117	134.4039	52.1556	20.9880	75.3167
Median	40.3691	184.4935	24.3953	97.0081	53.3412	22.7437	70.3918

Table 5.9: SA Percentage Error Difference for the Transaction Cost Non-Cardinality Constrained Frontiers

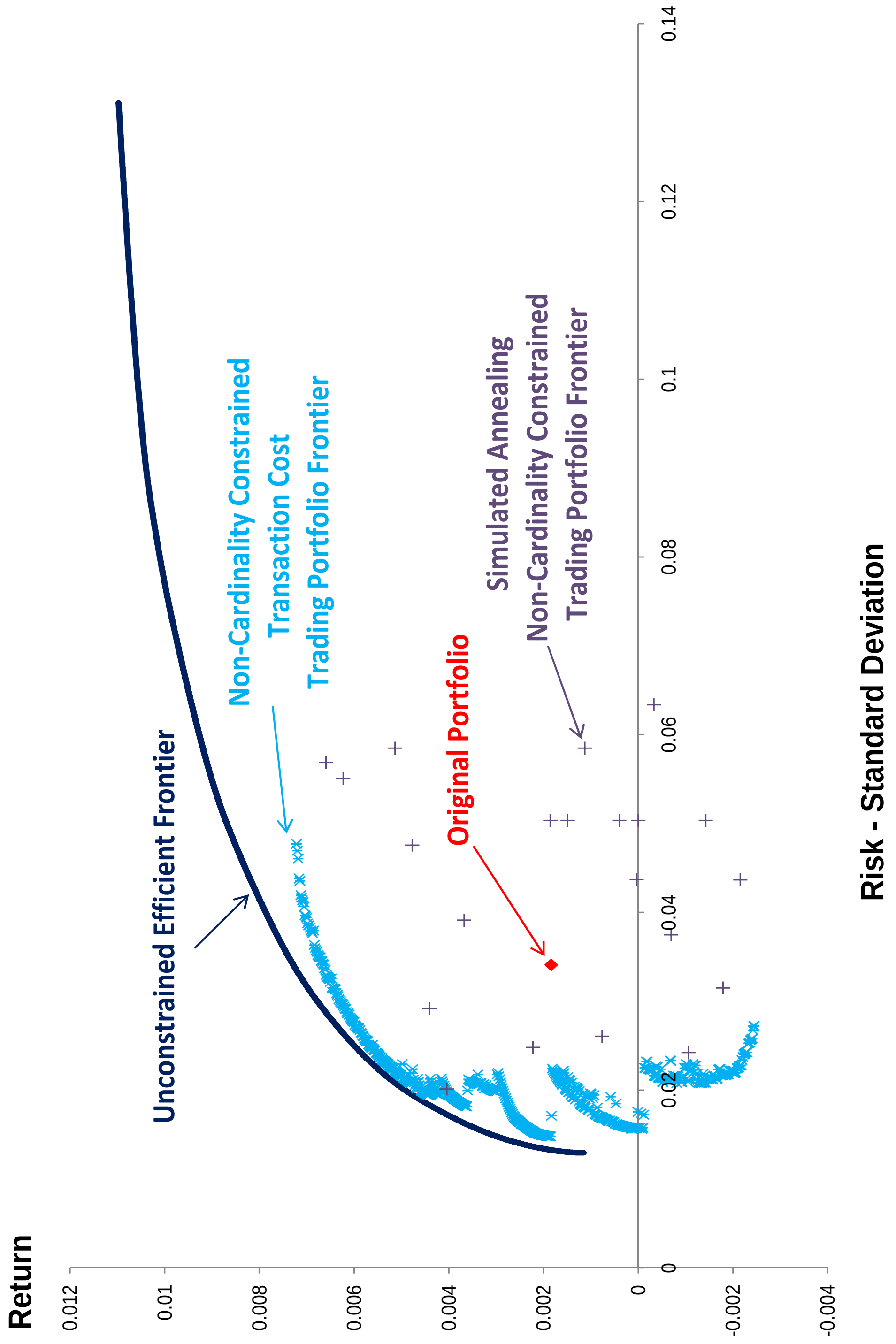

Figure 5.9: S&P 500 SA Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

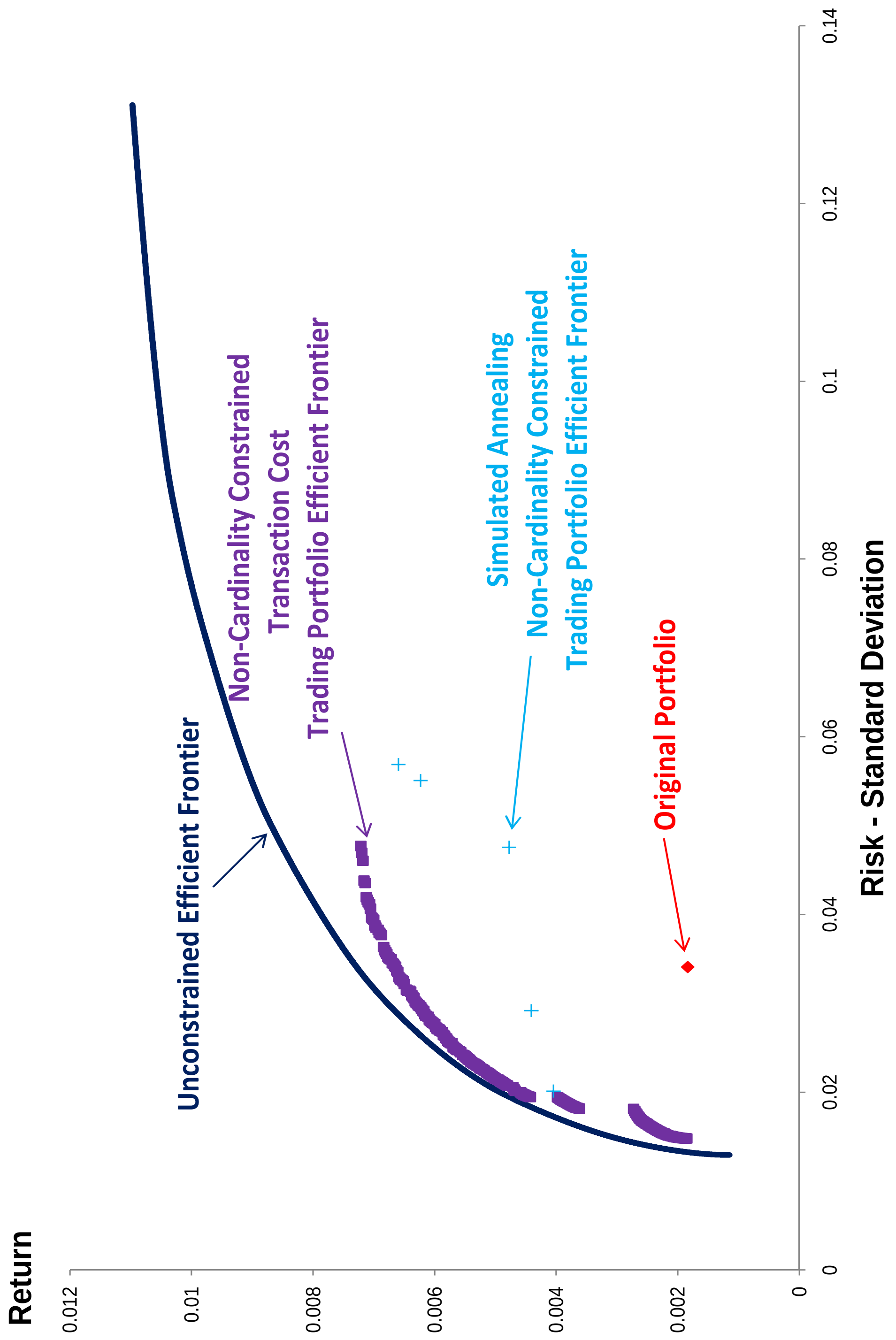

Figure 5.10: S&P 500 SA Heuristic Algorithm Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

In Figures 5.9 and 5.10 we graphically present the portfolio and then the efficient frontier for the S&P 500 (the only market index for the SA heuristic algorithm to produce a result which had mean and median percentage error better than the GA heuristic algorithm but still less than the TS heuristic algorithm). In Figure 5.9 we illustrate the non-cardinality constrained trading portfolio frontier and the trading portfolio frontier produced by the SA heuristic algorithm. In Figure 5.10 we show the non-cardinality constrained efficient frontier and the SA heuristic algorithm efficient frontier. Each Figure highlights how ineffective the SA heuristic algorithm was at estimating the portfolio and subsequently, the efficient frontier for the non-cardinality constrained transaction cost model (despite the S&P 500 being the SA heuristic algorithm's best results).

5.5.4 Pooled Heuristic Algorithms

Table 5.10 gives the pooled heuristic algorithm results for the transaction cost non-cardinality constrained efficient frontier. In the Table we state the market indices, the number of assets and the pooled heuristic algorithms of (1) GA,TS and SA, (2) GA and TS, (3) GA and SA and (4) TS and SA. For each of these pooled results, the mean and median percentage error from the UEF, the percentage error difference between the UEF and the transaction cost non-cardinality constrained efficient frontier, the total time in seconds and the time per return level (in seconds) is given. Also, made available in the Table are the average mean and median percentage error and the average time for the small and for all market indices.

If we consider the pooled heuristic algorithms of GA,TS and SA for the DAX 100 from Table 5.10 we see that it has a mean percentage error of 23.7485% and a median percentage error of 24.8043%. The total computational time to compute these results was 727 seconds which is approximately 15 seconds per return level. Then, the mean and median percentage error differences for the DAX 100 are given as 12.2520% and 16.0711% respectively.

Examining the values in the Table 5.10, we note that in each of the first five market indices' pooled heuristic algorithms' results for the GA,TS and SA was the same as the pooled heuristic algorithms' results for the GA and TS. Then, if we look at the pooled heuristic algorithm columns of GA and SA and TS and SA (for the small market indices), we notice that these are the same as the GA heuristic algorithm and TS heuristic algorithm respectively. Thus, the results for the SA heuristic algorithm played no part in any of the first five market indices' pooled heuristic algorithms' results. These results further highlight that the SA heuristic algorithm was not as valuable as the GA and TS heuristic algorithms at calculating the non-cardinality constrained efficient frontier for the five small markets.

In Figures 5.11 to 5.22 we graphically represent our pooled heuristic algorithm results for the non-cardinality constrained transaction cost problem for all six of the market indices. The pictures illustrate the original portfolio, the non-cardinality constrained transaction cost trading portfolio frontier (then the non-cardinality constrained transaction cost efficient frontier) along with the portfolio (then the efficient) frontier of the pooled heuristic algorithms' results (GA, TS and SA).

5.6 Cardinality Constrained Transaction Cost Heuristic Algorithms' Results

In this Section we present the GA (Section 5.6.1), TS (Section 5.6.2), SA (Section 5.6.3) and pooled (Section 5.6.4) heuristic algorithms' results for the cardinality constrained transaction cost trading portfolio frontier.

As was the case in the non-cardinality constrained Section above, within each of the subsections below, we look at the total computational time of the heuristic algorithm(s), the percentage error of the undominated points produced by the heuristic algorithm(s) from the unconstrained efficient frontier of the market index and the difference in percentage error of the heuristic algorithm(s) efficient frontier and that produced by the cardinality constrained efficient frontier (given in Chapter 3, Section 3.8).

Also included in each subsection are graphical illustrations of some of the results. All the pictures include the trading portfolio frontier (dark blue) of the particular market index and the original portfolio (red). Some pictures will contain the cardinality constrained trading portfolio frontier of the original portfolio (brown circles) along with the trading portfolio

Market Index	N	Percentage Error and Time (seconds)	Pooled Heuristic Algorithms							
			GA,TS,SA	Percentage error difference	GA,TS	Percentage error difference	GA,SA	Percentage error difference	TS,SA	Percentage error difference
Hang Seng	31	Mean	6.3610	2.4268	6.3610	2.4268	6.3881	2.4539	7.9987	4.0645
		Median	4.8559	2.5807	4.8559	2.5807	4.9373	2.6621	6.9831	4.4079
		Time (total)	389		334		201		243	
		Time (per return)	8		7		4		5	
DAX 100	85	Mean	23.7485	12.2520	23.7485	12.2520	23.7485	12.2520	36.9511	25.4546
		Median	24.8043	16.0711	24.8043	16.0711	24.8043	16.0711	36.9511	28.2179
		Time (total)	727		632		469		353	
		Time (per return)	15		13		9		7	
FTSE 100	89	Mean	14.8851	13.4190	14.8851	13.4190	16.8221	15.3560	14.7862	13.3201
		Median	17.3881	16.0253	17.3881	16.0253	17.7893	16.4266	15.6763	14.3135
		Time (total)	1182		942		791		631	
		Time (per return)	24		19		16		13	
S&P 100	98	Mean	13.3370	9.3272	13.3370	9.3272	16.2903	12.2805	19.5118	15.5020
		Median	9.2013	7.0301	9.2013	7.0301	15.4661	13.2949	17.0604	14.8892
		Time (total)	1713		1358		1096		972	
		Time (per return)	34		27		22		19	
Nikkei 225	225	Mean	10.6132	1.8623	10.6132	1.8623	14.9424	6.1914	10.2698	1.5188
		Median	10.6132	3.0479	10.6132	3.0479	14.9424	8.7861	10.2698	2.7045
		Time (total)	3041		2342		1916		1824	
		Time (per return)	61		47		38		36	
S&P 500	457	Mean	32.6596	23.0973	37.2423	27.6800	40.2550	30.6927	24.0611	14.4988
		Median	29.2228	21.1189	30.8555	22.7516	34.3203	26.2164	26.1844	18.0805
		Time (total)	10038		8896		6359		4821	
		Time (per return)	201		178		127		96	
Average (all markets)		Mean	16.9341	10.3975	17.6979	11.1613	19.7411	13.2045	18.9298	12.3931
		Median	16.0143	10.9790	16.2864	11.2511	18.0997	13.9096	18.8542	13.7689
		Time (total)	2915		2484		1872		1474	
		Time (per return)	58		50		37		29	

Table 5.10: Pooled Heuristic Algorithms' Results for the Transaction Cost Non-Cardinality Constrained Frontiers

