Portfolio Optimization: Scenario Generation, Models and Algorithms

by

Gianfranco Guastaroba

A thesis submitted for the degree of

Doctor of Philosophy

SUPERVISOR

Prof. M.Grazia Speranza

PHD IN COMPUTATIONAL METHODS FOR FORECASTING AND DECISIONS IN ECONOMICS AND FINANCE
UNIVERSITÁ DEGLI STUDI DI BERGAMO
DEPARTMENT OF MATHEMATICS, STATISTICS, COMPUTER SCIENCE AND APPLICATIONS

January 7, 2010

Author's e-mail: guastaro@eco.unibs.it

Author's address:

Department of Quantitative Methods
University of Brescia
C.da Santa Chiara, 50
25122 Brescia – Italy
tel. +39-30-2988599
fax. +39-30-2400925
web: http://www.unibs.it/on-line/dmq/Home/Personale/scheda785.html

Abstract

In single-period portfolio optimization several facets of the problem may influence the goodness of the portfolios selected. Despite that, some of these facets are frequently ignored when the optimization problem is solved. In this thesis, we aim at investigating the impact of these facets on the optimization problem and on the performances of the portfolios selected.
Firstly, we consider the problem of generating scenarios. In the domain of single-period portfolio optimization, scenarios are used to compute the expected value of the portfolio return and the value of a risk (or safety) measure. Historical data observations, taken as equally probable scenarios, are frequently used to this aim. However, several other parametric and non-parametric methods can be alternatively applied. Specifically, when dealing with *scenario generation techniques* practitioners are mainly concerned on how reliable and effective such methods are when embedded into portfolio selection models. To this aim, we survey different techniques to generate scenarios for the rates of return. We also compare these techniques by providing in-sample and out-of-sample analysis of the portfolios selected using these techniques to generate the rates of return. Evidence on the computational burden required to generate scenarios by the different techniques is also provided. As reference model we use the Conditional Value-at-Risk (CVaR) model with transaction costs. Extensive computational results based on different historical data sets from the London Stock Exchange Market (FTSE) are presented and some interesting financial conclusions are drawn.
Secondly, we analyze portfolio optimization when data uncertainty is taken into consideration. Data uncertainty is a common feature in most of real-life optimization problems. In deterministic mathematical optimization, it is assumed that all the input data are known with certainty and equal to some nominal values. Nevertheless, the optimal solution of the nominal problem can reveal itself sub-optimal or even infeasible when some of the data take values different from the nominal ones. An area where data uncertainty is a natural concern is portfolio optimization. As a matter of fact, in portfolio selection every optimization model deals with the estimate of the portfolio rate of return. In the literature several techniques that are immune to data uncertainty have been proposed. These techniques are called *robust*. We investigate the effectiveness of two well-known robust optimization techniques when applied to a specific portfolio selection problem, and compare the portfolios selected by the corresponding robust counterparts. As reference model we consider the portfolio optimization problem with the CVaR as performance measure. We carried out extensive computational experiments based on real-life data from the London Stock Exchange Market (FTSE) under different market behaviors.
Thirdly, we study the optimal portfolio selection problem in a *rebalancing framework*, considering fixed and proportional transaction costs and evaluating how much they affect a re-investment strategy. Specifically, we modify the single-period portfolio optimization model with transaction

costs, based on the CVaR as performance measure, to introduce portfolio rebalancing. The aim is to provide investors and financial institutions with an effective tool to better exploit new information made available by the market. We then suggest a procedure to use the proposed optimization model in a rebalancing framework. Extensive computational results based on different historical data sets from the German Stock Exchange Market (XETRA) are presented.
The last part of the thesis is devoted to the problem of replicating the performances of a stock market index, but considering transaction costs and without purchasing all the securities that constitute the index, i.e. the *index tracking* problem. Furthermore, we also consider the problem of out-performing a market index, i.e. the so-called enhanced index tracking problem. We present mixed-integer linear programming formulations of these two problems. Our formulations include both fixed and proportional transaction costs, a cardinality constraint limiting the number of securities that constitute the portfolio, upper and lower limits on the investment in the securities, and a threshold on the total transaction costs paid. Both models can be used to rebalance a current portfolio composition. We also introduce a heuristic framework, called Enhanced Kernel Search, to solve the problem of tracking an index. We test and analyze the behavior of several implementations of the Enhanced Kernel Search framework. We show the effectiveness and efficiency of the framework comparing the performances of the tested heuristics with those of a general-purpose solver. The computational experiments are carried out using benchmark instances for the index tracking problem.

Contents

List of Figures

List of Tables

1

Introduction

In single-period portfolio optimization literature, several assumptions are commonly ignored. Despite that, some of these assumptions can determine the effectiveness of the mathematical models as decision tools in supporting financial decision making.
First of all, the goodness of any mathematical model heavily depends on the quality of the input data. In the domain of portfolio optimization, most of the selection models require a set of scenarios, i.e. a realization of the multivariate random variable representing the rates of return of all the securities. Though in the literature several techniques for generating scenarios have been proposed, when solving a single-period portfolio optimization model the usual assumption is to take historical data as the only descriptor of future possible outcomes. Obviously, different other methods can be alternatively used to generate scenarios for rates of return. In this thesis we propose a comparison among some scenario generation techniques when embedded in single-period portfolio optimization. Among the techniques available in the literature, we selected three non-parametric and two parametric techniques that we feel are more promising than others when applied to the problem under examination. Specifically, we assume as reference model the single-period portfolio selection model with the Conditional Value-at-Risk as performance measure. We propose a mathematical formulation that considers the presence of both proportional and fixed transaction costs. The study aims at evaluating the effectiveness of the selected techniques at generating the input data for the reference optimization model. In our opinion, their effectiveness should be evaluated by means of out-of-sample analysis, particularly considering different data sets that span different market behaviors. To this aim, we carried out extensive computational experiments on real-life data from the London Stock Exchange. The selected data sets consider different in-sample and out-of-sample time periods reflecting different market trends. The effectiveness of the scenario generation techniques is then evaluated observing the performances yielded out-of-sample by the optimized portfolios.
Secondly, some of the input parameters of any portfolio optimization model are, by their nature, uncertain. Despite that, a common approach in single-period portfolio optimization is to assume that the input data are known deterministically and equal to some nominal value. However, this assumption ignores the influence of parameter uncertainty on the optimality and feasibility of the solutions. Therefore, when some of the parameters differ from the assumed nominal values, the nominal optimal solution may violate some constraints, i.e. the solution becomes infeasible, or may perform poorly, i.e. the solution becomes sub-optimal. The former observations motivated the need for methodologies that are immune to data uncertainty. Robust optimization is, indeed, one technique that addresses the problem of the presence of uncertain parameters in optimization models. The main advantage of most robust optimization techniques is that the only information

needed is the mean and the range of variability of each random parameter instead of, for instance, the exact distribution function required by stochastic optimization methodologies. Several papers appeared in the literature under the name of robust optimization. Nevertheless, most of them provide only theoretical contributions, with little or no empirical evidence about their effectiveness. We consider as nominal formulation the single-period portfolio optimization model that assumes the Conditional Value-at-Risk as performance measure. We then propose two robust formulations of the nominal problem according to two well-known robust optimization techniques. The main objective of this research is to study the impact of the selected robust optimization techniques on the portfolio selection problem under examination. To this end, we carried out extensive computational experiments on real-life data from the London Stock Exchange, under different in-sample and out-of-sample scenarios. The performances realized out-of-sample by the robust portfolios and the portfolio selected by the nominal model are then compared.
Thirdly, in single-period portfolio optimization problems, it is assumed that the investor selects the portfolio composition at a certain date and then maintains that composition unchanged over the entire investment horizon. This strategy is also referred to as buy-and-hold. Nevertheless, this investment strategy may fail due to sudden changes of the market trend from the in-sample to the out-of-sample period. In these cases, a better investment strategy is dynamic, i.e. a strategy that considers re-optimizing the portfolio composition at some time in order to take into consideration the new set of information provided by the market. A first issue that come up implementing a dynamic strategy is how frequently the portfolio composition should be rebalanced. A second issue is that the payment of transaction costs, both proportional and fixed, cannot be neglected implementing any dynamic strategy. We assume as reference model the single-period Conditional Value-at-Risk with transaction costs. We then modify the reference model in order to consider rebalancing the current portfolio composition. The two mathematical models allow, respectively, to implement a buy-and-hold and any dynamic strategy. The principal goal of this research is to compare different investment strategies characterized by different degrees of dynamism. To this end, we consider a static investor, that implements a buy-and-hold strategy, and several dynamic investors, that re-optimize the portfolios at different times and with different frequencies. Then, we carried out extensive computational experiments on real-life data from the German Stock Exchange testing several data sets to analyze different possible market trends. The performances of the considered investment strategies are then evaluated considering the rates of return realized out-of-sample. Attention is paid to the impact of transaction costs on the portfolio composition.
Finally, index tracking is a popular financial management strategy in which a fund manager tries to mimic the performances of a specified stock market index. The simplest way to track a market index is full replication, i.e. purchasing the securities in the exact same proportions that compose the market index. However, full replication implies several disadvantages, first of all because the market index return does not consider transaction costs that are paid by the fund manager. On the other hand, some fund managers try to outperform the market index. The latter strategy is usually referred to as enhanced index tracking and aims at generating portfolio returns outperforming those yielded by the market index. We propose a MILP formulation for both problems of index tracking and enhanced indexation. Our formulations include real-life features from the practice of portfolio selection, such as proportional and fixed transaction costs, a limit on the maximum number of securities composing the portfolio, limits on the minimum and maximum weights of each security in portfolio to avoid excessive fragmentation of the capital invested, and a constraint limiting the total amount of transaction costs paid. After validating the index tracking model using benchmark in-

stances, we propose a heuristic framework for its solution. We have called the heuristic framework Enhanced Kernel Search. Despite the heuristic framework is introduced referring to the specific problem of index tracking, the Enhanced Kernel Search can be easily generalized to a large class of combinatorial problems. The proposed framework allows implementing heuristics that have two major features from a practical point of view. Firstly, they require only little implementation efforts because the most demanding part of the search is carried out by a (any) general-purpose LP and MILP solver. Secondly, the same heuristic is applicable to a large set of problems. Several heuristic implementations of the framework are proposed. In this research we formally introduce the Enhanced Kernel Search and show its effectiveness solving the index tracking problem by means of the proposed heuristics. Benchmark instances for the index tracking problem have been used as data set.

1.1 Structure of the Thesis

The thesis is organized as follows

- **Chapter 2: Portfolio Optimization.** The chapter is devoted to a brief survey on the main single-period portfolio optimization models available in the literature. Evidence is provided to the risk measure analyzed and tested in this thesis.

- **Chapter 3: On the Effectiveness of Scenario Generation Techniques in Single-Period Portfolio Optimization.** In this chapter we survey different techniques to generate scenarios for the rates of return. We provide in-sample and out-of-sample analysis of the portfolios selected by using these techniques to generate the rates of return. Evidence on the computational burden required by the different techniques to generate scenarios is also provided.