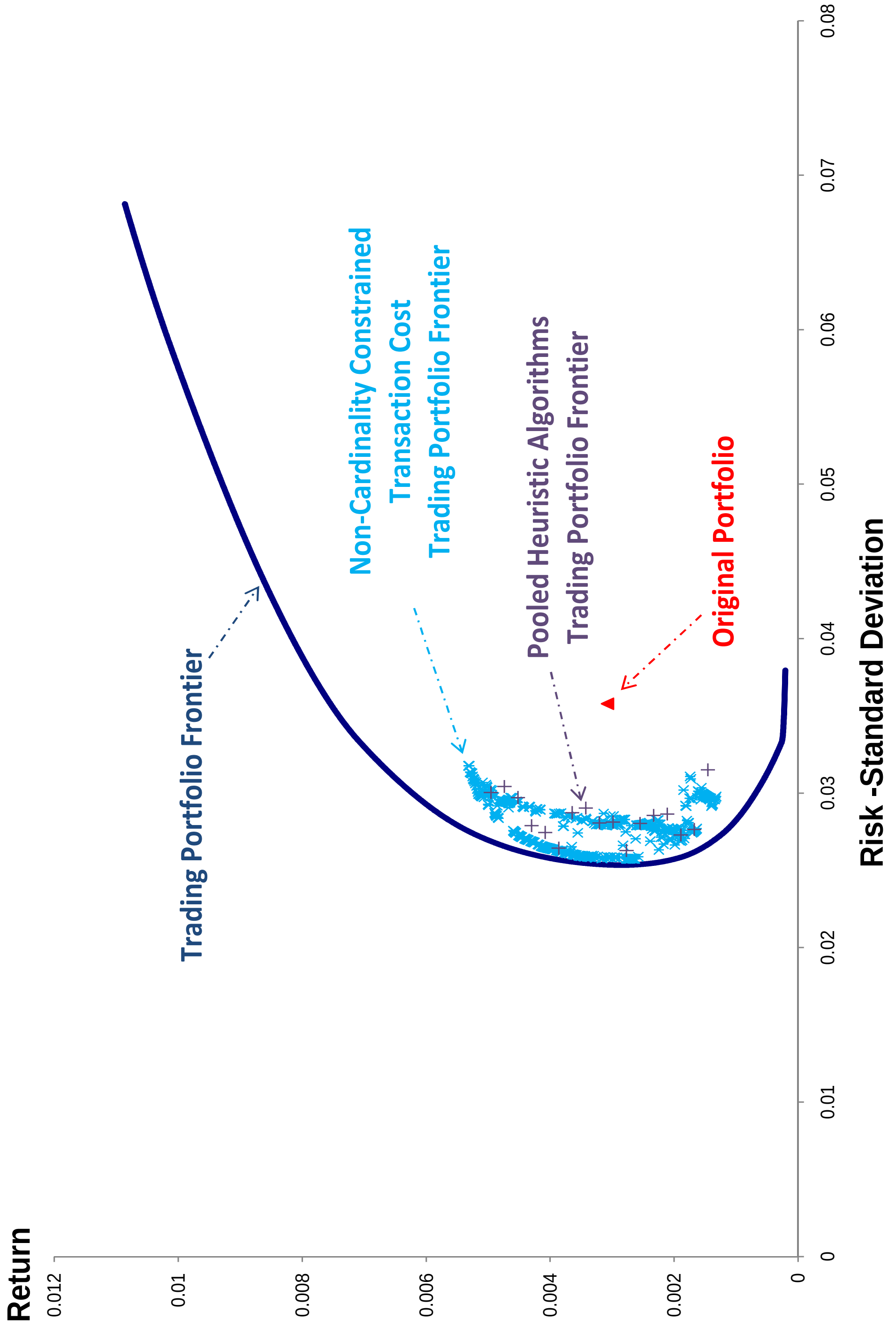

Figure 5.11: Hang Seng Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

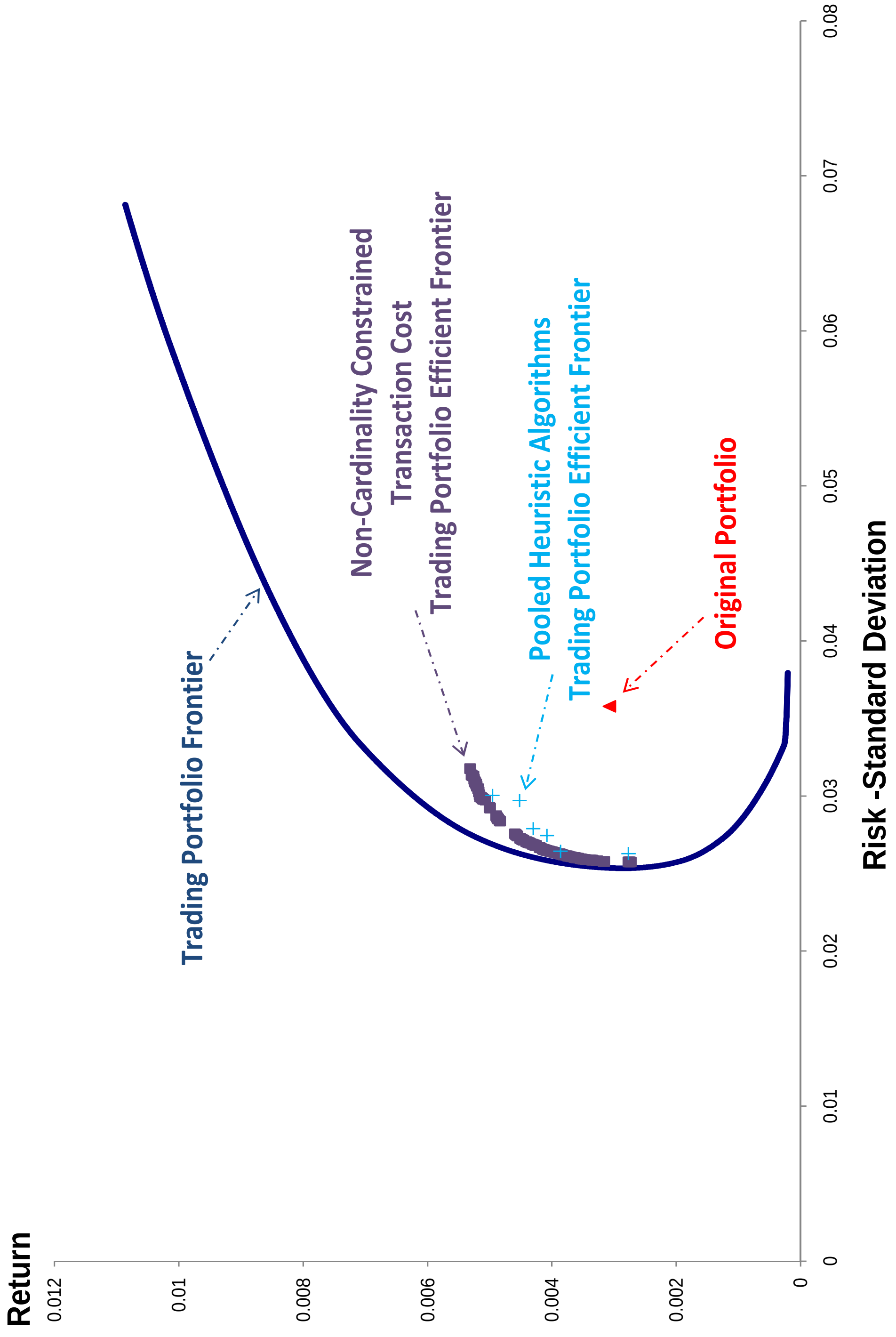

Figure 5.12: Hang Seng Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

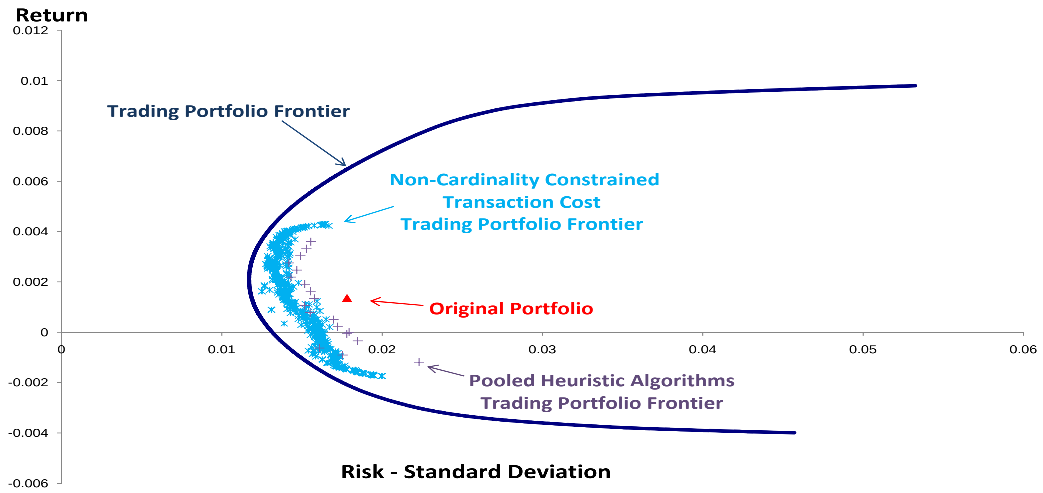

Figure 5.13: DAX 100 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

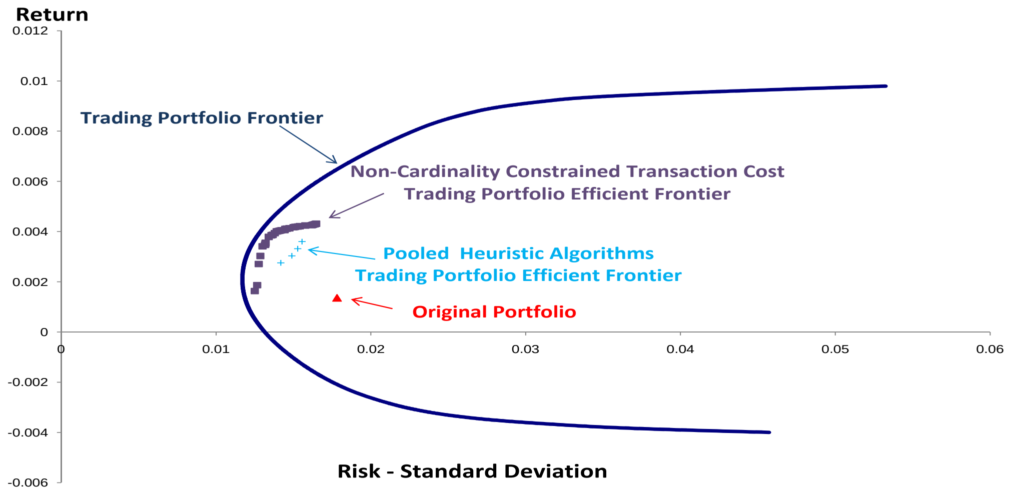

Figure 5.14: DAX 100 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

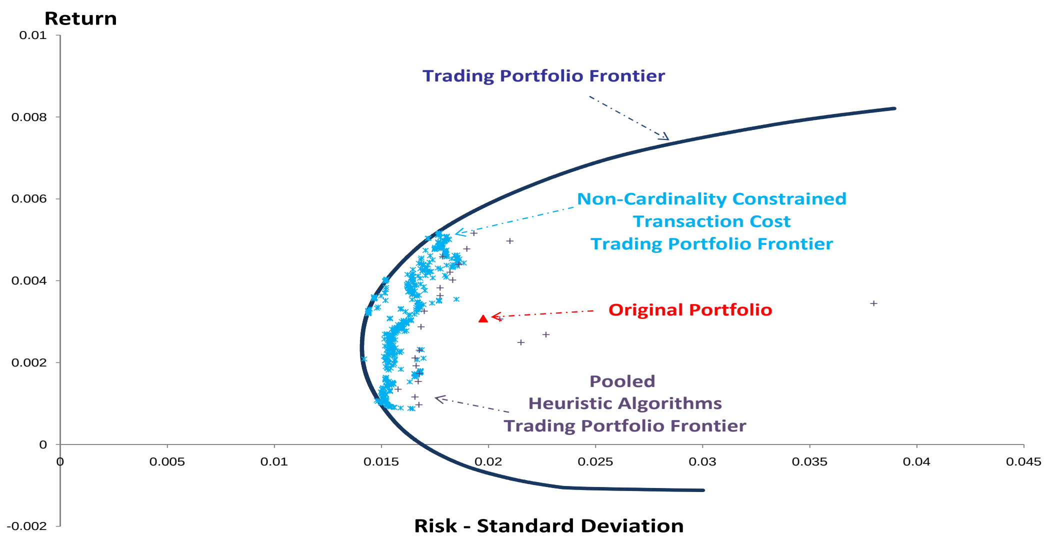

Figure 5.15: FTSE 100 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

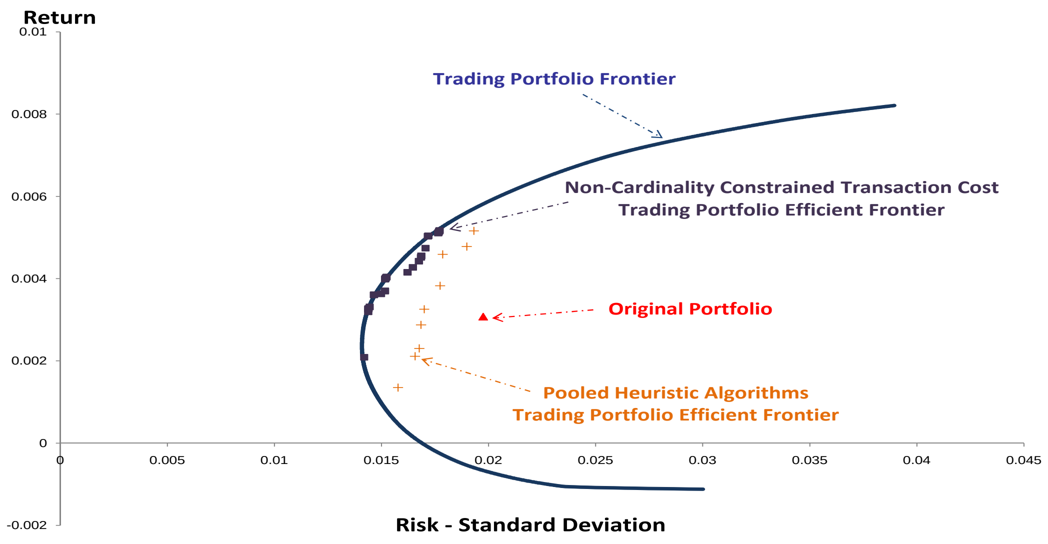

Figure 5.16: FTSE 100 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

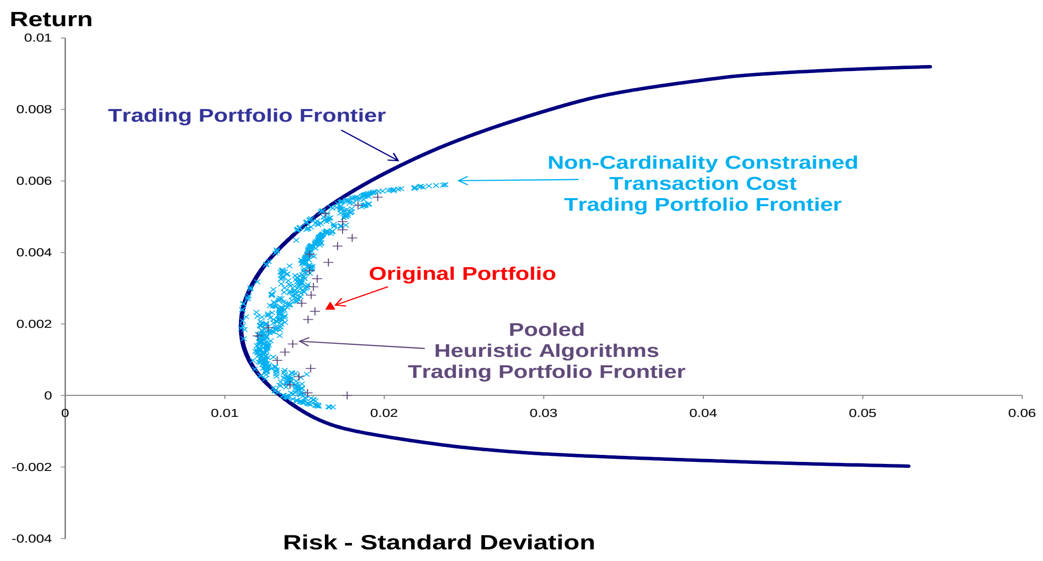

Figure 5.17: S&P 100 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

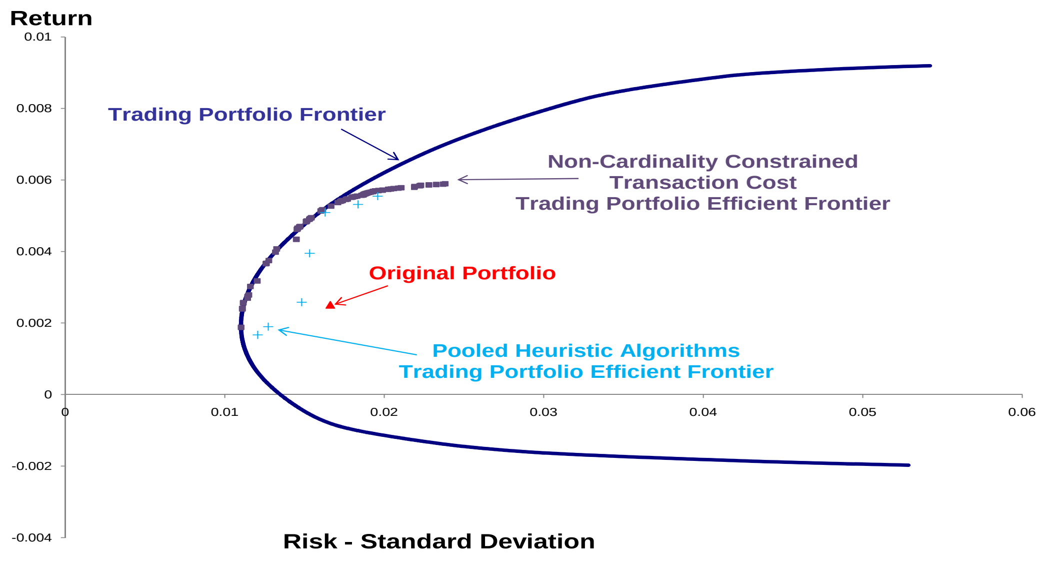

Figure 5.18: S&P 100 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

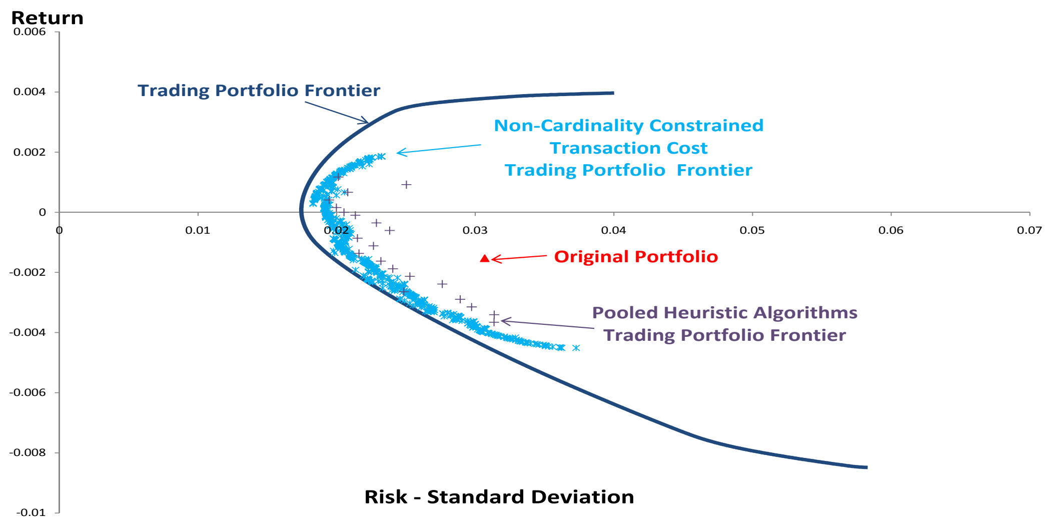

Figure 5.19: Nikkei 225 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

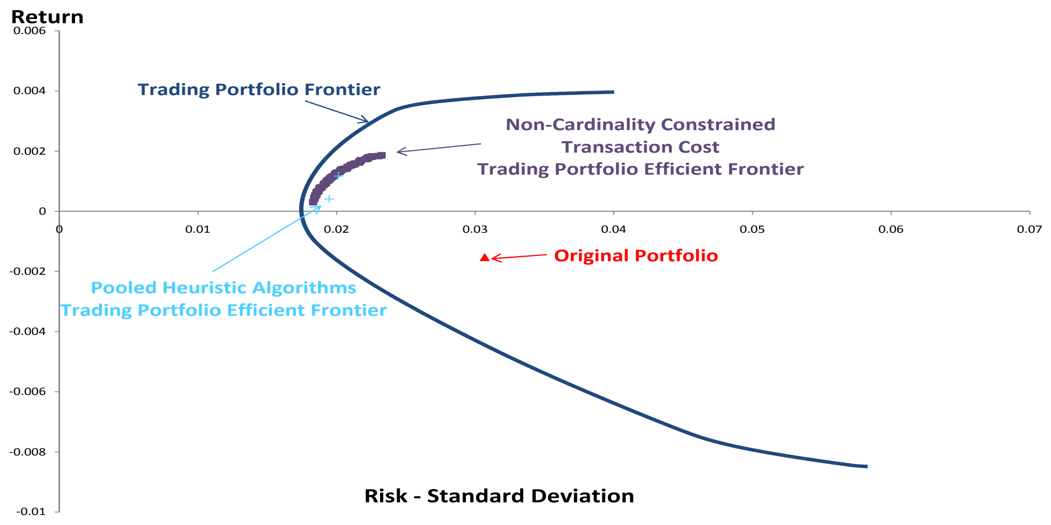

Figure 5.20: Nikkei 225 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

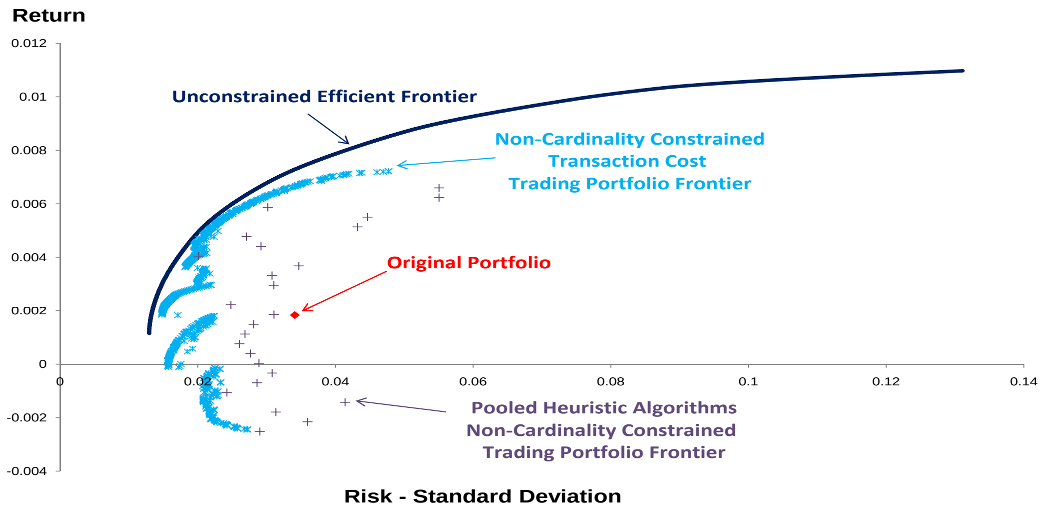

Figure 5.21: S&P 500 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Frontier

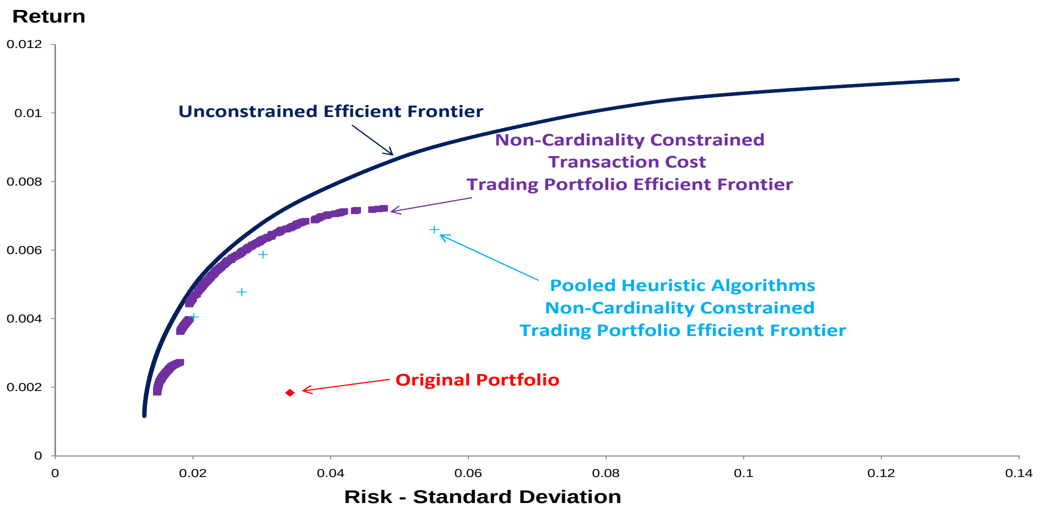

Figure 5.22: S&P 500 Pooled Heuristic Algorithms Transaction Cost Non-Cardinality Constrained Trading Portfolio Efficient Frontier

frontier produced by the heuristic algorithm (orange triangles for the individual heuristics of GA, TS and SA while pooled heuristic results are blue); the other pictures will contain the cardinality constrained efficient frontier of the original portfolio (pink squares). Here we note that some of the crosses, squares and triangles have been made larger to aid the reader in being able to see them.

5.6.1 Genetic Algorithm

Table 5.11 gives the GA heuristic algorithm total computational times for the cardinality constrained transaction cost model. Interestingly, the times are almost the same per return level as those of the GA heuristic algorithm non-cardinality constrained transaction cost model when only considering the small market indices. For instance, the time per return level for both transaction cost models (cardinality and non-cardinality constrained) for the FTSE 100 is 11 seconds; the Nikkei 225 non-cardinality constrained problem times were 24 seconds per return level and 23 seconds per return level for the cardinality constrained problem. But, when we consider the S&P 500 a time difference in the cardinality constrained model and the non-cardinality constrained model is revealed. The time per return level for the S&P 500 turned out to be 104 seconds for the non-cardinality constrained case and 147 seconds for the cardinality constrained case. The average time for all markets was 1704 seconds for the cardinality constrained model and 1374 seconds for the non-cardinality constrained model. Hence, the cardinality constrained transaction cost optimisation for the GA heuristic algorithm took longer than the non-cardinality constrained problem.

When comparing the times of the optimal solution to that of the heuristic algorithm one notes that for the Hang Seng the times are relatively the same with average time per return level being 1.44 seconds for the optimal solution to 2 seconds for the GA heuristic algorithm. But, as we move into larger markets the relative time difference changes significantly. For instance, the Nikkei 225 market index had a time of 11330.74 seconds per return level for the optimal solution while the GA heuristic algorithm posted a time of only 147 seconds per return level.

Table 5.12 gives the GA heuristic algorithm error results for the transaction cost cardinality

Market Index	Assets N	Optimal Solution		GA Heuristic Algorithm	
		Total Computational Time (seconds	Average time (seconds) per return level	Total Computational Time (seconds)	Average time (seconds) per return level
Hang Seng	31	72	1.44	124	2
DAX 100	85	348	6.96	404	8
FTSE 100	89	3426	68.52	570	11
S&P 100	98	127746	2554.92	613	12
Nikkei 225	225	566537	11330.74	1160	23
S&P 500	457			7355	147
Average			2793		34

Table 5.11: GA Heuristic Algorithm Computational Times for the Cardinality Constrained Transaction Cost Model

constrained efficient frontier. From the Table, the average mean and median percentage error of the small markets are given as 15.9408% and 15.7131% respectively. Then for all market indices the average mean and median percentage error are given as 21.3713% and 19.7337% respectively. If we were to compare these to the average of the small and all the market indices of the GA heuristic algorithm non-cardinality constrained case we find that the averages are almost the same.

Furthermore, if we are to consider Table 5.13 where we subtract the percentage error difference between the UEF and the cardinality constrained transaction cost efficient frontier, we realize that each of the small market index is actually closer to it's cardinality constrained transaction cost efficient frontier than those in the non-cardinality constrained case for the GA heuristic algorithm. For instance, the DAX 100 percentage error difference is 0.9118% for the cardinality constrained problem but only 2.4539% for the non-cardinality constrained problem; the FTSE 100 has a percentage error difference of 9.3022% when considering the cardinality constrained problem and this rises to 15.3560% for the non-cardinality constrained problem. Then considering all the small markets, the average mean and median percentage error differences of the cardinality constrained case is approximately 4% and 5% (respectively) lower than for the non-cardinality constrained case. This leads us to conclude that given an existing portfolio, the GA heuristic algorithm would produce better results on the cardinality constrained case than the non-cardinality constrained case for the transaction cost model for

Percentage Error	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	S&P 500 $N = 457$	Average all markets
Mean	11.8986	7.9792	14.7586	21.9592	23.1082	48.5237	21.3713
Median	9.6337	7.9792	13.6133	24.2900	23.0492	39.8369	19.7337
Minimum	6.8931	0.5147	12.1323	9.9724	21.2941	28.3904	
Maximum	19.0396	15.4436	19.9292	28.9381	24.5726	87.2460	

Table 5.12: GA Percentage Error Results for the Transaction Cost Cardinality Constrained Efficient Frontier

Percentage Error Difference	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	Average all markets
Mean	0.9118	3.5191	9.3022	10.8250	5.5942	6.0305
Median	0.2197	3.2701	8.8365	12.7336	6.2237	6.2567

Table 5.13: GA Percentage Error Difference for the Transaction Cost Cardinality Constrained Frontiers

small market indices.

Figures 5.23 to 5.26 graphically depicts the GA heuristic algorithm results for the Hang Seng and the DAX 100. Figures 5.23 and 5.25 show the trading portfolio frontiers for the cardinality constrained transaction cost model and the GA heuristic algorithm. Figures 5.24 and 5.26 presents the efficient frontier of the cardinality constrained transaction cost model and the GA heuristic algorithm. The Figures give a visual idea of the quality of the results for the GA heuristic algorithm cardinality constrained transaction cost model.

5.6.2 Tabu Search

Table 5.14 provides the TS heuristic algorithm computational times for the cardinality constrained transaction cost model. Considering all the market indices the TS heuristic algorithm was quicker than the GA heuristic algorithm for the cardinality constrained transaction cost model, with average times per return level being 28 seconds and 34 seconds respectively. When we compare the TS heuristic algorithm model times per return level for the cardinality constrained case to the non-cardinality constrained case on average we see that the cardinality constrained case produced a higher time: 28 seconds (cardinality constrained problem) to 21 seconds (non-cardinality constrained problem).

Market Index	Assets N	Optimal Solution		TS Heuristic Algorithm	
		Total Computational Time (seconds	Average time (seconds) per return level	Total Computational Time (seconds)	Average time (seconds) per return level
Hang Seng	31	72	1.44	120	2
DAX 100	85	348	6.96	307	6
FTSE 100	89	3426	68.52	583	12
S&P 100	98	127746	2554.92	611	12
Nikkei 225	225	566537	11330.74	1122	22
S&P 500	457			5585	112
Average			2793		28

Table 5.14: TS Heuristic Algorithm Computational Times for the Cardinality Constrained Transaction Cost Model

When we compare the solution times of the optimal solution to that of the heuristic

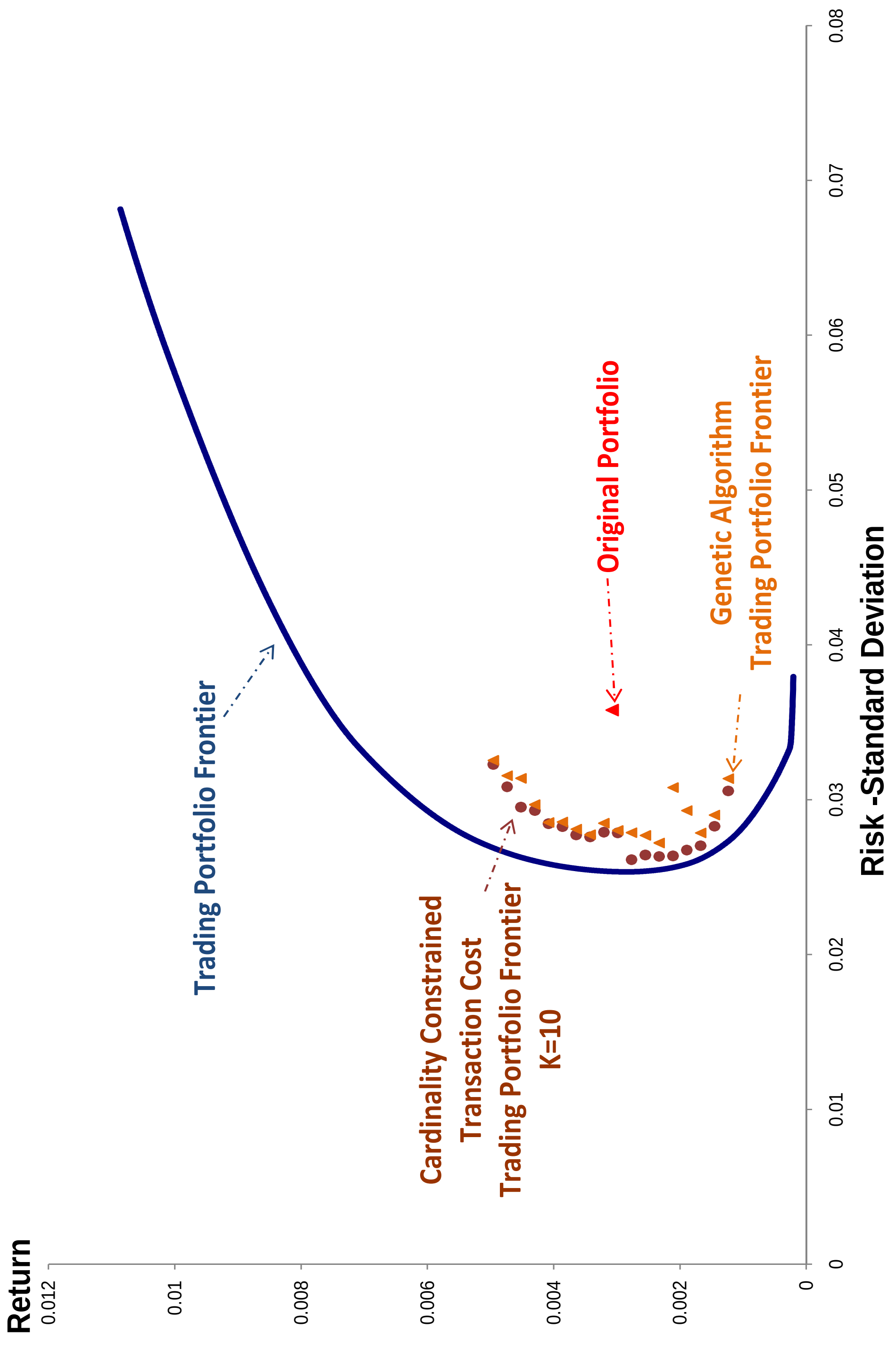

Figure 5.23: Hang Seng GA Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Frontier