- **Chapter 4: Effectiveness of Robust Portfolio Optimization Techniques.** In this chapter we summarize the main contributions appeared in the literature on robust optimization. Attention is paid to papers dealing with robust portfolio optimization. After introducing the basic definitions used in robust optimization, we briefly recall two well-known robust optimization approaches. We then propose two robust formulations for the Conditional Value-at-Risk (CVaR) optimization model according to the two former techniques. The performances of the optimal portfolios selected by the two robust models are compared with those yielded by the portfolios selected by the nominal model.

- **Chapter 5: Models and Simulations for Portfolio Rebalancing.** In this chapter we modify the single-period CVaR optimization model with transaction costs to introduce portfolio rebalancing. The aim is to provide investors and financial institutions with an effective tool to better exploit new information made available by the market. We then suggest a procedure to use the proposed optimization model in a rebalancing framework.

- **Chapter 6: Index Tracking and Enhanced Index Tracking.** In this chapter we propose a new formulation to both problems of passive and active portfolio management. Firstly, we briefly review the literature related to index tracking and enhanced indexation, as well as to heuristic methods solving the former problems. Subsequently, we introduce a heuristic

framework called Enhanced Kernel Search. We analyze and evaluate the behavior of several implementations of the Enhance Kernel Search framework to the solution of the index tracking problem.

- **Chapter 7: Conclusions.** This chapter contains the concluding remarks of the thesis.
- **Chapter 8: Appendix: Introduction to Linear Programming.** This appendix provides a short introduction to linear and integer linear programming problems.

2

Portfolio Optimization

2.1 Optimizing Portfolios: Literature Review

In 1950's Markowitz (see [117] [118]) formalizes the portfolio selection problem in terms of two criteria to be optimized: the mean, representing the expected outcome to be maximized, and the risk, a scalar measure of the volatility of outcomes to be minimized. Markowitz assumptions imply that only expected return and volatility matter to the investor. In his view, the investor is indifferent to any other characteristic of the distribution of returns, such as its skewness (a measure of the level of asymmetry) or its kurtosis (a measure of the thickness of the tails).
The original Markowitz formulation adopts the variance as risk measure, thus resulting in a quadratic programming problem. Actually, the LP solvability is of crucial importance for financial applications dealing with real-life features that require the introduction of binary and/or integer decision variables, such as minimum transaction lots (e.g., see [113]), cardinality constraints (e.g., see [86]), and transaction costs (e.g., see [34] and [96]). Therefore, since Markowitz, a number of other portfolio optimization models have been proposed in the literature. Following Sharpe [139], many other attempts to linearize the portfolio optimization problem have been made. Though the mean absolute deviation has been early considered in portfolio analysis (see [140]), only in 1990's a portfolio LP formulation that minimizes this risk measure has been proposed and analyzed by Konno and Yamazaki [97], the so-called Mean-Absolute Deviation (MAD) model. Yitzhaki [149] proposes the Gini's Mean (absolute) Difference (GMD) as risk measure, leading to the so-called GMD model. Young [151] suggests adopting the worst-case scenario as risk measure (the minimax approach), whereas Ogryczak [125] introduces the multiple criteria LP model that covers all the above as special aggregation techniques.
In spite of its historic importance, the Markowitz model is often criticized since it is not consistent with axiomatic model of preferences for choices under risk (see [136]). On the other side, researchers found that under the assumption of rates of return normally distributed, the mean absolute deviation and the Gini's mean difference become proportional to the standard deviation (see [100], pp. 1216-1217). Therefore, under some assumptions, the MAD and GMD models are equivalent to the Markowitz mean-variance model. Moreover, for general random variables consistency with the stochastic dominance relations is shown for the MAD model by Ogryczak and Ruszczyński [126], for the GMD model by Yitzhaki [149], and for other LP solvable models by Ogryczak [125]. Furthermore, Artzner *et al.* [5] define the class of coherent risk measures by means of four axioms. The coherence has been shown for the MAD model (see [127]) and for some other LP computable measures (see [1]).
As a matter of fact any rational investor is more concerned with underperformances rather than

overperformances of a portfolio. Therefore, many scholars claim that the volatility of the rate of return over the mean should not be penalized. Markowitz [118] proposes using downside risk measures and suggests to use the (downside) semivariance as a risk measure. Some authors point out that the MAD model allows specific modeling of the downside risk (e.g., see [144]). In fact, most of the LP solvable models may be reformulated as based on some downside risk measures. Additionally, the models may be extended with some piecewise linear penalty (risk) functions to provide opportunities for more specific modeling of the downside risk (see Carino *et al.* [31] and Michalowski and Ogryczak [119]). Recently, shortfall or quantile risk measures have gained more popularity in various financial applications (e.g., see [111] and references therein). Those measures can be formulated as safety measures to be maximized, like the worst realization analyzed by Young [151], and the second order quantile risk measure commonly called Conditional Value-at-Risk (CVaR) or Expected Shortfall differently introduced by several authors (see Rockafellar and Uryasev [135], and Acerbi and Tasche [1]) and representing the mean shortfall at a specified confidence level. In contrast to risk measures, the safety measures are in general consistent with formal models of risk-averse preferences. Moreover, as shown in Mansini *et al.* [111], for any risk measure a corresponding safety measure can be defined and vice-versa. The CVaR measure has several interesting theoretical properties. It satisfies the requirements of coherent risk measures (see [132] for a proof of the coherence) and it is consistent with the Second-degree Stochastic Dominance (SSD) (see [127]). Furthermore, the frequent application of CVaR models to various financial optimization problems is demonstrated by several empirical studies that appeared in the literature (e.g., see [2], [111], [135]) and from the fact that this measure is gaining importance even in banking, substituting the more commonly used measure of Value-at-Risk (VaR) defined as the maximum loss at a specified confidence level (see [87] and references therein). Actually, VaR as a risk measure is heavily criticized for not being sub-additive, i.e. VaR is not a coherent measure of risk. This means that the risk of a portfolio, when measured by the VaR, can be larger than the sum of the stand-alone risks of its components (see [5]). Hence, managing risk by VaR may fail to stimulate diversification. Moreover, VaR does not take into account the severity of an incurred damage event. As a response to these deficiencies several authors propose to replace the VaR with the CVaR as a measure of risk.

Structure of the Chapter. In Section 2.2 we introduce the classical Mean-Variance model proposed by Markowitz and the notation we will use throughout the thesis. In Section 2.3 we describe the CVaR safety measure and we present the corresponding LP optimization model.

2.2 The MV Portfolio Selection Model

Let $N = \{1, \ldots, n\}$ denote a set of securities (risky assets) available for an investment. For each security $j \in N$, its rate of return is represented by a random variable $\tilde{r}_j$ with a given expected value $\bar{r}_j = \mathbb{E}(\tilde{r}_j)$. Let us define as x_j, $j = 1, \ldots, n$, the continuous decision variable representing the fraction of capital (weight) invested in security j. We assume that the sum of the weights is equal to one and that no short sales are allowed, i.e. $\sum_{j=1}^{n} x_j = 1$ and $x_j \geq 0$ for $j = 1, \ldots, n$. Each portfolio $\mathbf{x}$ defines a random variable $\tilde{r}_{\mathbf{x}} = \sum_{j=1}^{n} \tilde{r}_j x_j$ representing the portfolio return. Its expected value is given by $\mu(\mathbf{x}) = \mathbb{E}(\tilde{r}_{\mathbf{x}})$.

Let σ_j, $j = 1, \ldots, n$, denotes the standard deviation of random variable $\tilde{r}_j$. Moreover, let $\Sigma \in \mathcal{M}(n \times n)$ denotes the covariance matrix, i.e. $\Sigma = \mathbb{E}[(\tilde{r} - \bar{r})^2]$ where $\tilde{r} \in \mathcal{R}^n$ is the vector of

random rates of return and $\bar{r} \in \mathcal{R}^n$ is the vector of expected returns. Therefore, given a portfolio $\mathbf{x}$ its variance is computed as $\sigma^2(\mathbf{x}) = \mathbf{x}^T \Sigma \mathbf{x}$.
In the following we assume that rates of return are *discrete* random variables, and define the support of each random variable as a set of cardinality T, i.e. every random variable can take at most T different possible values. We assume the knowledge of the random variable realization in each point of its support, and we call a point of the support a *scenario*. Hence, the random returns are defined on a discrete probability space $\{\Omega, \mathcal{F}, \mathbb{P}\}$, where the nonempty set $\Omega = \{\omega_1, \ldots, \omega_T\}$ is the sample space, $\mathcal{F}$ is a σ-field of subsets of Ω and $\mathbb{P}$ is a probability measure. For each random variable $\tilde{r}_j$, let r_{jt} denotes its realization under scenario t. For the sake of simplicity, let us assume that the T possible values the random variable $\tilde{r}_j$ can take are equally likely, i.e. $p_t = 1/T$, for $t = 1, \ldots, T$. Thus, the mean rate of return for security j is computed as $\bar{r}_j = \mathbb{E}(\tilde{r}_j) = \sum_{t=1}^{T} p_t r_{jt} = \frac{1}{T} \sum_{t=1}^{T} r_{jt}$. Let $\mu_t = \sum_{j=1}^{n} r_{jt} x_j$ be the realization of the portfolio return $\tilde{r}_{\mathbf{x}}$ under scenario t, then $\mu(\mathbf{x}) = \mathbb{E}(\tilde{r}_{\mathbf{x}}) = \sum_{t=1}^{T} p_t \mu_t = \frac{1}{T} \sum_{t=1}^{T} \left[\sum_{j=1}^{n} r_{jt} x_j \right] = \sum_{j=1}^{n} \bar{r}_j x_j$. Furthermore, the covariance between the random return of security $j \in N$ and security $i \in N$ is expressed as $\sigma_{ij} = \mathbb{E}[(\tilde{r}_i - \bar{r}_i)(\tilde{r}_j - \bar{r}_j)] = \frac{1}{T} \sum_{t=1}^{T} [(r_{it} - \bar{r}_i)(r_{jt} - \bar{r}_j)]$. In the end, the portfolio variance is computed as $\sigma^2(\mathbf{x}) = \frac{1}{T} \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j \sigma_{ij}$.
The Markowitz Mean-Variance (MV) portfolio selection problem can then be formulated as the following QP model

$$MV(\mathbf{x}) = \min \quad \sigma^2(\mathbf{x}) = \frac{1}{T} \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j \sigma_{ij} \tag{2.1}$$

$$\text{subject to } \sum_{j=1}^{n} \bar{r}_j x_j \geq \mu_0 \tag{2.2}$$

$$\sum_{j=1}^{n} x_j = 1 \tag{2.3}$$

$$x_j \geq 0 \quad j = 1, \ldots, n. \tag{2.4}$$

The objective function (2.1) aims at minimizing the portfolio risk, measured by its variance. The mean portfolio rate of return is constrained, by means of (2.2), to be at least equal to a lower bound μ_0 chosen by the investor. Constraint (2.3) represents the budget constraint, i.e. the investor invests the entire capital available. In the end, the non-negativity constraints (2.4) avoid short selling.
Each feasible portfolio, i.e. every feasible solution of model (2.1)-(2.4), can be plotted in a risk-return space, where the collection of all such feasible portfolios defines a specific region.
For the general MV model the notion of "optimal" portfolio can be defined in one of the two following ways.

1. For any value of variance, one should consider all the portfolios which have that variance. From among them all, one should select the one which has the highest expected return.