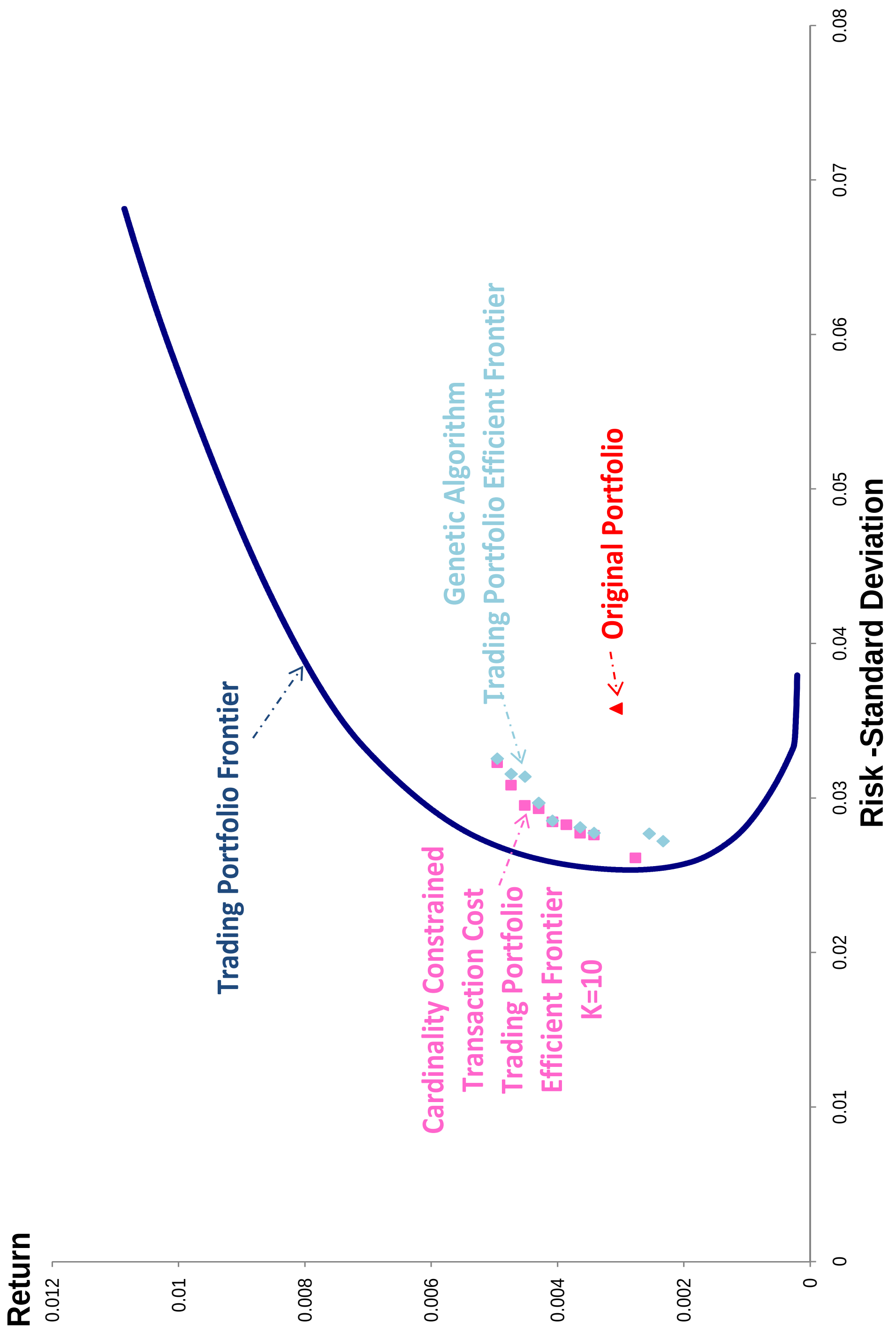

Figure 5.24: Hang Seng GA Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

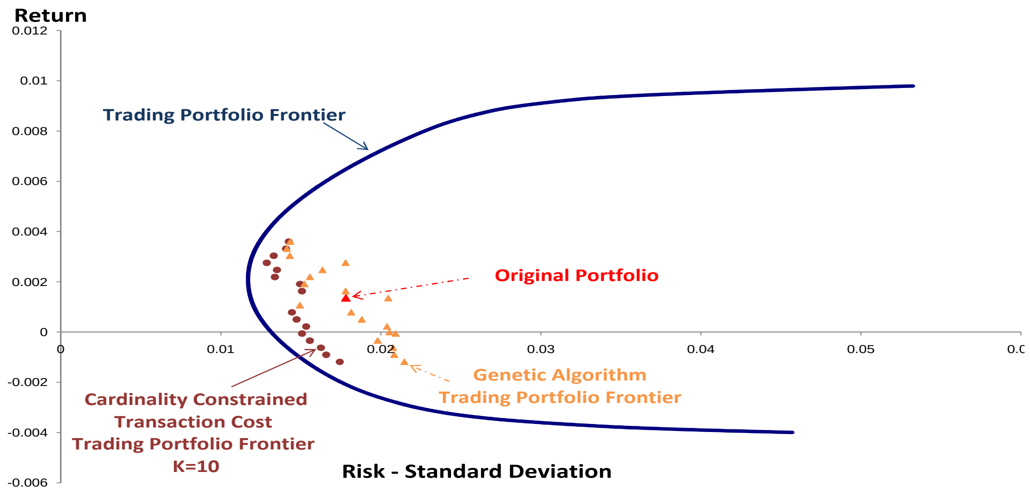

Figure 5.25: DAX 100 GA Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Frontier

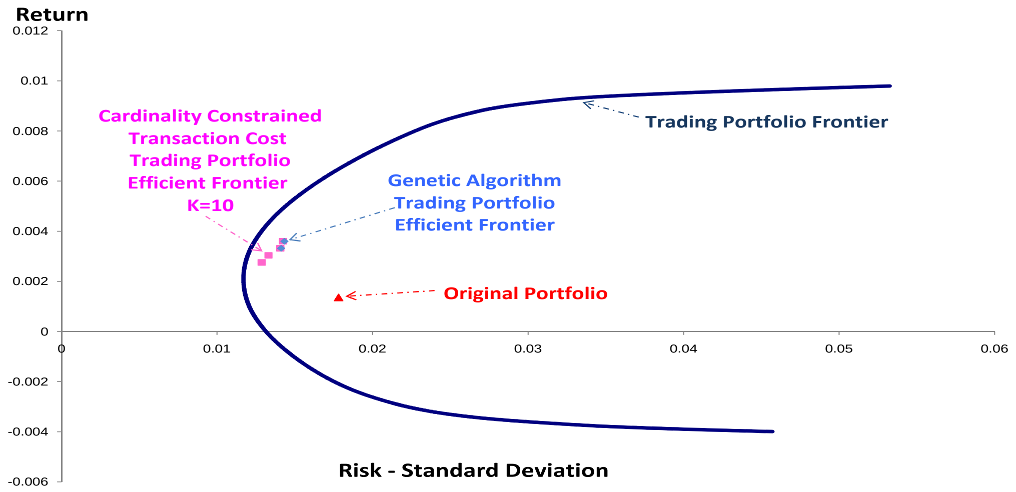

Figure 5.26: DAX 100 GA Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

algorithm solution on average we note that the heuristic algorithm solution performed faster: 2793 seconds (optimal solution) to 28 seconds (TS heuristic algorithm solution).

Table 5.15 gives the percentage errors of the TS heuristic algorithm from the UEF. The mean average percentage error of all small market indices is 20.6322% and the median average percentage error of all small market indices is 19.1584%. As was the case with the GA heuristic algorithm cardinality constrained problem the percentage errors for the TS heuristic algorithm seem to be higher on average than the TS heuristic algorithm non-cardinality constrained case. Additionally, if we are to consider the percentage error difference (Table 5.16) we note that the TS heuristic algorithm had an average small market mean percentage error difference of 9.3143% and an average small market median percentage error difference of 8.3061% over all small markets. These differences means that in the cardinality constrained problem the mean percentage error was 2.6577% closer and the median percentage error was 4.6005% closer than in the non-cardinality constrained problem for all small markets. Therefore, the TS heuristic algorithm was better for the cardinality constrained case than for the non-cardinality constrained case.

Comparing the TS heuristic algorithm results to the GA heuristic algorithm results for the small markets of the cardinality constrained case on average the GA heuristic algorithm gave better mean and median percentage error differences than the TS heuristic algorithm. But, the cardinality constrained problem for the TS heuristic algorithm had two markets indices which produced better results than the GA heuristic algorithm. In this case they are the FTSE 100 and the S&P 100. The FTSE 100 had a mean and median percentage error difference of 3.7595% and 3.5802% (respectively) for the TS heuristic algorithm while, they were 9.3022% and 8.8365% (respectively) for the GA heuristic algorithm. The S&P 100 had a mean and median percentage error difference of 7.1011% and 7.3239% (respectively) for the TS heuristic algorithm while, they were 10.8250% and 12.7336% (respectively) for the GA heuristic algorithm.

In Figures 5.27 to 5.30 we illustrate the two market indices which offered the two lowest percentage error differences: the FTSE 100 and the Nikkei 225. Each picture shows the TS heuristic algorithm estimation of the cardinality constrained portfolio or efficient fron-

Percentage Error	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	S&P 500 $N = 457$	Average all markets
Mean	20.9953	31.5762	9.2159	18.2353	23.1385	20.1937	20.5592
Median	16.8399	31.5762	8.3570	15.8803	23.1385	19.1133	19.1509
Minimum	9.3113	25.4758	5.7822	0.08622	22.0941	12.4137	
Maximum	36.8349	37.6767	14.3672	41.7926	24.1829	28.4819	

Table 5.15: TS Percentage Error Results for the Transaction Cost Cardinality Constrained Efficient Frontier

Percentage Error Difference	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	Average all markets
Mean	10.0085	20.0779	3.7595	7.1011	5.6245	9.3143
Median	6.9865	20.3269	3.5802	4.3239	6.3130	8.3061

Table 5.16: TS Percentage Error Difference for the Transaction Cost Cardinality Constrained Frontiers

tier. Depicted in Figures 5.27 and 5.29 are the trading portfolio frontiers for the cardinality constrained transaction cost problem and the TS heuristic algorithm. Figures 5.28 and 5.30 presents the efficient frontier of the cardinality constrained transaction cost model and the TS heuristic algorithm.

5.6.3 Simulated Annealing

Table 5.17 gives the SA heuristic algorithm computational times for the cardinality constrained transaction cost model. The times are approximately 30% those of the GA and TS heuristic algorithm cardinality constrained model. Thus, computationally the SA heuristic algorithm was the quickest of the three heuristic algorithms for the cardinality constrained problem. Then when comparing to the SA heuristic algorithm cardinality and non-cardinality constrained results, they are relatively the same with both problems taking an average of 9 seconds per return level.

Comparing the optimal solution to that of the heuristic algorithm solution on average we note that the heuristic algorithm solution performed faster: 2793 seconds (optimal solution) to 9 seconds (SA heuristic algorithm solution). In fact, each heuristic algorithms gave much better solution times than the optimal solution for the cardinality constrained case.

Market Index	Assets N	Optimal Solution		SA Heuristic Algorithm	
		Total Computational Time (seconds	Average time (seconds) per return level	Total Computational Time (seconds)	Average time (seconds) per return level
Hang Seng	31	72	1.44	56	1
DAX 100	85	348	6.96	145	3
FTSE 100	89	3426	68.52	229	5
S&P 100	98	127746	2554.92	260	5
Nikkei 225	225	566537	11330.74	444	9
S&P 500	457			1592	32
Average			2793		9

Table 5.17: SA Heuristic Algorithm Computational Times for the Cardinality Constrained Transaction Cost Model

Tables 5.18 and 5.19 gives the percentage error and the percentage error differences for

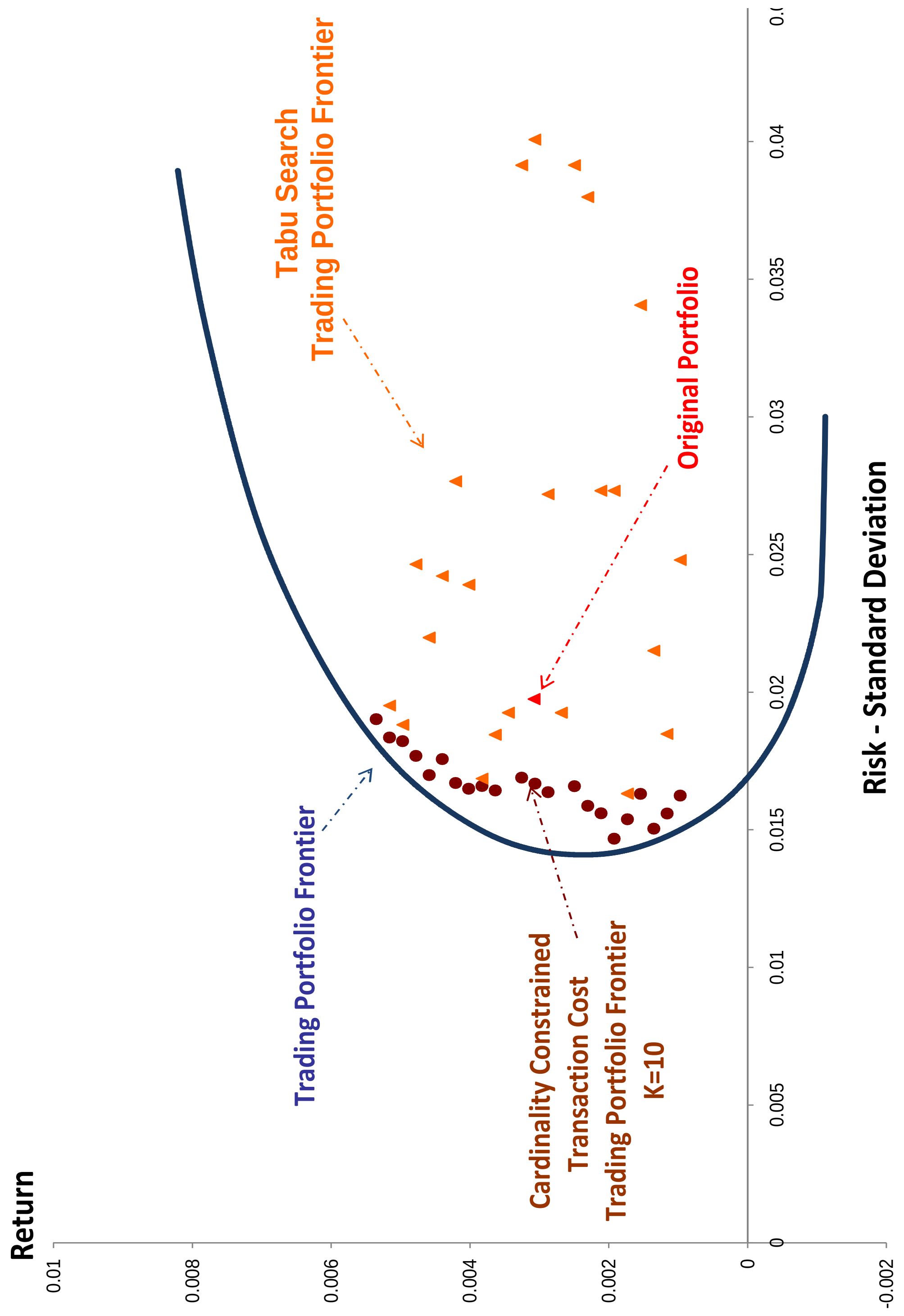

Figure 5.27: FTSE 100 TS Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Frontier

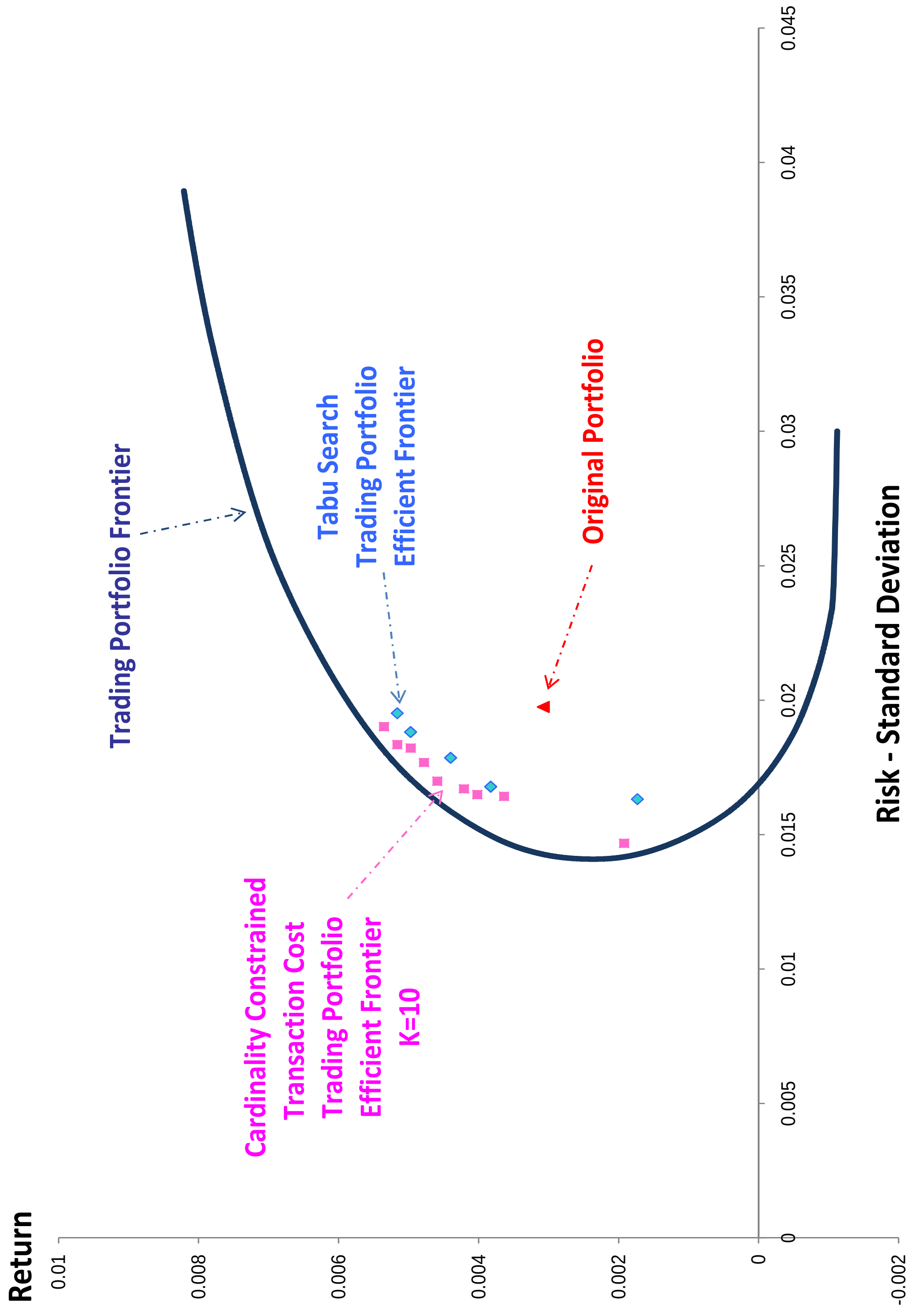

Figure 5.28: FTSE 100 TS Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

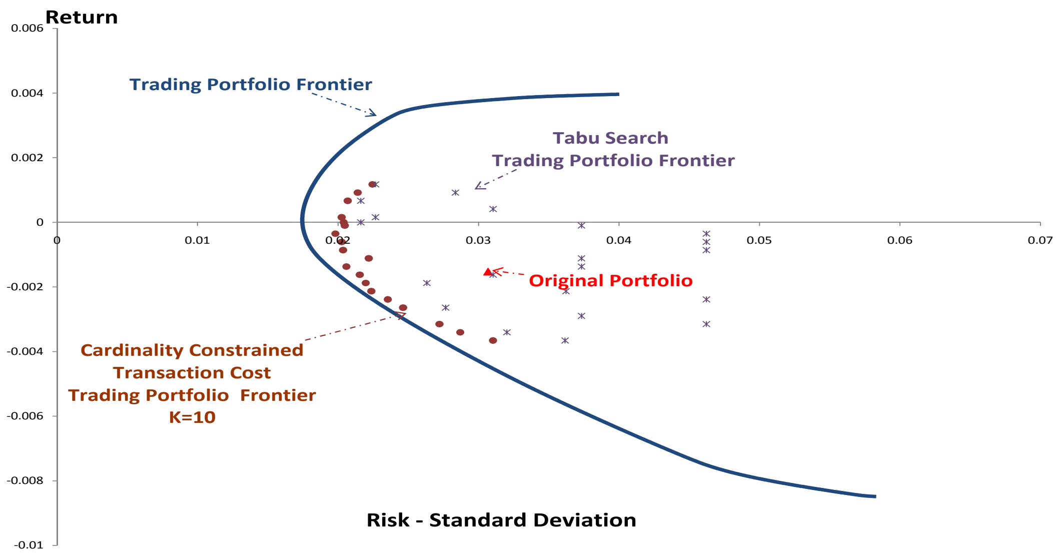

Figure 5.29: Nikkei 225 TS Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Frontier

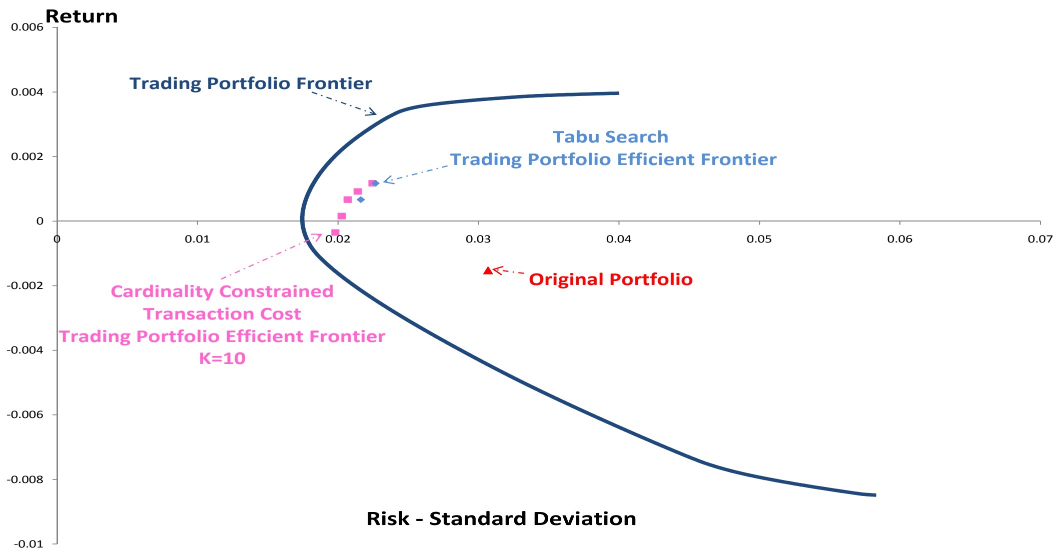

Figure 5.30: Nikkei 225 TS Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

the market indices. Interestingly, the cardinality constrained problem for the SA heuristic algorithm produced percentage errors which are better than the percentage errors of the non-cardinality constrained problem (for the SA heuristic algorithm). Subsequently, the percentage error differences were better (i.e. the results were closer to it's efficient frontier over the small market indices); the mean and median percentage error differences was approximately 45% and 37% (respectively) less than the non-cardinality constrained case. Subsequently, the SA heuristic algorithm of the cardinality constrained transaction cost model provides a better approximation for it's efficient frontier than that of the non-cardinality constrained case (for the SA heuristic algorithm).

Additionally, despite doing better than the non-cardinality case, the SA heuristic algorithm was the least effective (of our three heuristic algorithms) at calculating the cardinality constrained efficient frontier. Once again as was the case in the non-cardinality constrained case of the SA heuristic algorithm, the cardinality constrained case of the SA heuristic algorithm has the highest mean and median percentage error results of the three heuristic algorithms. Examining the average of the small market indices, the SA heuristic algorithm had average mean and median error differences of 41.4967% and 42.6630% (respectively); the GA heuristic algorithm had average mean and median percentage error differences of 6.0305% and 6.2567% (respectively); the TS heuristic algorithm had average mean and median error differences of 9.3143% and 8.3061% (respectively).

In Figures 5.31 to 5.32 we graphically present the SA heuristic algorithm results for the S&P 100. In the Figures, we show the market index trading portfolio frontier, the original portfolio, the non-cardinality constrained trading portfolio frontier (then it's efficient frontier) and the SA heuristic algorithm trading portfolio frontier (then it's efficient frontier). The pictures illustrate that most results produced by the SA heuristic algorithm are far away from the trading portfolio frontier.

5.6.4 Pooled Heuristic Algorithms

Table 5.20 gives the pooled heuristic algorithm results for the transaction cost cardinality constrained efficient frontier. Considering the pooled heuristic algorithms of TS and SA for

Percentage Error	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	S&P 500 $N = 457$	Average all markets
Mean	41.7974	36.0473	42.5471	31.5143	112.1701	38.2443	50.3868
Median	41.7974	38.7018	44.0942	31.5143	111.4188	29.9331	49.5766
Minimum	40.9506	14.2890	37.1320	17.8381	110.9622	21.4039	
Maximum	42.6443	55.1510	46.4150	45.1905	114.1291	84.0481	

Table 5.18: SA Percentage Error Results for the Transaction Cost Cardinality Constrained Efficient Frontier

Percentage Error Difference	Hang Seng $N = 31$	DAX 100 $N = 85$	FTSE 100 $N = 89$	S&P 100 $N = 98$	Nikkei 225 $N = 225$	Average all markets
Mean	30.8106	24.5460	37.0907	20.3801	94.6561	41.4967
Median	31.9440	27.4525	39.3174	19.9579	94.5934	42.6630

Table 5.19: SA Percentage Error Difference for the Transaction Cost Cardinality Constrained Frontiers

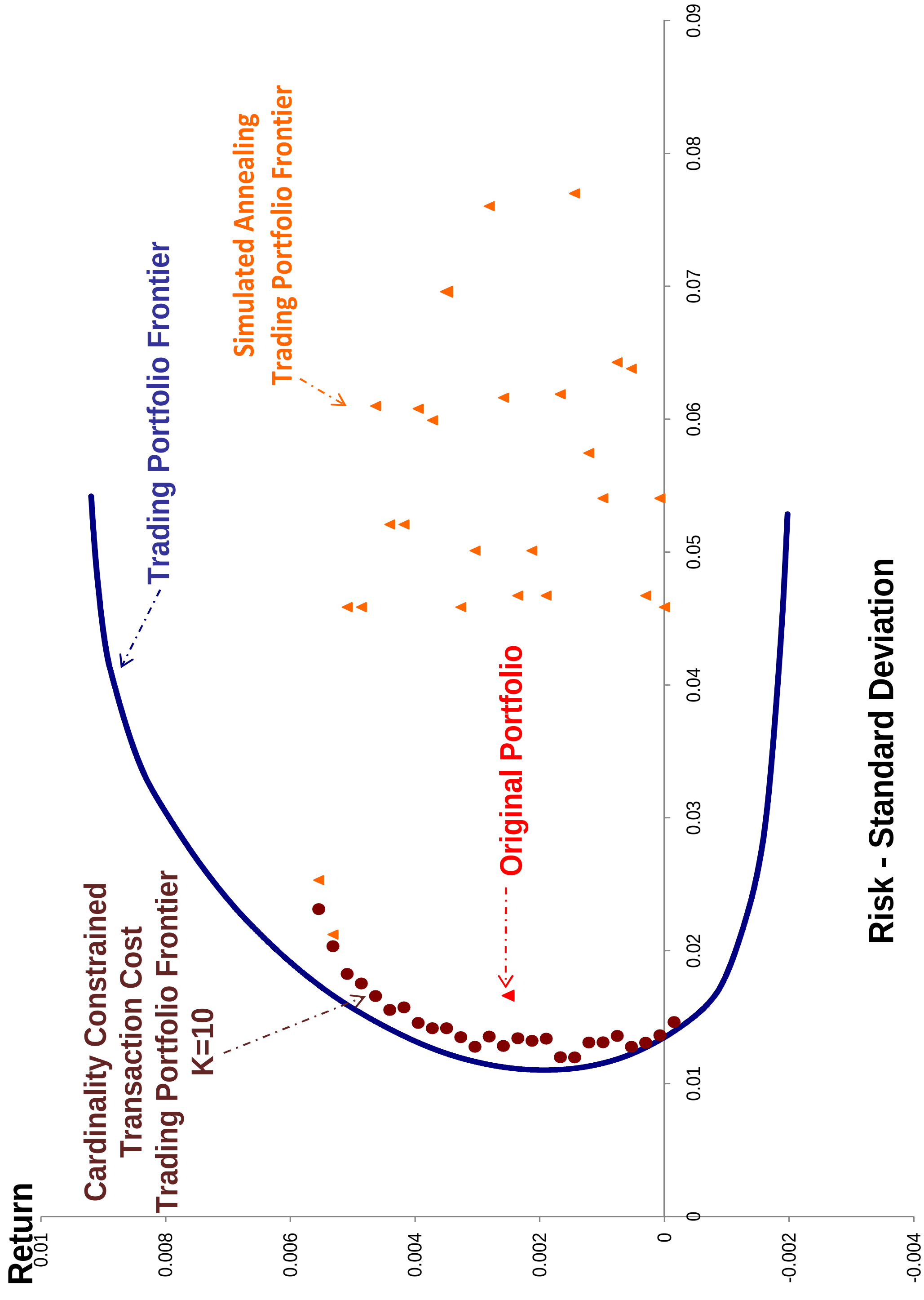

Figure 5.31: S&P 100 SA Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Frontier

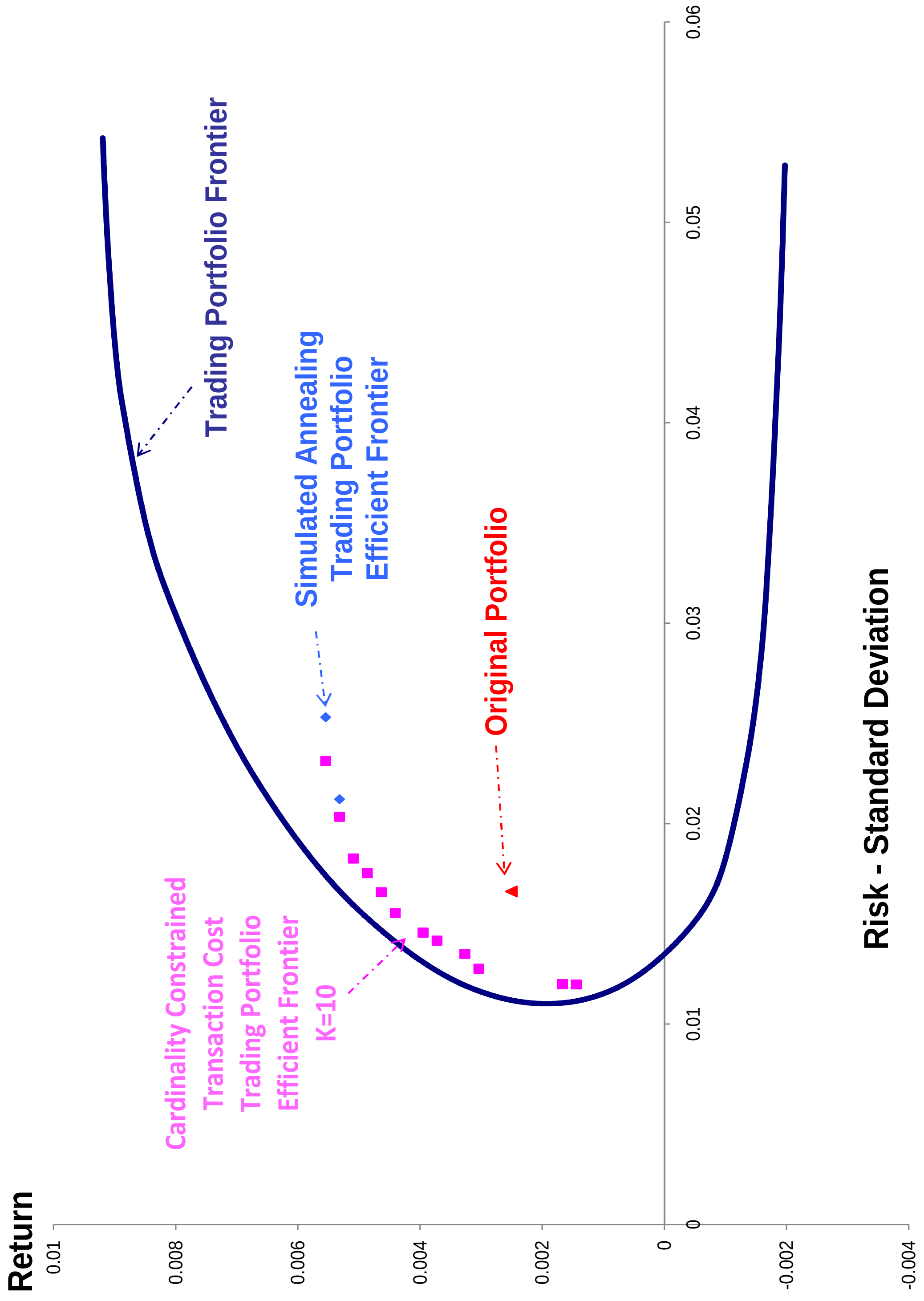

Figure 5.32: S&P 100 SA Heuristic Algorithm Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

the Nikkei 225 from the Table we see that it has a mean and median percentage error of 23.1385%. The total computational time to compute these results was 1566 seconds which is approximately 31 seconds per return level. Then, the mean and median percentage error differences for the Nikkei 225 are given as 5.6245% and 6.3130% respectively.

In Figures 5.33 to 5.42 we graphically represent our pooled heuristic algorithm results for the cardinality constrained transaction cost problem. The pictures illustrate the original portfolio, the cardinality constrained transaction cost trading portfolio frontier (then the cardinality constrained transaction cost efficient frontier) along with the portfolio (then the efficient) frontier for the Hang Seng, DAX 100, FTSE 100, S&P 100 and Nikkei 225 market indices respectively. Each Figure shows that the pooled heuristic algorithms provided some solutions which are close approximations to the actual cardinality constrained transaction cost frontier.