2. For any value of expected return, one should consider all the portfolios which yield that expected return. From among them all, one should select the one which has the lowest volatility.

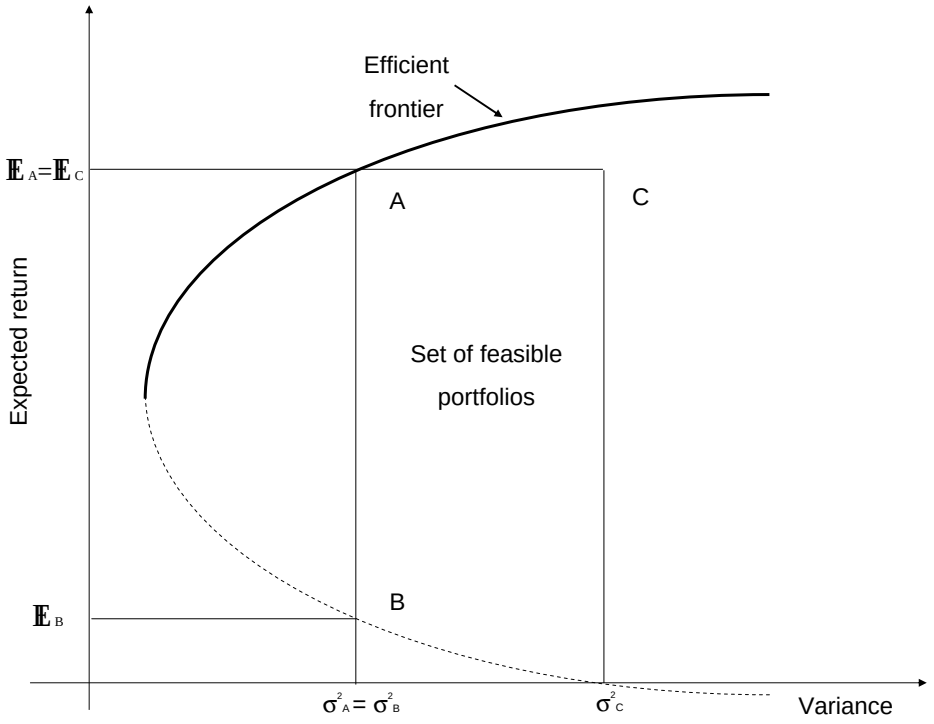

Figure 2.1: The Mean-Variance efficient frontier.

Each definition produces a set of optimal portfolios. Definition (1) produces an optimal portfolio for each possible level of risk. Definition (2) produces an optimal portfolio for each value of the expected return. Actually, the two definitions are equivalent. Therefore, the set of optimal portfolios obtained using one definition is exactly the same set which is obtained from the other. That set of optimal portfolios is called the *efficient frontier*. An example of the MV efficient frontier is illustrated in Figure 2.1. The region inside the parabola corresponds to the set of feasible portfolios. Indeed, for each point in that region, there exists at least one feasible portfolio selected investing in the universe of assets that has the risk and return corresponding to that point. The region outside the parabola is the unachievable portfolio space. No portfolios can be constructed corresponding to any point in that region. The line along the upper edge of the parabola is known as the efficient frontier. The portfolios that correspond to points on that curve are *optimal*. As an example, consider the three feasible portfolios A, B, and C depicted in Figure 2.1. Any rational investor, for a given level of risk, prefers the portfolio that yields the highest expected return (definition (1)). Therefore, portfolio A dominates portfolio B since it yields a higher expected return incurring the same level of risk. Moreover, the same rational investor prefers portfolio A to portfolio C, since it incurs a lower level of risk yielding the same expected return (definition (2)). The efficient frontier is convex. This is because the risk-return characteristics of a portfolio change in a non-linear (parabolic) fashion as its component weights change. Finally, the lower edge of the feasible portfolio region is the non-efficient frontier. For any portfolio that corresponds to one point on that portion of the curve, there exists at least one feasible portfolio that dominates it in terms of risk-return trade-off, i.e. for the same value of variance it yields a higher expected return.

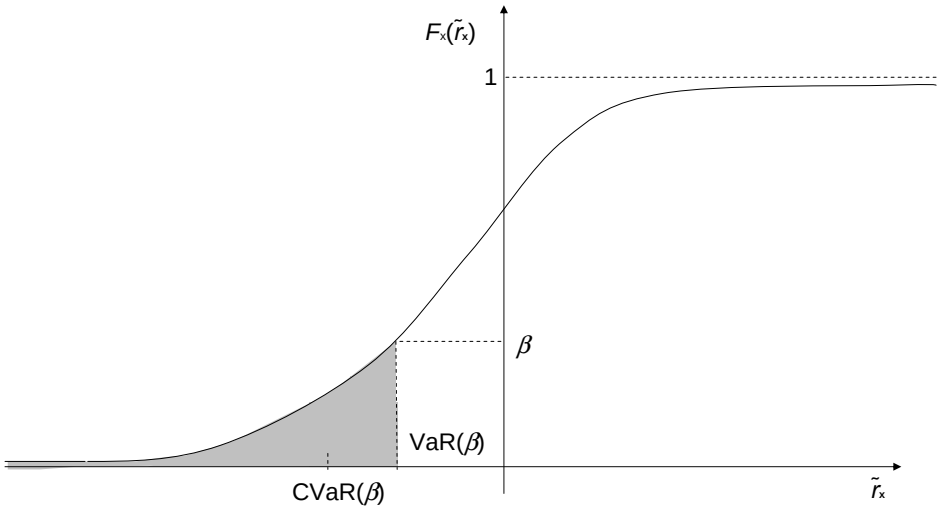

Figure 2.2: An example of portfolio VaR(β) and CVaR(β).

2.3 The CVaR(β) Portfolio Selection Model

For the discrete random variable $\tilde{r}_{\mathbf{x}}$, we can define the *worst realization* as

$$M(\mathbf{x}) = \min_{t=1,\dots,T} \mu_t.$$

Young [151] proposes selecting a portfolio based on the maximization of the worst realization, i.e. $\max \quad M(\mathbf{x}) = \max_{\mathbf{x}} \min_{t=1,\dots,T} \mu_t$. A natural generalization of measure $M(\mathbf{x})$ is the *Conditional Value-at-Risk* (CVaR) defined as the mean of a specified size (quantile) of worst realizations. For the simplest case of equally probable scenarios, one may define the CVaR($\frac{k}{T}$) as the mean return under the k worst scenarios, where $k = \beta T$. Formally, the CVaR(β) for any (real) tolerance level $0 < \beta \leq 1$ is defined as

$$\text{CVaR}(\beta) = \frac{1}{\beta} \int_0^{\beta} F_{\mathbf{x}}^{(-1)}(\alpha) d\alpha \tag{2.5}$$

where $F_{\mathbf{x}}^{(-1)}(p) = \inf \{\eta : F_{\mathbf{x}}(\eta) > p\}$ is the right-continuous inverse of the cumulative distribution function of the rate of return $F_{\mathbf{x}}(\eta) = \mathbb{P}(\tilde{r}_{\mathbf{x}} \leq \eta)$. For any $0 < \beta \leq 1$, the CVaR(β) is an SSD consistent measure, whereas -CVaR(β) is coherent (for details see Mansini *et al.* [111]). Note that CVaR(1)= $\mu(\mathbf{x})$ and CVaR(β) tends to Young's $M(\mathbf{x})$ measure when β tends to 0.
Generally speaking, the CVaR(β) can be illustrated as follows. In Figure 2.2 we depicted the cumulative distribution function of a possible portfolio rate of return. Subsequently, we chose a tolerance level β. For that value, one can easily find the corresponding Value-at-Risk (VaR(β)), defined as the β-quantile of the distribution, i.e.

$$\text{VaR}(\beta) = F_{\mathbf{x}}^{(-1)}(\beta).$$

Intuitively, for a given value β the VaR(β) of a portfolio is the value such that there is a probability equal to β that the return is worse than VaR(β). On the other side, the CVaR(β) can be interpreted as the expected value of the returns worse than the VaR(β), i.e. the expected value of the shaded area. A possible value for the CVaR(β) has been depicted.

The portfolio selection problem based on this safety measure can be formulated as the following LP problem (see [135],[132], but also [127])

$$CVaR(\beta)(\mathbf{x}) = \max\,[\eta - \frac{1}{\beta T}\sum_{t=1}^{T} d_t] \tag{2.6}$$

$$\text{subject to} \quad d_t \geq \eta - \mu_t \quad t = 1, \ldots, T \tag{2.7}$$

$$\sum_{j=1}^{n} \bar{r}_j x_j \geq \mu_0 \tag{2.8}$$

$$\sum_{j=1}^{n} x_j = 1 \tag{2.9}$$

$$x_j \geq 0 \quad j = 1, \ldots, n \tag{2.10}$$

$$d_t \geq 0 \quad t = 1, \ldots, T, \tag{2.11}$$

where η is an auxiliary (unbounded) variable representing the β-quantile at optimum, i.e. the VaR(β). The non-negative variable $d_t, t = 1, \ldots, T$, measures the maximum positive deviation of the portfolio return realization μ_t from the β-quantile, i.e. $d_t = \max\{0, \eta - \mu_t\}$. The objective function (2.6) maximizes the safety measure represented by the CVaR(β). Under each scenario t, constraint (2.7) along with constraint (2.11) force variable d_t taking the maximum value between 0 and $\eta - \mu_t$. Specifically, if under scenario t the portfolio return μ_t is lower than the β-quantile η, i.e. $\eta > \mu_t$, then by means of constraint (2.7) the non-negative variable d_t takes value equal to the deviation of the portfolio return from the β-quantile. Conversely, in case $\eta \leq \mu_t$, the maximization of the objective function forces variable $-d_t$ to take value equal to 0. Constraints (2.8) and (2.9) represent the minimum portfolio return and budget constraints, respectively.
For a given value of the quantile parameter β, the set of optimal solutions of model (2.6)-(2.11) defines a mean-CVaR(β) efficient frontier that can be depicted, similarly to what we have done for the MV model, in the risk-return space, where the risk is now measured by the CVaR(β).

3

On the Effectiveness of Scenario Generation Techniques in Single-Period Portfolio Optimization

3.1 Introduction

The main challenge in portfolio selection problems is to provide evidence on how effective the mathematical models are as decision tools in determining reliable solutions to face uncertain future events. As a matter of fact, the effectiveness of the models strongly depends on the data input required by the models and on the methods used to generate it. Most of mathematical models for portfolio selection require the availability of a set of *scenarios*, where a scenario is a realization of a multivariate random variable representing the rates of return of all the securities. The accuracy and effectiveness of the models rely on the quality of the generated scenarios. Different methods can be used to generate scenarios. They range from the simple historical approach, based on the assumption that past realizations are representative of future outcomes, to more complex methods based on randomly re-sampling from historical data (Bootstrapping methods) or on randomly sampling from a chosen distribution function of the multivariate random variable (Monte Carlo simulation) or, again, forecasting methods. Each method used to generate scenarios is called a *scenario generation technique*.

Several academic researchers and practitioners have used scenario generation techniques as tools for supporting financial decision making. The applicability of these techniques for financial purposes has been first recognized by Bradley and Crane in [26] and, more recently, by Mulvey and Vladimirou [122] for asset allocation. Dembo *et al.* [47] have discussed the importance of using scenarios in risk management, and have presented various methodologies to generate them. Scenario generation techniques have also been used by Consiglio and Zenios [40] for designing the debt structure of an agency issuing a portfolio of callable bonds, by Carino *et al.* [30] and by Consiglio *et al.* [38] for insurance companies.

Although most of the scenario generation techniques proposed in the literature have been developed for stochastic programs based upon event trees (e.g., see [98]), many of these techniques are suitable, or easily adaptable, to single-period portfolio optimization problems.

This chapter aims at comparing different techniques for the generation of scenarios of rates of return in a portfolio selection problem and at evaluating their effectiveness for single-period portfolio management. As the domain of scenario generation techniques is very wide, we have selected a

few techniques that seems to be more promising than others when applied to the problem under examination. Specifically, we have analyzed five different scenario generation techniques, three of which are non-parametric while the remaining two are parametric. We have compared the in-sample and the out-of-sample characteristics of the optimal portfolios obtained by solving the same portfolio selection model when such techniques are used to generate the input data (the rates of return scenarios). Several historical data sets are used to span different market trends. The main objective of the work is to provide evidence on the model effectiveness in guiding financial decisions under different market conditions and to find out whether there exists a technique that outperforms all the others under different market trends. The portfolio optimization model used as a basis for the comparison of the scenario generation techniques is based on the CVaR(β) model introduced in Section 2.3.
Finally, notice that, according to the aims of this research, any other single-period portfolio selection model based on a different risk measure could have been used instead. Furthermore, the reader should be aware that the scenarios could have been alternatively generated by means of any other scenario generation technique suitable for single-period portfolio optimization models, e.g. the moment matching method proposed by Høyland and Wallace [85].
Structure of the Chapter. In Section 3.2 we provide a short description of the selected portfolio optimization model. Section 3.3 introduces the scenario generation techniques analyzed and tested in the present chapter. Section 3.4 deals with the experimental analysis and the comparison of the characteristics of the optimal portfolios obtained by using the different scenario generation techniques in the CVaR(β) model. Some concluding remarks are presented in Section 3.5, while in the Appendix a more precise description for one of the analyzed parametric scenario generation techniques is provided.
The content of this chapter has been published in [81].

3.2 The CVaR(β) Model with Transaction Costs

We consider a situation where an investor intends to optimally select a portfolio of securities and to hold it until the end of a defined investment horizon. In order to correctly simulate the trading conditions encountered by the investor, we assume that, for each selected security, a fixed and a proportional cost are applied. The introduction of fixed transaction costs implies some changes to the model described in Section 2.3. Firstly, let $\mathbf{X} = (X_j)_{j=1,\dots,n}$ denote the vector of decision variables where X_j, $j = 1, \dots, n$, represents the fractional value of stock units invested in security j. Let q_j be the quotation of security j, $j = 1, 2, \dots, n$, at the date of portfolio selection and $q_j X_j$ be the amount invested in security j at the same date. We define as f_j the fixed transaction cost incurred by the investor when selecting security j and as c_j the corresponding proportional transaction cost. Finally, C is the capital available for the investment whereas $u_j, j = 1, 2, \dots, n$, represents the upper limit on the fractional number of units of security j that the investor can purchase. For any $0 < \beta \leq 1$, the CVaR(β) model with transaction costs, by simplicity referred to as CVaR(β) model, can be defined as follows

CVaR(β) Model

$$\max \quad \eta - \frac{1}{\beta}\sum_{t=1}^{T} p_t d_t \tag{3.1}$$

$$\text{subject to} \quad \eta - \sum_{j=1}^{n}(r_{jt} - c_j)q_j X_j + \sum_{j=1}^{n} f_j z_j \leq d_t \quad t = 1, \ldots, T \tag{3.2}$$

$$\sum_{j=1}^{n}(\bar{r}_j - c_j)q_j X_j - \sum_{j=1}^{n} f_j z_j \geq \mu_0 \sum_{j=1}^{n} q_j X_j \tag{3.3}$$

$$\sum_{j=1}^{n} q_j X_j = C \tag{3.4}$$

$$X_j \leq u_j z_j \quad j = 1, ..., n \tag{3.5}$$

$$d_t \geq 0 \quad t = 1, ..., T \tag{3.6}$$

$$X_j \geq 0 \quad j = 1, ..., n \tag{3.7}$$

$$z_j \in \{0, 1\} \quad j = 1, ..., n. \tag{3.8}$$

The objective function (3.1) maximizes the safety measure represented by the Conditional Value-at-Risk. Constraints (3.2) along with constraints (3.6) define the non-negative variables d_t as $\max\{0, \eta - y_t\}$, where $y_t = \sum_{j=1}^{n}(r_{jt} - c_j)q_j X_j - \sum_{j=1}^{n} f_j z_j$ is the net portfolio return under scenario t. Thus, each variable d_t measures the deviation of the portfolio net return y_t from the $\beta-$quantile η when $y_t < \eta$, whereas it is equal to zero in all the other cases. Constraint (3.3) establishes that the portfolio net mean return, expressed as difference between the portfolio mean return and the total transaction costs (proportional and fixed), must be at least equal to the portfolio required return $\mu_0 C$. Constraint (3.4) imposes that the total investment in the portfolio must be equal to C. Constraints (3.5) define the upper bound u_j on the investment in each security j. Since the fixed cost f_j, $j = 1, 2, \ldots, n$, is paid only if security *j* is selected, we introduce a binary variable z_j, $j = 1, 2, \ldots, n$, which is forced by constraints (3.5) to take value 1 if $X_j > 0$. Notice that if $X_j = 0$ then z_j is free to take any value. However, since the fixed costs influence the risk function, at optimum z_j will take value 0 as the most convenient of the two. Finally, constraints (3.7) avoid short sales on securities whereas constraints (3.8) define the binary conditions on variables z_j.

3.3 Scenario Generation Techniques

The LP based portfolio selection model implemented assumes that returns are discrete random variables. Given n securities, a scenario consists of n return realizations, one for each security. We will refer to the t-th realization of the rate of return of security j as its realization under scenario t. To estimate portfolio expected return and risk, T mutually exclusive scenarios, each of which occurring with probability p_t, $t = 1, ..., T$, are required. Herein, we shortly present the scenario

generation techniques we selected. Additional information can be found in appendix of this chapter, while interested readers are referred to Kouwenberg and Zenios [99] and references therein. Among the techniques proposed in the literature, we have analyzed and tested the followings.

- *Historical Data technique* (Hist. Data). This method is one of the most frequently used for its simplicity. It is based upon the assumption that historical data are possible future scenarios. A scenario corresponds to the joint realizations of the rates of return for all securities as observed in a given time period. Usually scenarios are treated as equally probable. Such approach does not require any assumption on the distribution function for the rates of return and no correlations between securities have to be computed since they are implicitly considered in the data observed on the market. This procedure preserves the historical mean and variance of the returns. The potential drawback of this approach, and of all approaches making use of historical data, is that future price movements may be substantially different in nature from those observed in the past. Moreover, the number of scenarios that can be produced is limited by the amount of historical data available. In the remaining of the thesis we will refer to such scenario generation method as *Hist. Data*. Two of the most significant applications of this technique to single-period portfolio optimization problems can be found in Konno and Yamazaki [97] and in Young [151].

- *Bootstrapping technique* (Boot(T)). The method combines the use of historical data with a bootstrapping technique. As for the Hist. Data technique, a scenario corresponds to the joint realizations of the rates of return for all securities. Each scenario from the original historical data set can be re-sampled with a constant probability given by one over the cardinality of the set to select the scenario t to be re-sampled an integer number t, uniformly distributed between 1 and the cardinality of the historical data set, is generated. The procedure is repeated until the new sample reaches the desired size. The approach preserves the correlations between securities. In computational experiments we will refer to such method as *Boot(T)*, where T represents the sample size after the re-sampling. An excellent description of the technique used with inferential purposes can be found in Efron and Tibshirani [56], where the expanded sample is treated as a virtual population from which samples are drawn to verify the estimators variability. Frequently, the bootstrapping procedure is used when the size of the available sample is relatively small and one needs a larger number of observations (e.g., see Chong Ho [35]). The use of the bootstrapping technique for scenario generation purposes has been suggested by Kouwenberg and Zenios [99], and it has been used by, for example, Consiglio *et al.* [39] to study the problem of asset and liability management of participating insurance policies with guarantees.

- *Block Bootstrapping technique* (Block-Boot(T)). The Block Bootstrapping technique is a variant of the previous one which allows to retain original data correlations between periods through the use of the bootstrapping on blocks of scenarios (see Bühlmann [27]). A block is a set of consecutive historical scenarios. Given the original time series and once the block length has been chosen, the method, instead of single scenarios, re-samples blocks of the same length from the original set. The idea is to choose a block length large enough to guarantee that observations for an interval larger than such value will be nearly independent. For a more complete analysis on rules to correctly compute the block length we refer to Hall *et al.* [84]. In the remaining of the thesis we will refer to such scenario generation method

as *Block-Boot(T)*, where T is the cardinality of the scenarios after the re-sampling.

Both the Boot(T) and the Block-Boot(T) techniques are immune of systematic errors of selection. The potential drawback of these methods is due to the use of historical data, since future rates of return may be substantially different in nature from the past observed data.

- *Monte Carlo simulation techniques* (M-Norm(T), M-tStud(T)). Monte Carlo simulation is a parametric approach that consists in generating scenarios according to a specified distribution function. When using these techniques in the domain of portfolio selection problems, the first critical issue is the choice of the multivariate distribution function which better fits the historical rates of return. As a matter of fact, the most frequently used distribution function is the multivariate standard Normal distribution, $N(\mathbf{0}, \mathbf{I_n})$, with zero mean and unit variance-covariance matrix $\mathbf{I_n}$, but other distribution functions can be alternatively used. Since multivariate Normal distribution does not consider the "fat-tails" or "heavy-tails" effect (see Mandelbrot [110]) which frequently characterizes rates of return, we have decided to also consider the multivariate $t-$Student distribution as a possible alternative to the Normal distribution. We will refer to the former approach as to *M-Norm(T)* and to the latter as to *M-tStud(T)*, where T is the number of scenarios sampled from the chosen distribution. To generate scenarios from a multivariate Normal distribution with known mean and covariance matrix we have used the method proposed in Levy [103] (pages 234–243), while to estimate the number of degrees of freedom v that controls the heaviness of the tails in a multivariate $t-$Student we have used empirical evidence suggesting that such parameter should range between 3 and 7 for most of the markets (see Glasserman [71], pages 510–511). More precisely, we have decided to consider n different parameters v_j, one for each security $j = 1, ..., n$, to better take into account the different heaviness of the return tails in each of the available securities (kurtosis). This procedure is different from the traditional one but allows a more precise analysis of the extreme values in the return distribution of each security by better preserving similarities with historical data.