In Figures 5.43 to 5.44 we graphically represent our pooled heuristic algorithm results for the S&P 500 cardinality constrained transaction cost problem. In these pictures we illustrate the original portfolio, the efficeint frontier of the S&P 500 and the portfolio (then the efficient) frontier produced by pooling our heuristic algorithms. The S&P 500 is the only market index which we were unable to produce the cardinality constrained transaction cost frontier. Still, we present these pictures because they are of interest.

5.7 Summary

In this Chapter we presented transaction cost: heuristic algorithms. We began in Section 5.2 by presenting the subset optimisation model for the cardinality and non-cardinality constrained case of the problem. Also, in Section 5.2, we stated our heuristic algorithm implementation of genetic algorithm, tabu search and simulated annealing was the same as that used for the CCEF in Chapter 4. This was followed by data sets (Section 5.4) for our heuristic algorithms on six test problems considered previously in Chapter 4.

Within the non-cardinality and cardinality constrained computational results we showed the computational times, presented mean and median percentage error and percentage er-

Market Index	N	Percentage Error and Time (seconds)	Pooled Heuristic Algorithms							
			GA,TS,SA	Percentage error difference	GA,TS	Percentage error difference	GA,SA	Percentage error difference	TS,SA	Percentage error difference
Hang Seng	31	Mean	12.2838	1.2970	12.2838	1.2970	12.8585	1.8717	17.8684	6.8816
		Median	9.7135	0.1399	9.7135	0.1399	10.1156	0.2622	13.0756	3.2222
		Time (total)	300		244		180		176	
		Time (per return)	6		5		4		4	
DAX 100	85	Mean	11.7975	0.2992	11.7975	0.2922	11.7975	0.2922	25.4758	13.9775
		Median	15.4436	5.5902	15.4436	5.5902	15.4436	5.5902	25.4758	14.2265
		Time (total)	856		711		549		452	
		Time (per return)	17		14		11		9	
FTSE 100	89	Mean	11.1517	5.6953	11.1517	5.6953	14.7586	9.3022	9.2159	3.7595
		Median	12.1323	7.3555	12.1323	7.3555	13.6133	8.8365	8.3570	3.5802
		Time (total)	1382		1153		799		812	
		Time (per return)	28		23		16		16	
S&P 100	98	Mean	16.4095	5.2753	16.4095	5.2753	21.9592	10.8250	17.3279	6.1937
		Median	15.0168	3.4604	15.0168	3.4604	24.2900	12.7336	15.8803	4.3239
		Time (total)	1484		1224		873		871	
		Time (per return)	30		24		17		17	
Nikkei 225	225	Mean	22.7180	5.2040	22.7180	5.2040	23.1082	5.5942	23.1385	5.6245
		Median	22.9695	6.1440	22.9695	6.1440	23.0492	6.2237	23.1385	6.3130
		Time (total)	2726		2282		1604		1566	
		Time (per return)	55		46		32		31	
S&P 500	457	Mean	28.2263		20.1937		47.6423		28.2136	
		Median	20.4750		19.1133		36.7595		20.4822	
		Time (total)	14532		12940		8947		7177	
		Time (per return)	291		259		179		144	
Average (all markets)		Mean	17.0978		15.7591		22.0207		20.2067	
		Median	15.9584		15.7342		205452		17.7349	
		Time (total)	3547		3093		2159		1842	
		Time (per return)	71		62		43		37	

Table 5.20: Pooled Heuristic Algorithms' Results for the Transaction Cost Cardinality Constrained Frontiers

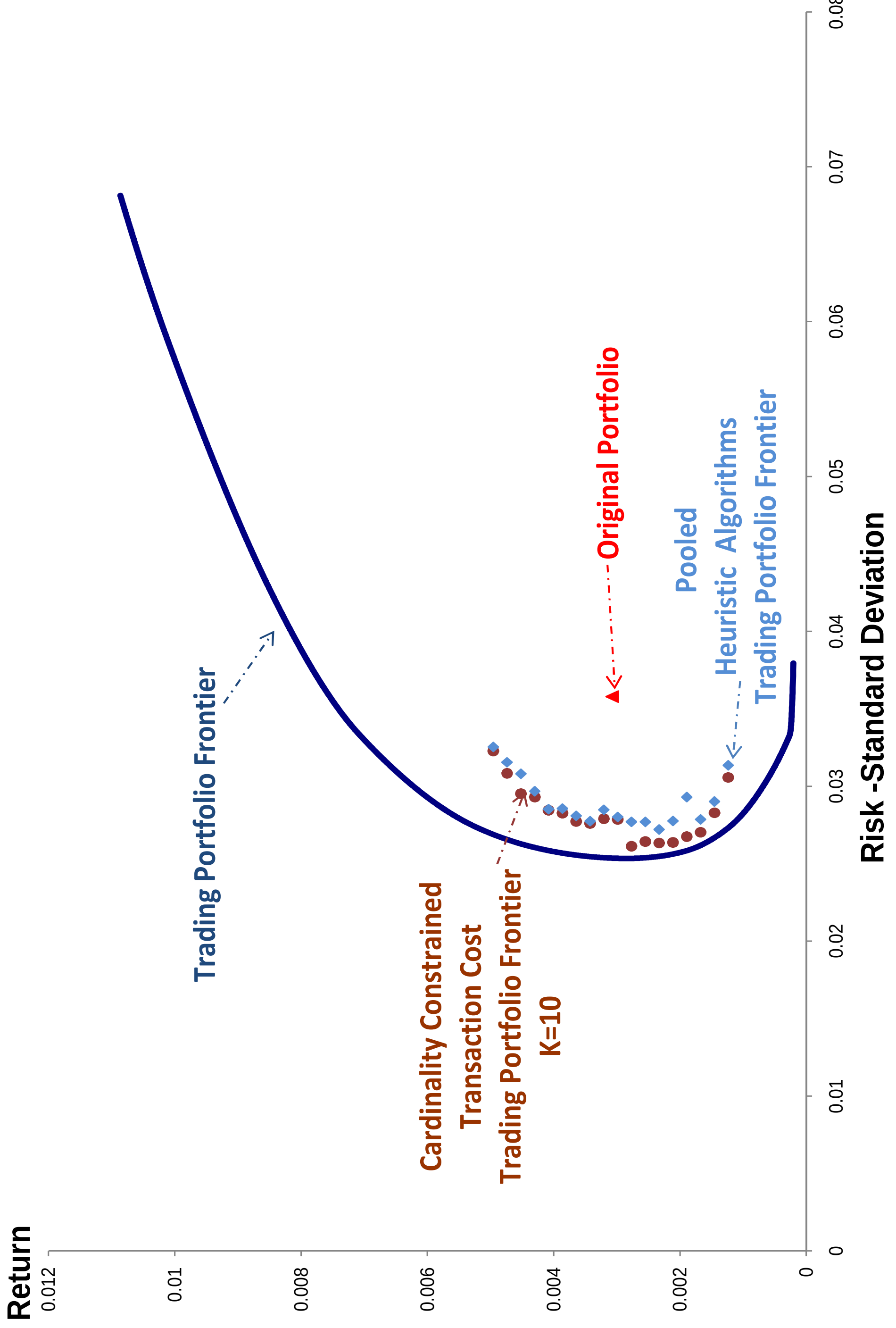

Figure 5.33: Hang Seng Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Frontier

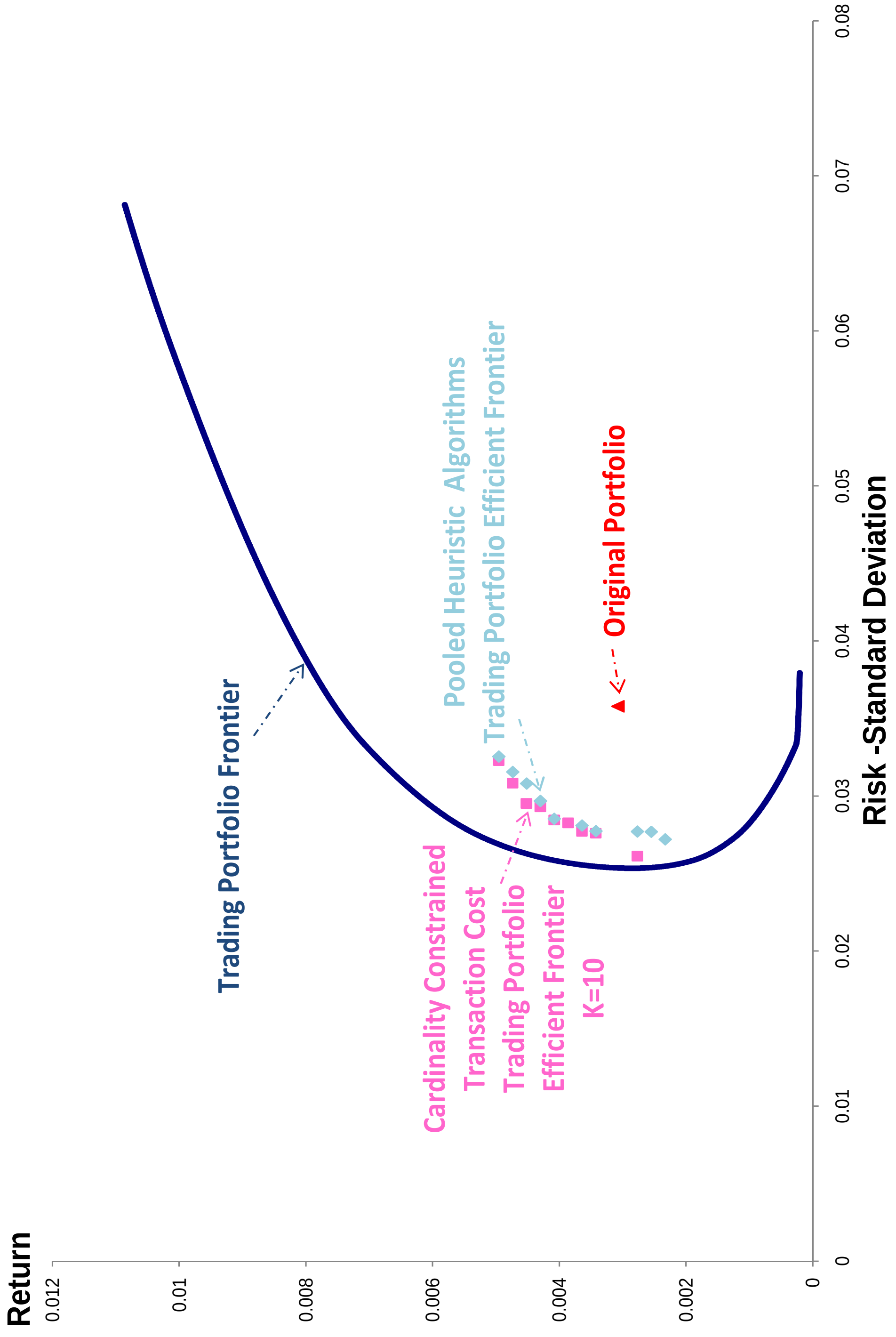

Figure 5.34: Hang Seng Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

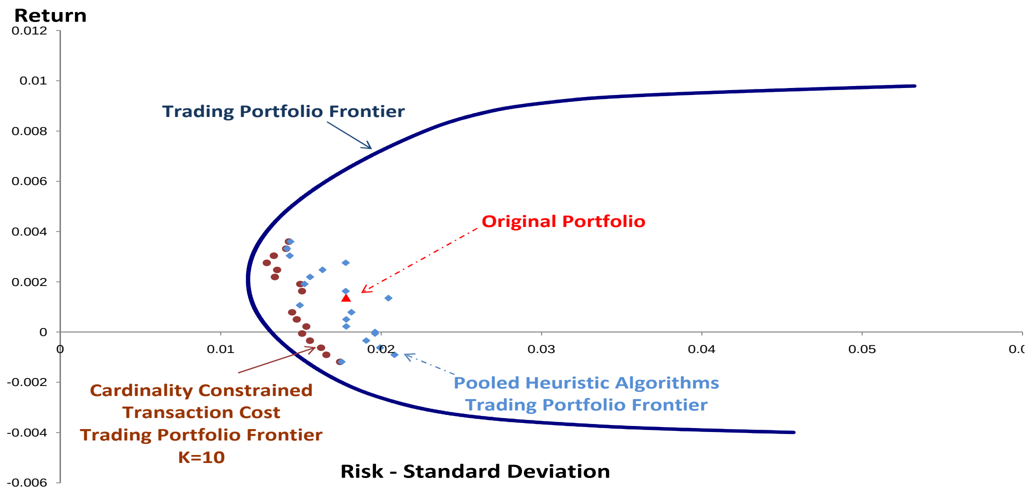

Figure 5.35: DAX 100 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Frontier

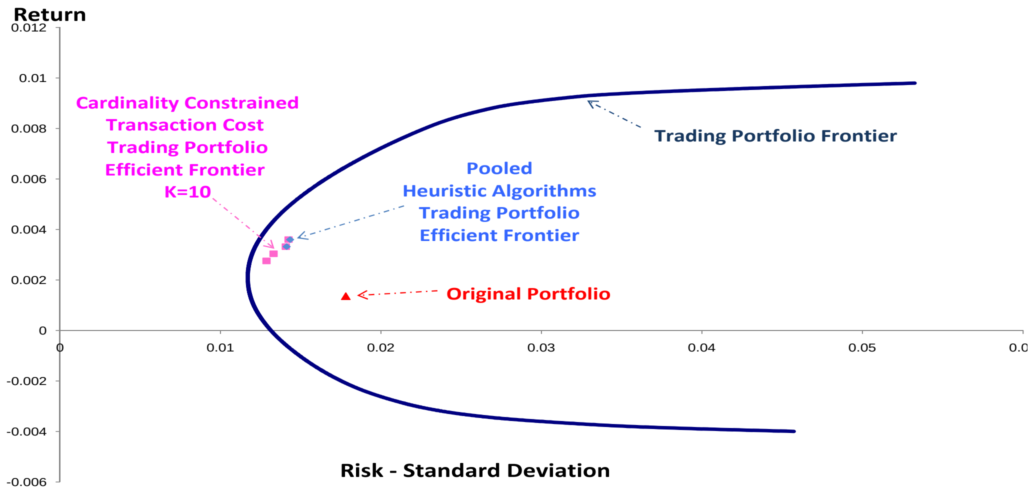

Figure 5.36: DAX 100 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

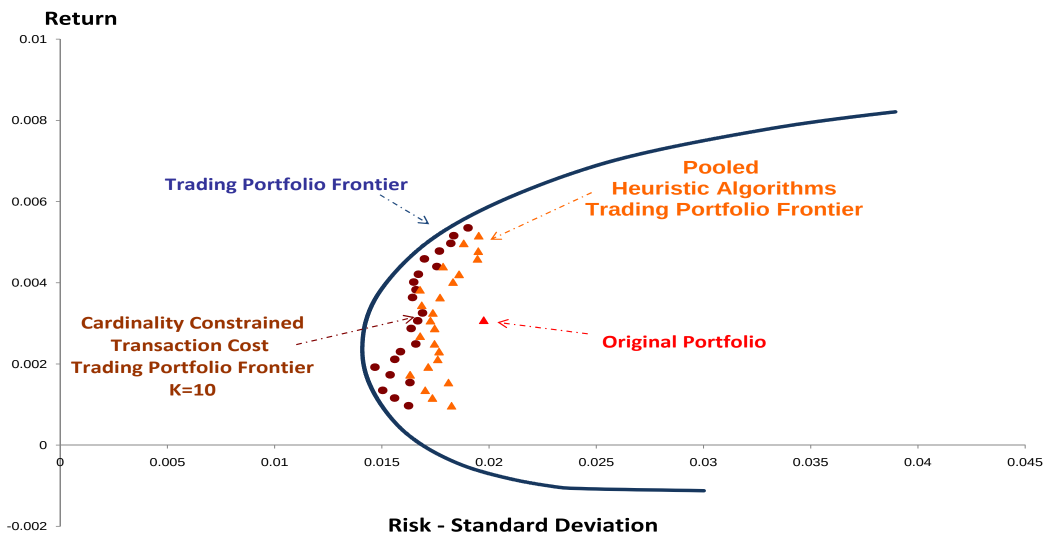

Figure 5.37: FTSE 100 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Frontier