- *Multivariate Generalized ARCH Process technique* (M-GARCH(1,1)(T)). Volatility cluster effect is a typical feature characterizing financial time series. Such effect is related to the fact that volatility for financial time series is usually not constant over time (homoskedasticity) but presents heteroskedasticity so that large rates of return tend to be followed by other large values, of both signs, whereas small rates of return are usually followed by small values (see Bollerslev *et al.* [24]). The most used stochastic process taking into account such effect is the GARCH(q, p) (Generalized ARCH process, see Bollerslev [22]), where q and p are the number of the required past residuals and of the past conditional volatility values, respectively. Many studies (see Bollerslev *et al.* [25] and, more recently, Ding and Engle [52]) extend the original univariate GARCH(q, p) stochastic process to the multivariate case. Usually, the number of parameters to be estimated in a conventional multivariate GARCH(q, p) is too large so that various approximations have been proposed. Due to its computational simplicity, the Constant Conditional Correlation GARCH (CC-MGARCH) proposed by Bollerslev [23] is widely used to estimate the multivariate GARCH stochastic process starting from univariate GARCH processes. We will refer to this approach as *M-GARCH(1,1)(T)*, since we have set parameters p and q to 1, where T is the number of generated scenarios. The M-

GARCH(1,1)(T) stochastic process can be used as a scenario generation technique through the construction of an *event tree*. Each *node* in the tree is associated with a joint outcome of the rates of return of all securities. An *edge* between two consecutive nodes means the child node has been derived from the parent node according to the implemented stochastic process. Each parent node can generate more than one child node. Nodes with the same distance from the root node are said to belong to the same *stage*. A *scenario* is associated to each leaf (pending node) of the tree. When generating the event tree, a nodal point concerns the choice of the number of nodes and of stages to take into account. The objective is to find the right trade-off between a number of scenarios large enough to avoid approximation errors and small enough to be computationally tractable. We have decided to construct a binomial event tree where from each parent node two child nodes are generated and 2^l nodes belong to stage l. Thus, for instance, if the number of stages is 12, then 4096 scenarios will be generated. Details about the use of the CC-MGARCH stochastic process to generate a binomial event tree are provided in the Appendix of this chapter.

The first three described scenario generation techniques are non-parametric while the last two are parametric. The basic difference between a non-parametric and a parametric approach is that in the former case scenarios generated are represented by real data, while in the latter scenarios can be totally hypothetical. The non-parametric approaches have the main advantage to be simple to compute and easy to understand. On the contrary, a parametric approach depends on the chosen distribution function, and the choice of such distribution, along with its parameters, may be difficult. Moreover, there is no evidence that the distribution that better fits the data will remain constant over time. Nevertheless, there are many reasons to support the use of a parametric approach with respect to a non-parametric one. First of all, non-parametric techniques are criticized to be strongly dependent upon a defined sample so that they can be hardly generalized to other samples. Furthermore, it has been claimed that if the observations in a sample are biased, then non-parametric approaches will be biased too. One of the main objectives of the research described in this chapter is to evaluate if the additional computational burden usually implied by parametric approaches is really compensated by better portfolio performances.

3.4 Experimental Analysis

In this section we compare the portfolios selected by solving the CVaR(β) model with scenarios generated by using the five discussed techniques. Computational experiments have been conducted on a PC with a 3,000 MHz Intel Pentium III processor and 1 Gb of RAM. The model has been implemented in C++ by means of Concert Technology 2.0 and solved with CPLEX 9.0. We first present the testing environment, then the results of the in-sample and finally those of the out-of-sample.

3.4.1 Testing Environment

Historical data are represented by weekly rates of return, computed by using closing stock prices of the 100 securities composing the FTSE100 Index at the date of September 25th, 2005. No dividends have been considered. The FTSE100 Index is the capitalization-weighted index of the 100

most highly capitalized companies traded on the London Stock Exchange, representing approximately about 80% of the entire UK market. It is recognized as the benchmark for all UK financial markets.

We have tested the model under different market behaviors. We have constructed four data sets corresponding to different in-sample and out-of-sample time periods. Each data set temporal positioning is shown in Figure 3.1. The first data set is characterized by an increasing market trend in the in-sample period as well as in the out-of-sample period (hereafter called *up-up period*), the second data set by an increasing trend in the in-sample period and by a decreasing one in the out-of-sample period (*up-down period*), the third data set by a decreasing trend in the in-sample period and by an increasing one in the out-of-sample period (*down-up period*) and, finally, the last set by a decreasing trend in both the in-sample and the out-of-sample periods (*down-down period*). Each of these data sets consists of 2 years of in-sample weekly observations (104 realizations) and 1 year of out-of-sample ones (52 realizations).

For each data set we have solved the CVaR(β) model by considering $C = 100,000$ Euros, three

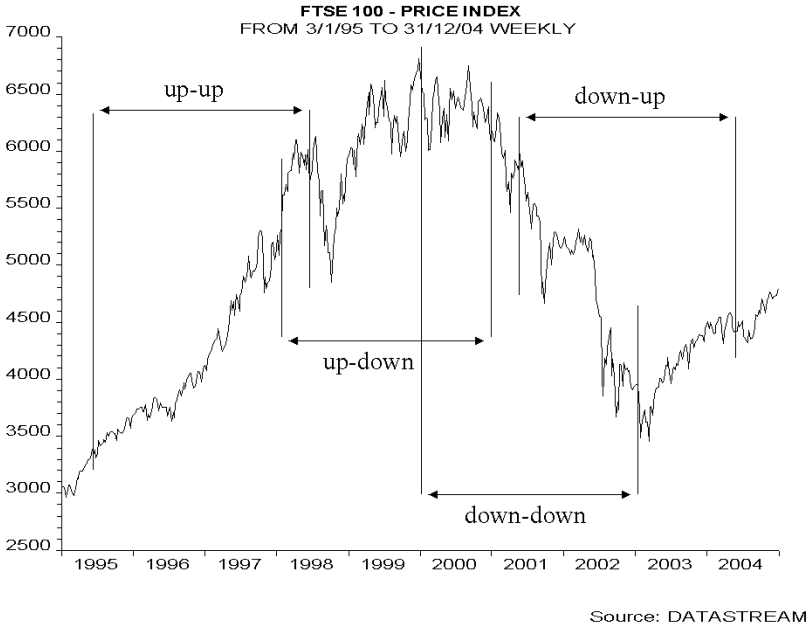

Figure 3.1: The four different market periods.

different values of the minimum required rate of return μ_0 (0, 0.05 and 0.10 on yearly basis) and three different levels for the quantile parameter β (0.01, 0.05 and 0.10). Moreover, we have assumed a fixed cost equal to 12 Euros and a proportional cost equal to 0.195% for all the securities. These are real-case conditions applied by Italian brokers for operations carried out on the FTSE market. Finally, upper bounds u_j are assumed to be computed as $u_j = C/q_j$, $j = 1, \ldots, n$.

Table 3.1 shows the number of scenarios generated by the different techniques for each set of data. For all the techniques but for the Hist. Data and for the M-GARCH(1,1)(T) the same number of scenarios has been set. In the case of historical data sampling (Hist. Data), there is only one sample represented by the 104 historical scenarios. For the M-GARCH(1,1)(T) approach the number of scenarios has to be a power of 2. Given a data set, for each value of β, each required rate of return μ_0, each technique and each scenario sample, an instance has been generated. This means 144 instances for each data set and 576 altogether.

Hist. Data	Boot(T)	Block-Boot(T)	M-Norm(T)	M-tStud(T)	M-GARCH(1,1)(T)
104	1000	1000	1000	1000	1024
	5000	5000	5000	5000	4096
	10000	10000	10000	10000	8192

Table 3.1: Number of scenarios (sample size) generated with each technique.

	sample size			
technique	104	1000	5000	10000
Hist. Data	0.17	–	–	–
Boot(T)	–	0.42	1.35	2.50
Block-Boot(T)	–	0.50	1.43	2.67
M-Norm(T)	–	5	14	25
M-tStud(T)	–	20	29	40
M-GARCH(1,1)(T)	–	60 (2^{10})	900 (2^{12})	3480 (2^{13})

Table 3.2: Average computational times (in minutes) to generate instances according to sample size.

Table 3.2 shows, for each scenario generation technique, the average (out of the 4 data sets) computational time required to build an instance as a function of the number of scenarios taken into account. For the M-GARCH(1,1)(T) model, the precise size of each sample is indicated in parentheses. Notice that, in the worst case, the non-parametric approaches build an instance in less than 3 minutes. Generating scenarios with both Monte Carlo simulation techniques is more time consuming (always more than 5 minutes). Larger times required to create an instance by using a multivariate $t-$Student distribution with respect to a Normal one are due to the additional computational burden implied by the estimation of the number of degrees of freedom v_j for each security. The most time consuming approach is the M-GARCH(1,1)(T) model. To create a binomial event tree even for the smallest instances, has required a time larger than one hour.

3.4.2 In-Sample Analysis

In this section we present the characteristics of the optimal portfolios selected by the CVaR(β) model when using the described techniques to generate the scenarios. For sake of clarity, we have decided to only report the results obtained by setting μ_0 equal to 5% on yearly basis and $\beta = 0.05$, respectively. The complete set of results is available in Guastaroba *et al.* [79].
As a starting point of the analysis, we have compared the portfolios obtained by using the same scenario generation technique when changing the size of the sample. The goal has been to verify if an ideal sample size can be worked out for each technique. At this aim we have analyzed the best ex-post performances over all the four data sets for each technique. The results indicate that the best sample size is equal to 1000 for the Boot(T) and Block-Boot(T) techniques, equal to 10000

for M-Norm(T) and M-tStud(T) techniques and to 4096 scenarios for M-GARCH(1,1)(T). A first conclusion which we feel we can draw is that with techniques like Bootstrapping and Block Bootstrapping it is not necessary to generate large samples. On the contrary, when using a parametric approach like Monte Carlo simulation it is evident how the larger the number of scenarios the better the selected portfolios. Usually, Monte Carlo simulation requires a massive number of scenarios. We have set the maximum sample size equal to 10000 since, in extensive preliminary analysis, we have evaluated that the slightly better performances which could be obtained with a sample size larger than 10000 do not justify the longer time required to solve the optimization model. For the M-GARCH(1,1)(T) technique the sample size equal to 4096 is selected since providing the best trade-off between average return and downside risk.
Tables 3.3–3.4 show the characteristics of the optimal portfolios selected by the model when using different scenario generation techniques over the four data sets. In particular, they report the results obtained by each technique when using its best sample size according to the previous analysis. Each table is divided into two parts providing the results for the two data sets characterized by the same trend in the in-sample period. Each part consists of four columns that show the number of securities selected (div), the minimum (min) and the maximum (max) portfolio shares and the computational time (in minutes) needed to optimally solve the instance. Summarizing the main figures about optimal in-sample portfolios, a consistent difference can be noticed between the cases with increasing trend in the in-sample period (i.e. *up-up* and *up-down* data sets) with respect to those with in-sample decreasing trend (i.e. *down-up* and *down-down* data sets). The number of securities selected when the market is decreasing in the in-sample period is considerably lower than the number when the market trend is increasing.

	up-up data set				**up-down data set**			
		shares		time		shares		time
instances	div	min	max	(in minutes)	div	min	max	(in minutes)
Hist. Data	14	0.026	0.244	6.10	11	0.020	0.247	0.02
Boot(1000)	14	0.027	0.244	65.65	11	0.027	0.316	0.45
Block-Boot(1000)	13	0.028	0.261	59.83	12	0.023	0.255	0.65
M-Norm(10000)	8	0.037	0.292	318.27	8	0.043	0.577	56.32
M-tStud(10000)	9	0.040	0.214	222.44	8	0.032	0.545	75.75
M-GARCH(1,1)(4096)	7	0.028	0.351	0.58	6	0.048	0.312	1.38

Table 3.3: Up-up and up-down data sets: Optimal portfolio characteristics with μ_0 = 5% and β = 0.05.

	down-up data set				**down-down data set**			
		shares		time		shares		time
instances	div	min	max	(in minutes)	div	min	max	(in minutes)
Hist. Data	4	0.078	0.679	0.01	5	0.066	0.588	0.01
Boot(1000)	5	0.061	0.663	0.14	5	0.083	0.544	0.43
Block-Boot(1000)	6	0.040	0.372	0.21	6	0.063	0.560	0.15
M-Norm(10000)	4	0.062	0.621	5.57	7	0.052	0.478	35.99
M-tStud(10000)	4	0.070	0.615	9.46	6	0.058	0.543	21.74
M-GARCH(1,1)(4096)	3	0.101	0.793	2.21	5	0.061	0.537	1.49

Table 3.4: Down-up and down-down data sets: Optimal portfolio characteristics with μ_0 = 5% and β = 0.05.

As far as the computational time is concerned, the most time consuming techniques are the two Monte Carlo approaches and, in particular, the M-Norm(10000). This is mainly due to the large number of scenarios required by this technique. Moreover, when comparing computational times required to solve instances of the same size, we have noticed that both Monte Carlo techniques usually require almost twice the computational time required by the Bootstrapping or the Block Bootstrapping techniques (see Guastaroba *et al.* [79]). When all the tested instances are taken into account the M-GARCH(1,1)(4096) is, on average, as time consuming as the Boot(1000) or the Block-Boot(1000) techniques and the Hist. Data technique is, on average, the most efficient one.
Finally, considering the computational times required to generate an instance (see Table 3.2), the overall most time consuming technique becomes the M-GARCH(1,1)(T) while the Hist. Data technique remains the most time saving technique.

3.4.3 Out-of-Sample Analysis

In the out-of-sample analysis, we examine the behavior of the portfolios selected by the CVaR(β) model in the 52 weeks following the portfolio selection date. As for the in-sample analysis, we only describe the results obtained with μ_0 equal to 5% on yearly basis and β equal to 0.05. Moreover, we describe the characteristics of the optimal portfolios selected by the model when each technique has been applied with its best sample size. Details on the remaining results can be found in Guastaroba *et al.* [79].
To compare the out-of-sample optimal portfolio performances we have computed cumulative returns. Let us define as r_1, r_2,..., r_m the daily ex-post portfolio rates of return over m periods. Then, the cumulative portfolio rate of return in period t, $t \leq m$, is equal to

$$((1 + r_1)(1 + r_2)...(1 + r_t)) - 1.$$

Furthermore, we computed the following six ex-post parameters: the number of times out of 52 the portfolio rate of return outperforms the corresponding required one (#), the average (r_{av}) and the median (r_{med}) of the rate of returns on yearly basis, the standard deviation (std), the semi-standard deviation (s-std), and the *Sortino* index. The latter parameter provides a measure of the over-performance of the portfolio mean rate of return with respect to the required one per unit of downside risk (here measured by the semi-standard deviation). All the dispersion measures (std, s-std) and the *Sortino* index have been computed with respect to the minimum required rate of return to make them directly comparable in the different instances.
In Table 3.5 we show all the resulting values for the performance measures and the dispersion parameters we have computed. For each data set the market behavior of the FTSE100 index is also shown. Figures 3.2–3.5 compare, for each data set, the ex-post cumulative returns of the market index FTSE100 with respect to the cumulative returns of the three best ex-post optimal portfolios. In some cases (see Figures 3.3 and 3.5) we have plotted four instead of three selected portfolios cumulative returns, since in that data set the behavior of two techniques (namely, the Hist. Data and the Boot(T) techniques) are very similar.
As far as the ex-post parameters are concerned, the portfolios selected by using the Hist. Data technique, despite the simplicity of the method, yield rates of return very close to those of all the other portfolios (see Table 3.5) and have a value of the downside risk measure very similar to that of the portfolios selected by the other non-parametric techniques.

model	#	r av	r med	std	s-std	Sortino index
up-up data set						
Hist. Data	32	19.29	175.58	0.0128	0.0072	0.3135
Boot(1000)	32	18.09	181.32	0.0125	0.0072	0.2876
Block-Boot(1000)	31	19.71	115.69	0.0115	0.0064	0.3758
M-Norm(10000)	32	25.28	20.48	0.0100	0.0048	0.7149
M-tStud(10000)	34	25.39	4.13	0.0116	0.0063	0.5324
M-GARCH(1,1)(4096)	33	39.55	-2.46	0.0177	0.0099	0.5193
FTSE100	32	45.92	53.36	0.0257	0.0153	0.3411
up-down data set						
Hist. Data	21	7.24	606.41	0.0358	0.0249	0.0087
Boot(1000)	24	3.79	460.22	0.0326	0.0229	-0.0054
Block-Boot(1000)	21	3.61	430.07	0.0331	0.0228	-0.0063
M-Norm(10000)	25	-8.74	54.38	0.0163	0.0137	-0.1342
M-tStud(10000)	25	-5.91	41.03	0.0149	0.0123	-0.1227
M-GARCH(1,1)(4096)	30	17.62	596.58	0.0355	0.0267	0.0430
FTSE100	29	-8.54	18.76	0.0267	0.0204	-0.0729
down-up data set						
Hist. Data	30	26.82	-25.71	0.0132	0.0062	0.5803
Boot(1000)	29	24.65	-30.93	0.0116	0.0052	0.6305
Block-Boot(1000)	34	70.19	33.67	0.0233	0.0080	1.2750
M-Norm(10000)	30	24.77	-31.40	0.0123	0.0052	0.6456
M-tStud(10000)	26	23.11	-32.11	0.0121	0.0052	0.5804
M-GARCH(1,1)(4096)	29	16.45	-36.70	0.0120	0.0073	0.2499
FTSE100	30	34.19	27.39	0.0228	0.0101	0.4236
down-down data set						
Hist. Data	22	-7.00	180.76	0.0121	0.0095	-0.1884
Boot(1000)	25	-4.70	165.91	0.0133	0.0106	-0.1321
Block-Boot(1000)	22	-5.01	-9.29	0.0143	0.0107	-0.1353
M-Norm(10000)	22	-7.28	98.39	0.0138	0.0111	-0.1577
M-tStud(10000)	23	-5.41	113.89	0.0128	0.0099	-0.1548
M-GARCH(1,1)(4096)	22	-9.82	142.82	0.0119	0.0102	-0.2146
FTSE100	22	-22.93	-14.30	0.0342	0.0283	-0.0852

Table 3.5: Out-of-sample statistics for $\mu_0 = 5\%$ and $\beta = 0.05$.

As expected, the Block-Boot(1000) dominates the Boot(1000) technique. In particular, the average rate of return of the portfolios selected by using the Block-Boot(1000) is altogether greater than that obtained by using the Boot(1000) technique, with a similar downside risk. In the *down-down* data set (see Figure 3.5) after 4 weeks from the portfolio selection date, the Block-Boot(1000) shows the best ex-post cumulative return until the 33rd week. In the *up-down* (see Figure 3.3) data set the portfolio selected by using the Block-Boot(1000) technique have a behavior similar to that of the other non-parametric approaches. Finally, in the *down-up* data set (see Figure 3.4) the Block-Boot(1000) is the technique that consistently outperforms all the other scenario generation techniques. To conclude, the Block-Boot(1000) seems to be the technique that in all the four data sets provides portfolios whose ex-post performance is never the worst and in several cases is the best. Moreover, such performances are obtained guaranteeing a downside risk that is not significantly different from that of the other non-parametric approaches and lower than the GARCH(1,1)(T) (see Table 3.5, especially for the s-std figures). Block-Boot(1000) turns out to be the best choice for an investor who might accept more risk only if compensated by a more than proportional increase in the expected return.

By analyzing the figures related to the parametric techniques, the portfolios selected by using the M-Norm(10000) and the M-tStud(10000) techniques are almost always those with the lowest downside risk, independently from the data set under analysis. In some data sets (see Table 3.5 for

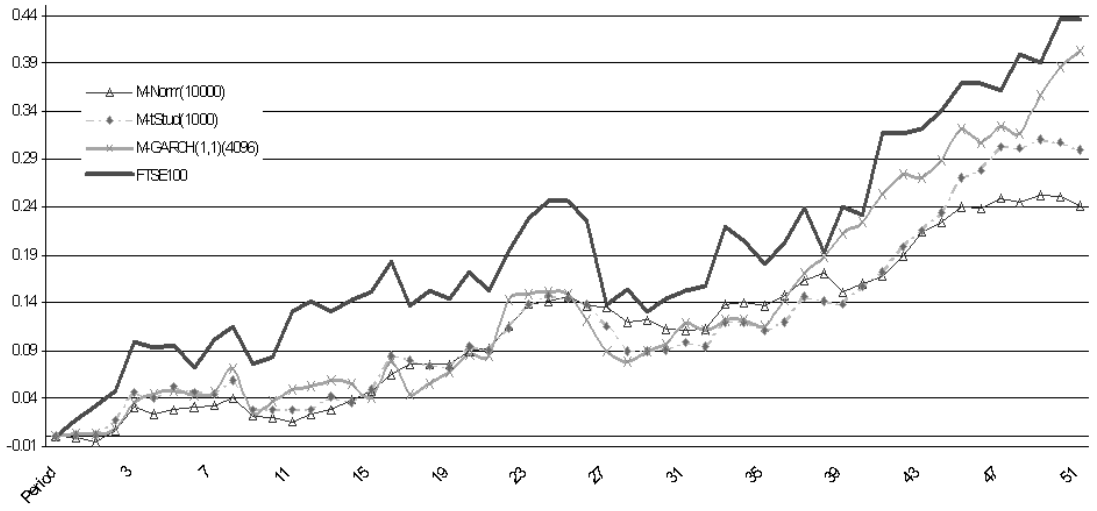

Figure 3.2: Cumulative returns (up-up data set): comparison between portfolio optimization models ($\mu_0 = 5\%$ and $\beta = 0.05$) and the market index.

the *down-up* figures), the differences with respect to other approaches are not too large, while in others (see Table 3.5 for the *up-down* figures) the downside risk incurred by using both the Monte Carlo generation techniques is significantly lower than that of all the other approaches. Nevertheless, such values are usually associated to an average rate of return very similar to that of portfolios obtained by applying other techniques and in at least one data set (see Table 3.5 for the *up-down* figures) even worse. In order to choose between the use of a multivariate Normal distribution with respect to a multivariate $t-$Student distribution in a Monte Carlo approach, one may notice that the portfolios selected by using the M-Norm(10000) technique often dominate those selected by using the M-tStud(10000) technique when the market is increasing in the out-of-sample period (see Table 3.5 for the *up-up* and the *down-up* figures). The reverse is true when the market is decreasing in the ex-post period (see Table 3.5 for the *up-down* and the *down-down* figures). This can be explained by considering that the heaviness of the tails, and thus the degrees of freedom of the $t-$Student distribution, is not constant over time. In particular, when the market trend is decreasing the tails are heavier than when the market is increasing, and the empirical probability distribution differentiates from the Normal one.