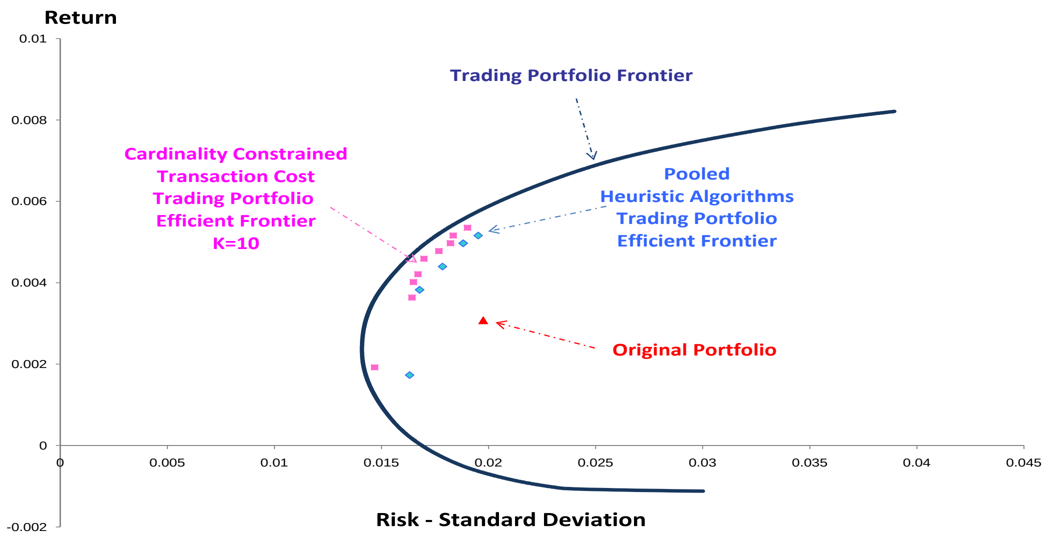

Figure 5.38: FTSE 100 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

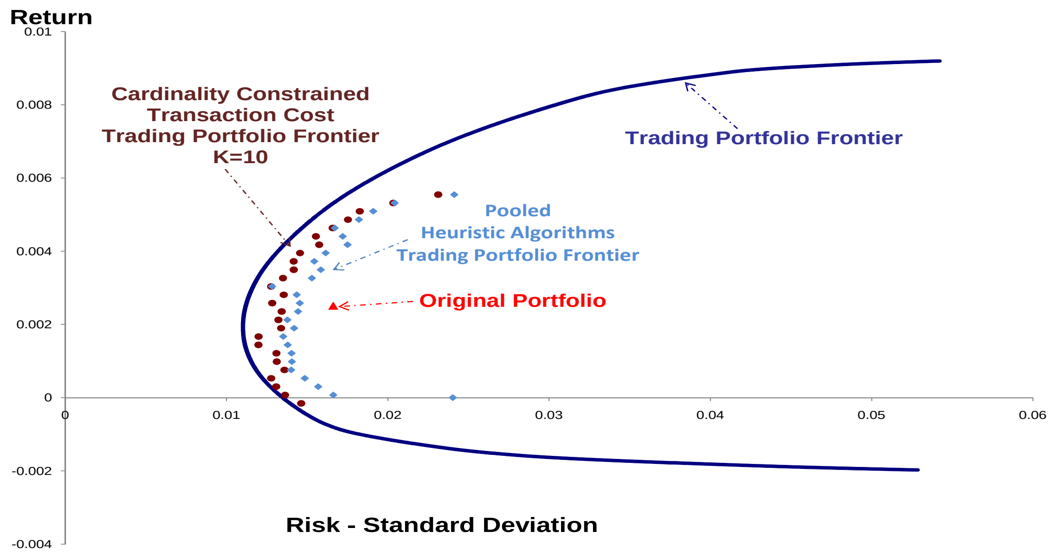

Figure 5.39: S&P 100 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Frontier

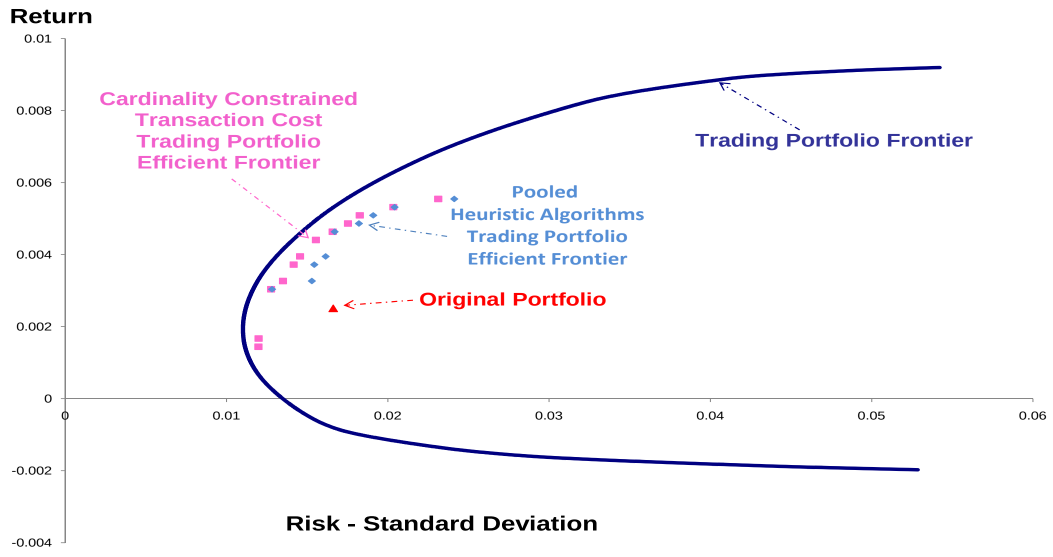

Figure 5.40: S&P 100 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

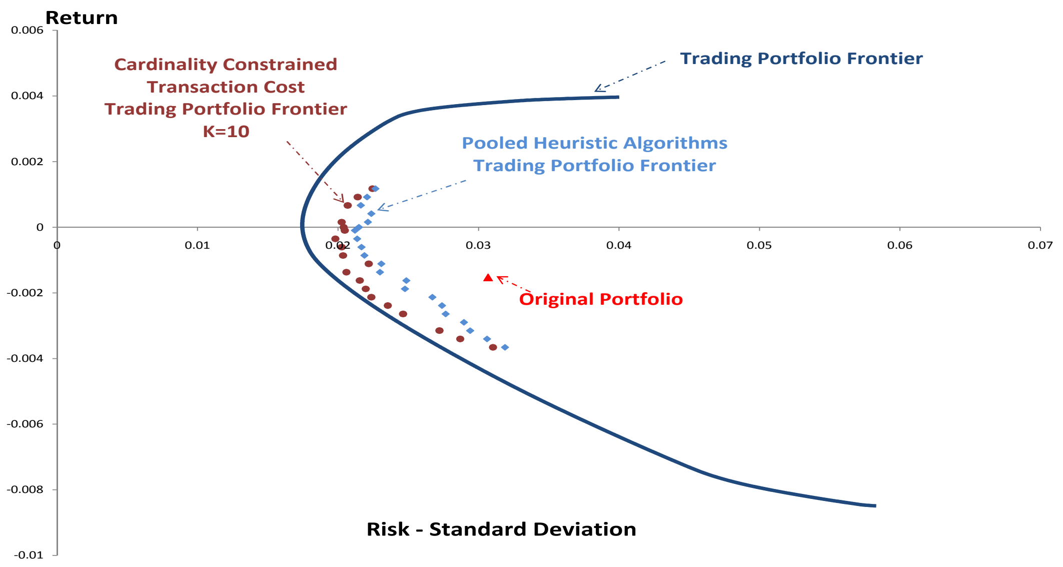

Figure 5.41: Nikkei 225 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Frontier

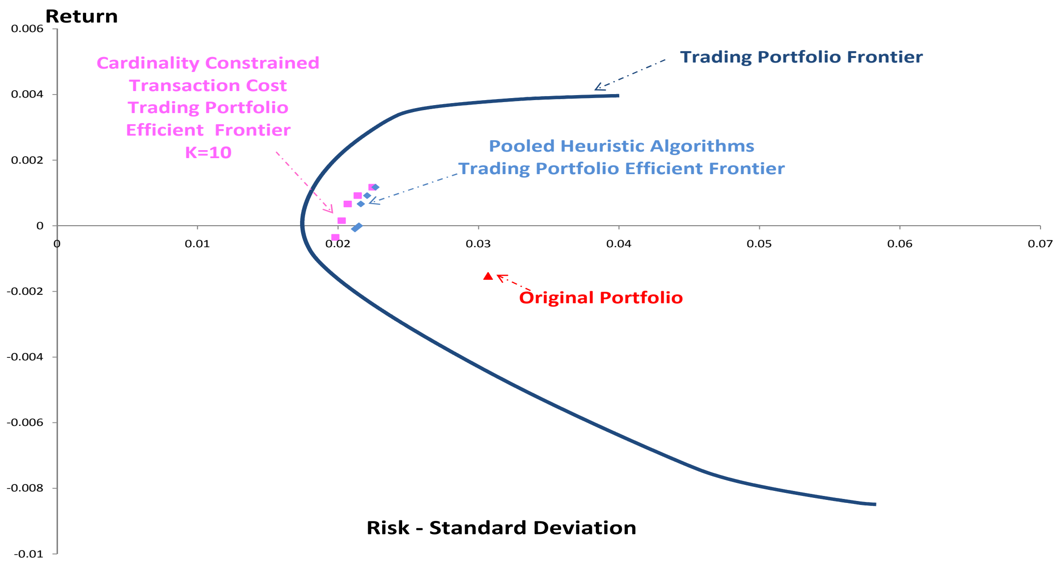

Figure 5.42: Nikkei 225 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

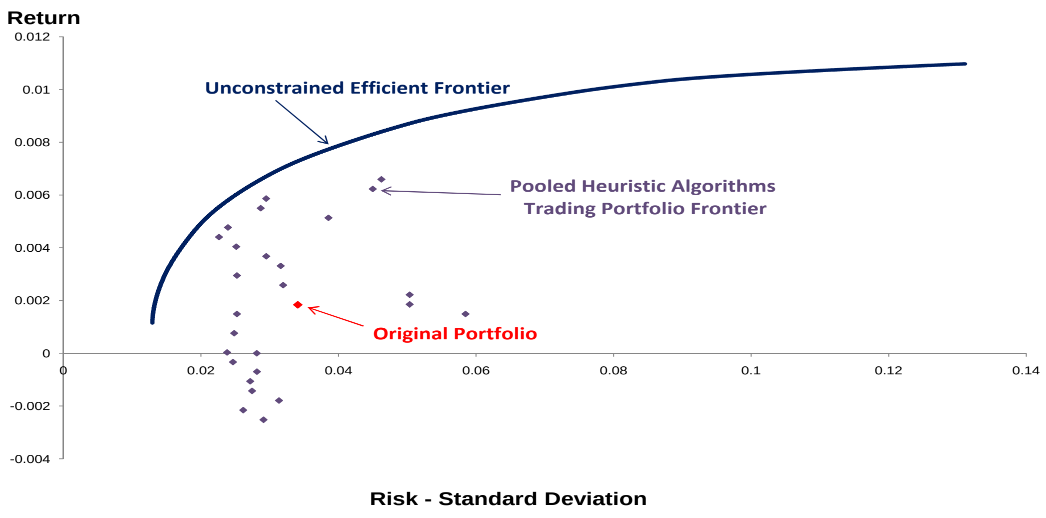

Figure 5.43: S&P 500 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Frontier

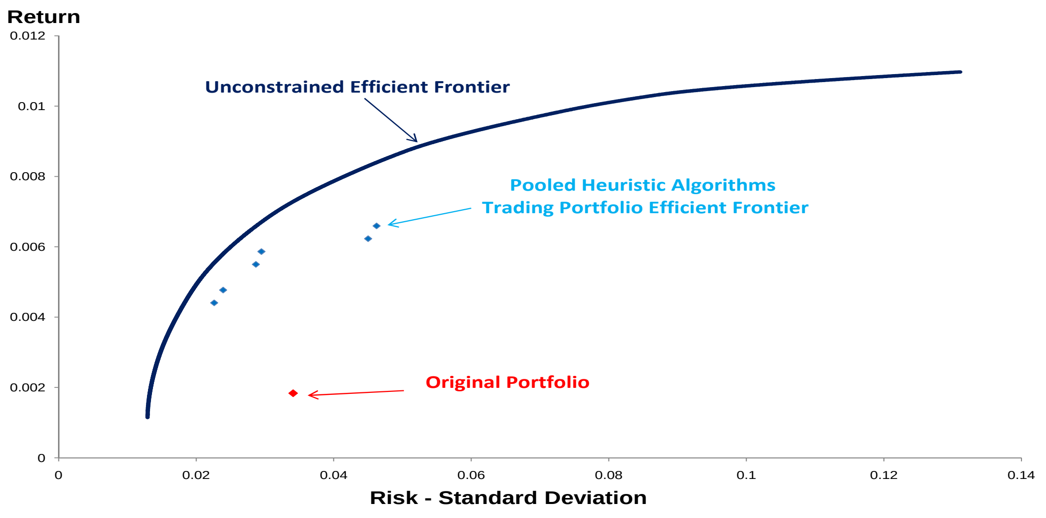

Figure 5.44: S&P 500 Pooled Heuristic Algorithms Transaction Cost Cardinality Constrained Trading Portfolio Efficient Frontier

ror difference for each of the heuristic algorithms and for the pooled heuristic algorithms. Additionally, we gave illustrative results.

We concluded that our GA, TS and pooled GA and TS heuristic algorithms were able to offer many optimal and near optimal solutions for both the cardinality and non-cardinality constrained case of the optimisation problem. SA was unable to produce near optimal solutions and only came into play when pooling for the largest market index.

We showed that when using the CPLEX solver optimal solutions were quicker than when using heuristic algorithms for the non-cardinality constrained transaction cost portfolio problem. Then, when we considered the cardinality constrained transaction cost portfolio problem solutions were obtained quicker using heuristic algorithms than using the CPLEX solver for the optimal solution. Thus, when considering only computational times the cardinality constrained heuristic algorithm technique requires significantly less time than finding the optimal solution and for the non-cardinality case finding optimal solutions required less time than our heuristic algorithms.

Chapter 6

Conclusion

6.1 Summary

The aim of this thesis was to contribute to the development of efficient and effective portfolio selection algorithms and their applications to portfolio optimisation problems involving cardinality constraints and transaction cost. We have presented quadratic mixed integer programming formulations for portfolio optimisation problems involving cardinality constraints and transaction cost. For these problems, we have created heuristic algorithms and proposed the subset optimisation problem as a component in the heuristic algorithm solution procedures.

In Chapter 2 we gave a review of previous studies on portfolio optimisation involving cardinality constraints and transaction cost. Work on cardinality constraints and heuristic algorithms principally stems from the work of Chang *et al.* [13] with the majority of the research using the same data set (as Chang *et al.* [13]) and focusing only on one heuristic algorithm. For the work on transaction cost the current work seems disjointed. There is no single underlying mathematical model, researchers tend only to solve problems with few assets often not detailing computational results and times, therefore making it impossible to perform a systematic comparison of the different approaches for the same data set.

Chapter 3, where we presented optimal solutions for the transaction cost model, contains the first original work in this thesis. We began by giving our model for the portfolio optimisation problem involving transaction cost, then we extended the model to include cardinality constraints. For both of these models we highlighted their benefits. We investigated the shape of the transaction cost trading portfolio and efficient frontiers and concluded that discontinuities occurred in both frontiers when there was only fixed transaction cost or there was cash invested in the portfolio. We further considered the non-cardinality and cardinality constrained trading portfolio and efficient frontiers for the portfolio optimisation problem involving transaction cost representing original work from this thesis not seen before in the literature. We presented graphical illustrations for the frontiers, gave computational times and compared the models (non-cardinality and cardinality) in terms of time and percentage error from the unconstrained efficient frontier. Then, from comparing the two transaction cost models we found that the non-cardinality constrained model produced quicker solution times. Furthermore, the efficient frontier produced by the non-cardinality constrained transaction cost model had a smaller mean as well as median percentage error from the unconstrained efficient frontier than the cardinality constrained transaction cost model.

In Chapters 4 and 5 we applied a genetic algorithm, tabu search and simulated annealing heuristic algorithm to the portfolio optimisation problem involving cardinality constraints and transaction cost which represents yet another original piece of work in this thesis. For each model (cardinality constrained, non-cardinality constrained transaction cost and cardinality constrained transaction cost) we developed a subset optimisation problem to solve the model. For portfolio optimisation problems subject to cardinality constraints and transaction cost with cardinality constraints, we deduced that our heuristic algorithms of genetic algorithm, tabu search and some of the pooled heuristics (mainly genetic algorithm and tabu search) provided an efficient and effective way of calculating the efficient frontier (for a particular problem) offering solutions with generally small percentage errors in a reasonable time. For portfolio optimisation problems subject to transaction cost with non-cardinality constraints, we concluded that despite heuristic algorithms offering solutions with relatively low percentage errors, it was not an efficient way to calculate the efficient frontier because the optimal efficient frontier could be calculated in less time.

In Chapter 4 we compared our results to those of Chang *et al.* [13] and extended our results to larger data sets, showing that our results offered a good quality of solution with no market index taking more than fifteen minutes. In Chapter 5 we compared our heuristic algorithms' results for the non-cardinality and cardinality constrained transaction cost optimisation problem. The computational results showed that the cardinality constrained transaction cost model produced results closer to it's unconstrained efficient frontier than the the non-cardinality constrained transaction cost model.