When all the tested data sets are considered, the best ex-post average rate of return is provided by the portfolios selected by using M-GARCH(1,1)(4096) as scenario generator. More precisely, in the *up-up* and in the *up-down* data sets the average rate of return associated to portfolios obtained with the M-GARCH(1,1)(4096) technique is always significantly greater than that of portfolios obtained by applying all the other techniques. The average rate of return in the *up-down* data set (see Table 3.5) is more than twice the average rate of return obtained by using the Hist. Data technique and it is almost 20% larger than the average of all the other techniques in the *up-up* data set. Unfortunately, the behavior is not confirmed in the remaining two data sets. In the *down-up* and in the *down-down* data set the portfolios selected by applying the M-GARCH(1,1)(4096) technique are dominated by all the other portfolios. Nevertheless, since the difference in terms of average rates

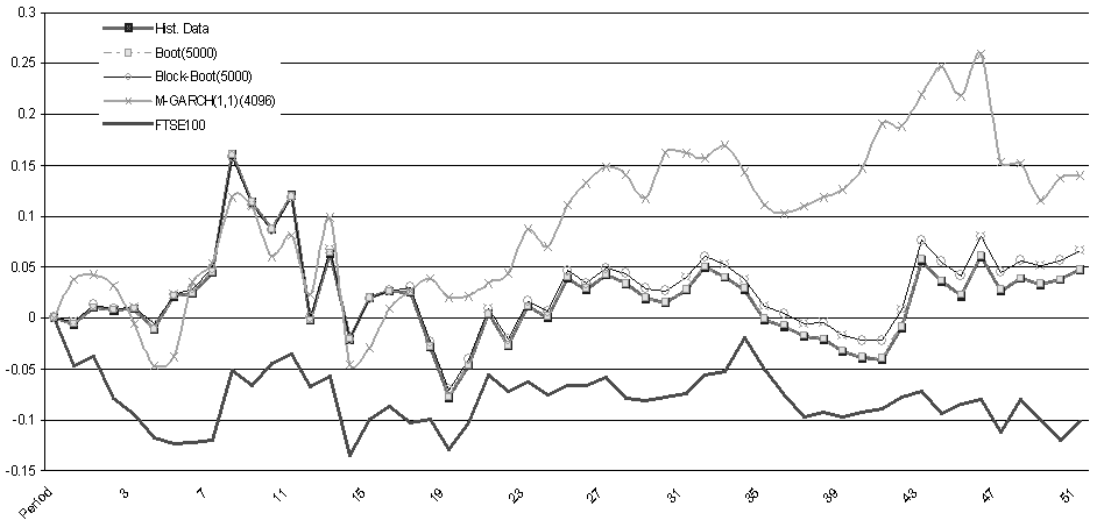

Figure 3.3: Cumulative returns (up-down data set): comparison between portfolio optimization models ($\mu_0 = 5\%$ and $\beta = 0.05$) and the market index.

of return in these two data sets between M-GARCH(1,1)(4096) and the other techniques is about 5% on yearly basis, and that the difference in the *up-up* and in the *up-down* data sets is significantly greater, a risk neutral investor may decide to choose the M-GARCH(1,1)(4096) technique in order to maximize the expected portfolio rate of return. Through a deep analysis of all the out-of-sample portfolio performances for all the values of μ_0 and β (see Guastaroba *et al.* [79]) one may notice that M-GARCH(1,1)(T) is the technique that quite often provides the most unstable portfolios. This can be also observed by looking at the std and the s-std values shown in Table 3.5 for the *up-up* and for the *up-down* data sets, and the s-std values computed in the *down-up* data set. In almost every data set there is at least a portfolio selected by the CVaR(β) model that, in terms of average rate of return, outperforms the market index. This is particularly true when the market index is decreasing in the ex-post data set. In all such cases the difference between the average rate of return associated to a portfolio selected by the optimization model and the average rate of return of the market index is consistent, confirming that portfolio optimization models represent a valuable tool for financial decisions. In the *down-down* data set all the portfolios selected by the optimization model yield a mean rate of return higher than that provided by the market index, which is at least 13% on yearly basis worse than that of the optimized portfolios. In the *down-up* data set (see Figure 3.4), the market index outperforms almost all the portfolios generated by the model by about 10% on yearly basis, but it is significantly outperformed by the portfolio obtained by using Block-Boot(1000) (the difference is about 36% on yearly basis, see Table 3.5). Only in the *up-up* data set the market obtains an average rate of return which is always greater than that reached by all the portfolios selected by the optimization model. Nevertheless, the difference using the M-GARCH(1,1)(4096) is on average lower than 10% and in some cases (see Table 3.5) it reduces to about 6% on yearly basis. Similar conclusions can be drawn by analyzing Figure 3.2. The market index has shown, on average, a cumulative return extremely unstable (see Figures 3.2–3.5), in particular if compared with the optimized portfolios. The higher volatility of the FTSE100 in-

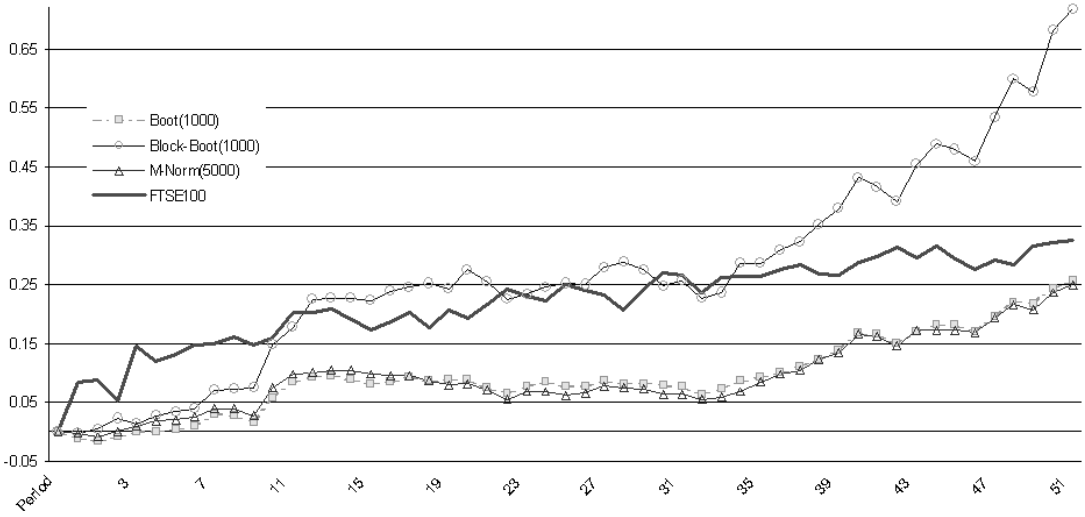

Figure 3.4: Cumulative returns (down-up data set): comparison between portfolio optimization models ($\mu_0 = 5\%$ and $\beta = 0.05$) and the market index.

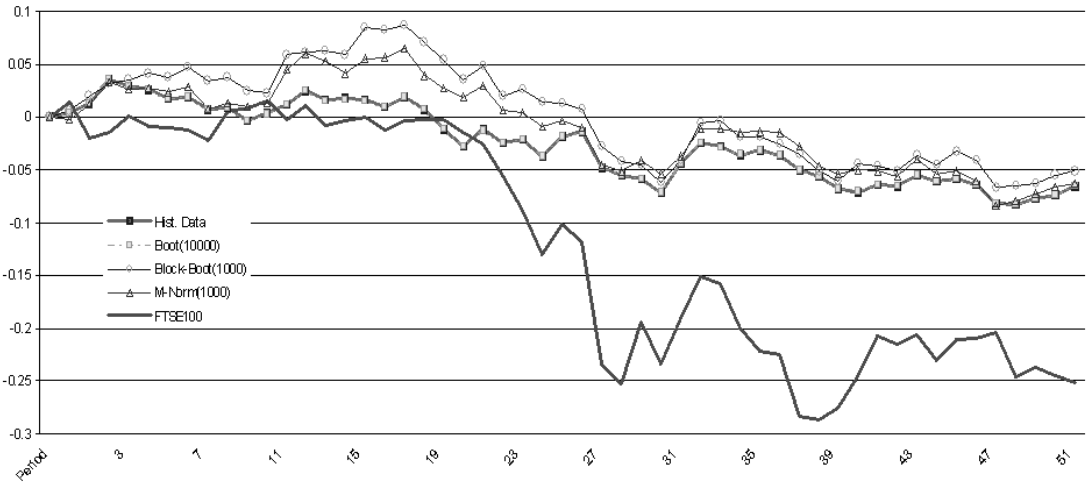

Figure 3.5: Cumulative returns (down-down data set): comparison between portfolio optimization models ($\mu_0 = 5\%$ and $\beta = 0.05$) and the market index.

dex ex-post returns is confirmed by the dispersion measures shown in Table 3.5. The s-std value is twice the values associated to almost all the portfolios selected by the model in the *up-up* and in the *down-up* data sets and to all the model portfolios in the *down-down* data set. Only in the *up-down* data set the market index shows a s-std value better than some portfolios selected by the CVaR(β) model. However, the lower downside risk is frequently associated to a worse value of the mean rate of return. To conclude, both portfolios and market index have, on average, similar fluctuations but the size of these fluctuations, and in particular of the decreasing ones, is significantly lower for the portfolios selected by the optimization model.
By summarizing all the above results, we can draw the following conclusions

1. independently of the scenario generation technique applied, the use of a portfolio optimization model represents a valuable tool for financial decisions. This is true when the market trend is increasing and becomes of crucial importance when the market trend is decreasing;

2. when comparing scenario generation techniques over different market trends, it is possible to conclude that the Block-Boot(1000) is the only technique that has shown good performance in terms of both mean return and downside risk, while the M-GARCH(1,1)(4096) technique is the best possible choice in order to maximize the expected portfolio rate of return;

3. the Hist. Data technique, despite its simplicity, generates portfolios with rates of return very close to those of the other optimized portfolios and with a downside risk incurred very similar to that of the portfolios selected by the other non-parametric approaches;

4. each of the analyzed techniques may require a different number of generated scenarios to be effectively used in an optimization model. In particular, computational results have shown that, on average, a well-defined size of the sample can be worked out for each scenario generation technique. Such size is equal to around 1000 scenarios for bootstrapping techniques, about 10000 scenarios for Monte Carlo simulation and no more than 5000 for M-GARCH technique.

3.5 Conclusions

In this research we have studied an optimal portfolio selection problem where scenarios for the rates of return are generated by means of three non-parametric and two parametric techniques. Specifically, we have solved a single-period portfolio optimization model with transaction costs and the Conditional Value-at-Risk as performance measure. Extensive computational results on real-life data from London Stock Exchange have allowed us to draw some interesting conclusions. The first aim of the research was to prove that the portfolio optimization model is a reliable tool to support financial decisions. The second aim was to find out whether there exists a technique to generate scenarios that outperforms all the others under different market trends. With respect to the first aim, we can conclude that portfolio optimization models represent a valuable tool for financial decisions providing good results when the market is increasing and becoming of crucial importance when the market is decreasing determining a strong reduction of the instability of the portfolio return. The effectiveness of portfolio optimization models is confirmed by observing that, in almost all the possible market behaviors, there is at least a portfolio selected by means

of the portfolio optimization model that significantly outperforms the market index. With respect to the second aim, if the investor is risk averse and would like to hedge his investment from a possible dramatic fall in market quotation during the ex-post period, the Block-Boot(1000) is the best technique to use. If the investor is, on the contrary, more risk seeking in order to maximize his portfolio expected return, the use of the M-GARCH(1,1)(4096) technique is the suggested choice. Finally, the Hist. Data approach, the most widely used technique in practice and in academic research, considering its simplicity and the quality of the portfolios selected, can be a sensible choice.