6.2 Contribution to Knowledge

Our literature review (in Chapter 2) demonstrated that we have read, and are familiar with, the relevant scientific literature with regard to portfolio theory. The mathematical and computational work presented in Chapters 3 and 5 as well as the subset optimisation model and the heuristic algorithms' psuedocode of Chapter 4 are, to the best of our knowledge, an original contribution to knowledge. That is these models presented, and enhancements suggested, have not been presented elsewhere in the literature.

In Chapter 3 our contribution has been

- the development of a clear mathematical model for the transaction cost problem,
- to present the transaction cost efficient frontier,
- to show the discontinuous nature of the transaction cost efficient frontier,
- to show that discontinuities occurred in the transaction cost frontier when subject to only fixed costs or cash investments, and
- to illustrate graphically the shapes of the cardinality and non-cardinality constrained transaction cost trading portfolio and efficient frontiers.

In Chapter 4 our contribution has been

- the development of the cardinality constrained subset optimisation problem,
- the development of effective heuristic algorithms based upon genetic algorithm, tabu search and simulated annealing,

- to consider larger test problems than considered previously in the literature, and
- the successful application of heuristic algorithms to computational problems for portfolio optimisation with cardinality constraints.

A paper based on Chapter 4 has been accepted for publication in the European Journal of Operational Research.

In Chapter 5 our contribution has been

- the development of the subset optimisation problems for the non-cardinality and cardinality constrained transaction cost models, and
- the successful application of heuristic algorithms to computational problems for portfolio optimisation involving transaction cost (non-cardinality and cardinality constrained).
- to show that for the non-cardinality constrained transaction cost portfolio optimisation problems, optimal solutions can be obtained quicker using a solver than when using heuristic algorithms, and
- to show that heuristic algorithms are better suited for cardinality constrained transaction cost portfolio optimistation problems than the commercial solver CPLEX.

6.3 Future Work

In this thesis, we have presented work for portfolio optimisation problems with the transaction cost and cardinality constraints. However, there are a number of areas that can be further researched some of which we touched upon in this thesis. These include:

- computationally applying restrictions on the amount of assets sold, bought and traded,
- class or sector constraints which specify minimum or maximum exposure to certain sectors (sets of assets), and

- lot size constraints which specify that the amount invested in any asset must be an integer multiplier of a known constant.

These extensions can all be applied to the transaction cost models (from Chapters 3 and 5) and the latter two extensions can be applied to the cardinality constrained model (from Chapter 4). These further research areas can be formulated in a linear manner by adding extra variables and/or constraints to the models we have presented. Thus, using standard solution packages such as CPLEX to determine the optimal solution is a valid approach.

There are also a number of enhancements that could be considered in the heuristic algorithms. These include:

- for the GA not allowing any duplicate solutions into the population,
- for the TS varying the tabu tenure
- applying variable neighbourhood search to the TS heuristic algorithm, and
- the creation of a hybrid of two of the heuristic algorithms such as GA and TS.

Chapter 7

Appendix

Asset	P_i	X_i	L_i^s	L_i^b	U_i^s	U_i^b	f_i^b	f_i^s	c_i^s	c_i^b
1	61	188	19	19	188	94	579	610	0.305	0.305
2	40	180	18	18	180	90	596	585	0.2	0.2
3	29	176	18	18	176	88	590	538	0.145	0.145
4	27	142	14	14	142	71	594	553	0.135	0.135
5	60	191	19	19	191	96	550	642	0.3	0.3
6	70	170	17	17	170	85	640	551	0.35	0.35
7	69	165	17	17	165	83	537	555	0.345	0.345
8	75	200	20	20	200	100	640	612	0.375	0.375
9	36	117	12	12	117	59	629	578	0.18	0.18
10	63	170	17	17	170	85	557	656	0.315	0.315
11	53	122	12	12	122	61	549	564	0.265	0.265
12	32	102	10	10	102	51	560	589	0.16	0.16
13	64	200	20	20	200	100	635	575	0.32	0.32
14	31	179	18	18	179	90	564	592	0.155	0.155
15	60	197	20	20	197	99	550	577	0.3	0.3
16	45	159	16	16	159	80	628	620	0.225	0.225
17	67	150	15	15	150	75	564	568	0.335	0.335
18	55	148	15	15	148	74	641	559	0.275	0.275
19	63	191	19	19	191	96	633	615	0.315	0.315
20	75	158	16	16	158	79	647	640	0.375	0.375
21	34	130	13	13	130	65	633	655	0.17	0.17
22	69	187	19	19	187	94	538	618	0.345	0.345
23	25	155	16	16	155	78	558	631	0.125	0.125
24	70	131	13	13	131	66	602	602	0.35	0.35
25	46	188	19	19	188	94	534	600	0.23	0.23
26	28	194	19	19	194	97	660	572	0.14	0.14
27	66	154	15	15	154	77	579	598	0.33	0.33
28	50	154	15	15	154	77	629	574	0.25	0.25
29	30	198	20	20	198	99	571	590	0.15	0.15
30	57	130	13	13	130	65	608	569	0.285	0.285
31	39	183	18	18	183	92	602	583	0.195	0.195

Table 7.1: Hang Seng parameter values

References

References

[1] Adcock C J and Meade N (1994). *A simple algorithm to incorporate transaction costs in quadratic optimisation*, European Journal of Operational Research, 79: 8594.

[2] Anagnostopoulos K P and Mamanis G, "A portfolio optimization model with three objectives and discrete variables," *Computers & Operations Research* 37 (2010) 1285-1297.

[3] Angelelli E, Mansini R and Speranza, "A comparison of MAD and CVaR models with real features," *Journal of Banking and Finance* 32 (2008) 1188–1197.

[4] Alander J T, "On optimal population size of genetic algorithm,"In Proceeding of CompEuro 92 (1992) 65–70, IEEE Computer Society Press.

[5] Beale E M L and Forest J J H, "Global optimization using special ordered sets," *Mathematical Programming* 10 (1976) 52–69.

[6] Beasley J E, "OR-Library: distributing test problems by electronic mail," *Journal of the Operational Research Society* 41 (1990) 1069-1072.

[7] Beasley J E, "Population heuristics,"in Pardalos P M and Resende M G C (eds.), Handbook of Applied Optimization Oxford University Press, Oxford, (2002) pp. 138–157.

[8] Bertsimas D and Shioda R, "Algorithm for cardinality-constrained quadratic optimization," *Computational Optimization and Applications* 43 (2009) 1-22.

[9] Baule R, "Optimal portfolio selection for the small investor considering risk and transaction costs," *O R Spectrum* 32 (2010) 61-76.

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

[11] Burke E K and Graham K (eds), "Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques,"Springer 2005.

[12] Černý V, "Thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm," *Journal of Optimization Theory and Applications* 45 (1985) 41-51.

[13] Chang T-J, Meade N, Beasley J E and Sharaiha Y M, "Heuristics for cardinality constrained portfolio optimisation," *Computers & Operations Research* 27 (2000) 1271-1302.

[14] Chen W and Cai Y M, "Study on the Efficient Frontier in Portfolio Selection by Using Particle Swarm Optimization,"2008 Chinese Control and Decision Conference, 11 (2008) 269–272.

[15] Chen W, Ling Y and Fasheng X, "PSO-based Possibilistc Mean-Variance Model with Transaction Costs,"CCDC 2009: 21st Chinese Control and Decision Conference, 1-6 (2009) 5993–5997.

[16] Corazza M and Favaretto D, "On the existence of solutions to the quadratic mixed-integer mean-variance portfolio selection problem," *European Journal of Operational Research* 176 (2007) 1947–1960.

[17] Crama Y and Schyns M, "Simulated annealing for complex portfolio selection problems," *European Journal of Operational Research* 150 (2003) 546-571.

[18] Cura T, "Particle swarm optimization approach to portfolio optimization," *Nonlinear Analysis: Real World Applications* 10 (2009) 2396-2406.

[19] Derigs U and Nickel N-H, "Meta-heuristic based decision support for portfolio optimisation with a case study on tracking error minimization in passive portfolio management," *OR Spectrum* 25 (2003) 345-378.

[20] Derigs U and Nickel N-H, "On a local-search heuristic for a class of tracking error minimization problems in portfolio management," *Annals of Operation Research* 131 (2004) 45-77.

[21] Dongarra J J, "Performance of various computers using standard linear equations software,"Report CS-89-85, University of Tennessee, USA. Available from http://www.netlib.org/benchmark/performance.ps last accessed February 7th 2010.

[22] Ehrgott M, Klamroth K and Schwehm C, "An MCDM approach to portfolio optimization," *European Journal of Operational Research* 155 (2004) 752-770.

[23] Ellison E F D, Hajian M, Levkovitz K, Maros I and Mitra G, "A Fortran based mathematical programming system, FortMP,"Brunel University, UK and NAG Ltd., Oxford, UK 1999.

[24] Fernandez A and Gomez S, "Portfolio selection using neural networks," *Computers & Operations Research* 34 (2007) 1177-1191.

[25] Fieldsend J Em Matatko J and Peng M, "Cardinality constrained portfolio optimisation,"Intelligent Data Engineering and Automated Learning Ideal 2004, Proceedings, Lecture notes in Computer Science 3177 (2004) 788–793.

[26] Fourer F, Gay D M and Kernighan B W, "AMPL: A Modeling Language for Mathematical Programming,"Brooks/Cole Publishing Company / Cengage Learning 2002.

[27] Goldberg D E, "Genetic Algorithms in Search, Optimization, and Machine Learning,"Addison–Wesley, Reading, Massachusetts, USA 1989.

[28] Glover F and Kochenberger G A , "Handbook of Metaheuristics,"Kluwer Academic Publishers 2003.

[29] Glover F and Laguna M, Tabu search, in "Modern Heuristic Techniques for Combinatorial Problems,"Reeves CR (ed.), Blackwell Scientific Publications, Oxford, UK (1993) pp 70-150.

[30] Glover F and Laguna M, "Tabu Search,"Kluwer Academic Publishers, Dordrecht, The Netherlands 1997.

[31] Gulipinar N, Le THA H A and Moeini M, "Robust investment strategies with discrete asset choice constraints using DC programming," *Optimization*, 59 (2010) 45–62.

[32] Holland J H, "Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence,"University of Michigan Press, Ann Arbor, MI, USA 1975.

[33] Kellerer H, Mansini R and Speranza M G, "On selecting a portfolio with fixed costs and minimum transaction lots,"*Annals of Operation Research* 99 (2000) 287–304.

[34] Kirkpatrick S, Gelatt C D and Vecchi M P, "Optimization by simulated annealing,"*Science* 220 (1983) 671-680.

[35] Konno H and Wijayanayake A, "Mean-absolute deviation portfolio optimization model under transaction costs,"*Journal of the Operations Research Society of Japan* 42 (1999) 422-435.

[36] Konno H and Yamazaki H, "Mean-absolute deviation portfolio optimization model and its applications to Tokyo stock market,"*Management Science* 37 (1991) 519-531.

[37] Konno H, Shirakawa H and Yamazaki H, "A mean-absolute deviation-skewness portfolio optimization model,"*Annals of Operations Research* 45 (1993) 205-220.

[38] Lemke C E and Howson J T, "Equilibrium points of bimatrix games,"*Journal of the Society for Industrial and Applied Mathematics* 12 (1964) 413-423.

[39] Lee E K and Mitchell J E, "Computational experience of an interior-point SQP algorithm in a parallel branch-and-bound framework,"Proc. High Performance Optimization Techniques, Springer, Berlin 1997.

[40] Lee S M and Chesser D L, "Goal programming for portfolio selection," *Journal of Portfolio Management* 7 (1980) 23-25.

[41] Li D, Sun X and Wang J, "Optimal lot solution to cardinality constrained meanvariance formulation for portfolio selection,"*Mathematical Finance* 16 (2006) 83-101.

[42] Li Z, Wang S, and Deng X, "A linear programming algorithm for optimal portfolio selection with transaction costs,"*International Journal of Systems Science* 31 (2000) 107-117.

[43] Lintner J, “The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets” *The Review of Economics and Statistics* 47 (1965) 1337.

[44] Lynch A W and Balduzzi P, “Predictability and transaction costs: The impact on rebalancing rules and behavior,” *Journal of Finance* 55 (2000) 2285-2309.

[45] Mansini R and Speranza M G, “Heuristic algorithms for the portfolio selection problem with minimum transaction lots,” *European Journal of Operational Research* 114 (1999) 219-233.

[46] Maringer D, “Portfolio Management with Heuristic Optimization,” Springer, The Netherlands 2005.

[47] Maringer D and Kellerer H, “Optimization of cardinality constrained portfolios with a hybrid local search algorithm,” *OR Spectrum* 25 (2003) 481-495.

[48] Markowitz H, “Portfolio selection,” *Journal of Finance* 7 (1952) 77-91.

[49] Markowitz H M, “The optimization of a quadratic function subject to linear constraints,” *Naval Research Logistics Quarterly* 3 (1956) 111-133.

[50] Michalewicz Z, “Genetic Algorithms + Data Structures = Evolution Programs” (3rd edition), Springer–Verlag, Berlin 1996.

[51] Moral-Escudero R, Ruiz-Torrubiano R and Suarez A, “Selection of optimal investment portfolios with cardinality constraints,” in Proceedings of the 2006 IEEE Congress on Evolutionary Computation, (2006) 2382-2388.

[52] Mossin J, “Equilibrium in a capital asset market,” *Econometrica,* 34 (1966) 768-783.

[53] Mulvey J M, “Incorporating transaction costs in models for asset allocation,” *Financial Optimization, Cambridge University Press* (1993) 243-259.

[54] Pai G A V and Michel T, “Evolutionary optimization of constrained k-means clustered assets for diversification in small portfolios,” IEEE Transactions on Evolutionary Computation, 13 (2009) 1030-1053.

[55] Perold A F, "Large Scale Portfolio Optimization," *Management Science* 30 (1984) 1143–1160.

[56] Roman D, Darby-Dowman K and Mitra G "Mean-Risk Models Using Two Risk Measures: A Multi-Objective Approach," *Quantitative Finance* 4 (2007) 443-458.

[57] Reeves C R, "Modern heuristic techniques, in: V J Raynard-Smith, I H Osman, C R Reeves and G D Smith, Modern Heuristic Search Methods,"Wiley, New York (1996) 1–25.

[58] Reilly F K and Brown K C, "Investment Analysis and Portfolio Management,"7th ed Thomson Learning, Thomson 2003.

[59] Rosenberg B, "Extra-market components of covariance in security returns," *J. Financial Quant. Anal.* 9 (1974) 263-273.

[60] Ross S A, "The arbitrage theory of capital asset pricing," *Journal of Economic Theory* 13 (1976) 34160.

[61] Sharpe W F, "A simplified model for portfolio analysis," *Management Science* 9 (1963) 277–293.

[62] Sharpe W F, "Capital asset prices: a theory of market equilibrium under conditions of risk," *Journal of Finance* 19 (1964) 425–442.

[63] Sharpe W F, "A linear programming approximation for the general portfolio analysis problem," *J. Financial Quant. Anal.* 6 (1971) 1263-1275.

[64] Shaw D X, Liu S and Kopman L, "Lagrangian relaxation procedure for cardinality-constrained portfolio optimization," *Optimization Methods & Software* 23 (2008) 411-420.

[65] Soleimani H, Golmakani H R and Salimi M H, "Markowitz-based portfolio selection with minimum transaction lots, cardinality constraints and regarding sector capitalization using genetic algorithm," *Expert Systems with Applications* 36 (2009) 5058-5063.

[66] Speranza M G, "A heuristic algorithm for a portfolio optimization model applied to the Milan stock market," *Computers & Operations Research* 23 (1996) 433–441.

[67] Stein M, Branke J, and Schmeck H, "A heuristic algorithm for a portfolio optimization model applied to the Milan stock market," *Computers & Operations Research* 35 (2008) 3945–3961 .

[68] Streichert F and Tanaka-Yamawaki M, "The effect of local search on the constrained portfolio selection problem,"in Proceedings of the 2006 IEEE Congress on Evolutionary Computation, (2006) 2368-2374.

[69] Tamiz M, Hasham D F, Jones B H and Fargher E K, "A Two-staged goal programming model for portfolio selection,"Multi-Objective Programming and Goal Programming: Theories and Application, ed. By M. Tamiz, Springer-Verlag, Berlin (1996) 286–299.

[70] Worzel K, Vassiadou-Zeniou C and Zenios S A, "Integrated simulation and optimization models for tracking fixed-income indices," *Operations Res.* 42 (1994) 223-233.

[71] Xia Y, Wang S, and Deng X, "A compromise solution to mutual funds portfolio selection with transaction costs," *European Journal of Operational Research* 134 (2001) 564-581.

[72] Xue H, Xu C, and Feng Z, "Meanvariance portfolio optimal problem under concave transaction cost," *Applied Mathematics and Computation* 174 (2006) 1-12

[73] Yoshimoto A, "The mean-variance approach to portfolio optimization subject to transaction costs," *Journal of the Operations Research Society of Japan* 39 (1996) 99117.

[74] Young M R, "A minimax portfolio selection rule with linear programming solution," *Management Science* 44 (1998) 673-683.