Appendix

MGARCH technique

Let $\mathbf{x_t}$ be the time series vector of the rates of return of n securities at time t. The MGARCH technique is based on the following relations

$$\mathbf{x_t} = \mathbb{E}\{\mathbf{x_t}|\psi_{\mathbf{t-1}}\} + \epsilon_\mathbf{t} \tag{3.9}$$

and

$$Var(\epsilon_\mathbf{t}|\psi_{t-1}) = \mathbf{H_t},$$

where ψ_{t-1} represents the σ-field generated by all the information available until time $t-1$ and $\epsilon_\mathbf{t}$ is the vector of residuals, see Bollerslev [23]. $\mathbf{H_t}$ is the time varying conditional covariance matrix that Bollerslev [23] suggests to compute as follows

$$\mathbf{H_t} = \mathbf{D_t R D_t}, \tag{3.10}$$

where $\mathbf{R}$ is a time invariant correlation matrix containing the unconditional correlations, $\mathbf{D_t} = diag\{\sqrt{h_{jt}}\}$ and h_{jt} is the conditional variance of security j at time t, $j = 1, ..., n$. In details, the diagonal matrix $\mathbf{D_t}$ containing the conditional standard deviation can be generated estimating n univariate GARCH models. The correlation matrix constant over time $\mathbf{R}$ can be simply estimated as the unconditional correlation matrix of the standardized residuals, whereas the vector of residuals $\epsilon_\mathbf{t}$ is computed as $\epsilon_\mathbf{t} = \xi_\mathbf{t}\mathbf{C_t}$, where $\mathbf{C_t C_t^T} = \mathbf{H_t}$ and $\xi_\mathbf{t}$ is distributed as a white noise, i.e. $\xi_\mathbf{t} \sim \mathbf{WN(0, 1)}$.
To forecast the following period, equation (3.10) can be rewritten as follows

$$\mathbf{H_{t+1}} = \mathbf{D_{t+1} R D_{t+1}} \tag{3.11}$$

where

$$h_{jt+1} = \alpha_{0j} + \alpha_{1j}\epsilon_{jt}^2 + \beta_{1j}h_{jt}. \tag{3.12}$$

To use CC-MGARCH model for scenario generation one has to forecast the time varying conditional covariance matrix after one period by using (3.11) and (3.12). Then, after performing the Cholesky's factorization, s scenarios using (3.9) one period after are generated, where in $\epsilon_\mathbf{t+1}$ the matrix $\xi_{\mathbf{t+1,k}}$, $k = 1, ..., s$, is constructed by randomly sampling n values s times from the historical standardized residuals. In this way, one may generate s time series vectors $\mathbf{x_{t+1,k}}$, $k = 1, ..., s$, representing s forecasted outcomes of the multivariate random variable $\mathbf{x_t}$. These s values represent the nodes at stage $t+1$ in the event tree. The nodes at stage $t+2$ can be computed recursively by forecasting one period after the time varying conditional variance by considering the realizations occurred in the parent nodes.

283. On the Effectiveness of Scenario Generation Techniques in Single-Period Portfolio Optimization

4

Effectiveness of Robust Portfolio Optimization Techniques

4.1 Introduction

In portfolio optimization the investor has to decide the best portfolio composition choosing among assets available in the financial market. In its most general form, a single-period portfolio optimization model can be cast into the following mean-risk bicriteria optimization model [111]

$$\begin{aligned} \min \quad & [\varrho(\mathbf{x}), -\tilde{r}_{\mathbf{x}}] \\ & \mathbf{x} \in X. \end{aligned} \tag{4.1}$$

In problem (4.1), function $\varrho(\mathbf{x})$ to be minimized measures the risk of the portfolio, whereas function $\tilde{r}_{\mathbf{x}} = \tilde{r}^T\mathbf{x}$ to be maximized measures the uncertain return of the portfolio.
The approach frequently adopted in the literature to tackle this kind of problems is the bounding approach, i.e. to keep the minimization of the risk as objective function and constrain the portfolio return. Therefore, model (4.1) is usually reformulated as follows

$$\begin{aligned} \min \quad & \varrho(\mathbf{x}) \\ & \tilde{r}^T\mathbf{x} \geq \mu_0 \\ & \mathbf{x} \in X \end{aligned} \tag{4.2}$$

where the parameter μ_0 is the minimum expected portfolio rate of return accepted by the investor. Notice that an uncertain element in model (4.2) is represented by the future realizations of vector $\tilde{r}$. This implies that the corresponding constraint could be violated for several possible realizations of the random vector $\tilde{r}$.
The Mean-Variance portfolio formulation introduced by Markowitz [117] suggests finding an asset allocation that results in the minimum risk portfolio, as measured by the portfolio variance, for a given level of target portfolio return. The Mean-Variance model can then be interpreted as an optimization model in which the uncertain parameters are substituted by their *nominal values* (represented by their means) and the uncertainty is captured by the portfolio variance. Therefore, vector $\tilde{r}$ is replaced by vector $\bar{r} = [\bar{r}_j]$ $(j = 1, \ldots, n)$, and the risk is measured by function $\varrho(\mathbf{x}) = \sigma^2(\mathbf{x}) = \mathbf{x}^T\Sigma\mathbf{x}$.
The classical approach used in single-period portfolio optimization, and in mathematical programming as well, is to design and solve a mathematical model assuming that the decision-maker has

certain knowledge about the uncertain parameters. A strategy frequently suggested to address this kind of uncertainty is based on the use of stochastic optimization. However, stochastic optimization models usually imply the exact knowledge of the probability distribution function of the random parameters, and they are often computationally very hard to solve.
Recently, another approach to deal with uncertain data has been proposed. The approach is referred to as *robust optimization*. Robust optimization overcomes most of the drawbacks implied by stochastic optimization since the only input data needed are the mean and the range of variability of the underlying data, instead of the exact probability distribution function. Moreover, most of the robust optimization approaches are computationally tractable and allow the decision-maker to control the level of conservatism of the solution.
As most of the research on robust optimization offers theoretical contributions, the pursuit of empirical evidence on the effectiveness of robust optimization techniques has motivated us to undertake this study in a context, such as portfolio optimization, where the problem by its nature involves uncertain parameters. The main objective of the research reported in this chapter is, therefore, to evaluate the effectiveness of robust optimization techniques when applied to the problem of portfolio selection. To the sake of homogeneity with the rest of the thesis, in this chapter we assume the CVaR(β) as performance measure.
Structure of the Chapter. In Section 4.2 we first summarize the main contributions appeared in the literature under the name of robust optimization and to alternative approaches dealing with uncertainty. Subsequently, we introduce the two CVaR(β) robust formulations we considered. Section 4.3 deals with the experimental analysis and the comparison of the optimal portfolios selected by the two CVaR(β) robust counterparts and by the nominal model. Several historical data sets are used to span different market trends. Some concluding remarks can be found in Section 4.4.

4.2 Robust Portfolio Optimization

4.2.1 Dealing with Uncertainty: Literature Review

The classical approach used in mathematical programming is to construct and process, i.e. solve, a model making the assumption that the descriptors, i.e. parameters, are known with certainty. Deterministic mathematical programming models by their nature do not take into consideration parameter uncertainty on the optimality and feasibility of the solution. However, in many decision problems where such deterministic assumptions do not hold as some of the data take values different than expected, the optimal solution (decision) vector found might be sub-optimal or even infeasible. To underline this problem we quote from Ben-Tal and Nemirovski [12]

> "In real-world applications of Linear Programming, one cannot ignore the possibility that a small uncertainty in the data (intrinsic for most real-world LP programs) can make the usual optimal solution completely meaningless from practical viewpoint."

The occurrence of such problematic effects has motivated the design of solution techniques that are comparatively immune to data uncertainty. These techniques are called "*robust*".
To the best of our knowledge, Soyster [143] made the first attempt to deal with data uncertainty in mathematical programming. Soyster considers the case of "column-wise" uncertainty proposing

a robust linear formulation, and his solution approach is often criticized for its excessive conservatism (see, among all, [11]). For further developments of Soyster's approach, see [66] and [141]. The focus on robust optimization was brought under limelight by Ben-Tal and Nemirovski [10], [11], [12]; a similar approach was also independently developed by El-Ghaoui and Lebret [57] and El-Ghaoui *et al.* [58]. The authors, see for instance Ben-Tal and Nemirovski [10], propose several formulations and applications and show that if the uncertainty set is ellipsoidal, then the robust counterpart of some important convex optimization problems is either an exactly or an approximately tractable problem that is efficiently solvable via, for instance, an interior point method. In details, if the uncertain parameters belong to ellipsoidal uncertainty sets the most unlikely realizations are not considered, and the robust counterpart of a linear programming problem (LP) becomes a second-order cone problem (SOCP), of a SOCP becomes a semidefinite programming problem (SDP), whereas of a SDP is NP-hard to solve. Hence, the computational complexity of the robust counterpart is greater than that of the original problem choosing an ellipsoidal uncertainty set. More recently, Bertsimas and Sim [14] show that, under some assumptions on the values that the uncertain parameters are allowed to take, the robust counterpart of an LP problem is also an LP problem. In Bertsimas *et al.* [15] the authors propose a framework for robust modeling of linear programming problems using uncertainty sets described by an arbitrary norm.
The robust problem is usually formulated as a single, comprehensive convex optimization problem. Some authors use another approach. For instance, Bienstock [19] assumes two types of uncertainty sets. The first type allows the modeler to specify rough frequencies of return shortfall as a function of the magnitude of the deviation; the second type models correlation among deviations and gives rise to VaR (Value-at-Risk) and CVaR problems. The author then proposes a cutting plane algorithm that runs a sequence of two-step iterations. In the first step the algorithm solves an implementor problem which picks up values for the decision variables, while in the second step solves an adversarial problem which finds the worst-case data corresponding to the decision variables selected by the implementor problem.
Usually, in robust optimization literature authors assume that the convex set describing the uncertainty is given. Indeed, as pointed out in Bertsimas and Thiele [17], the only information available to the decision maker is historical realizations of the random variables, and historical data *must be* incorporated into the uncertainty set. Bertsimas and Thiele [17] illustrate the robust and data-driven optimization framework as a modeling tool in decision-making under uncertainty using historical realizations of the random variables as direct inputs to the mathematical programming problem. Bertsimas and Thiele [16] present an application of data-driven optimization to inventory management.
The potential benefits deriving from the use of a robust optimization methodology applied to portfolio selection problems are remarkable. Indeed, every portfolio optimization model deals with the estimate of the portfolio rate of return and of either a risk or a safety measure. As pointed out by Black and Litterman [20] for the classical Mean-Variance model, the portfolio decision is very sensitive to the means and the covariance matrix changes of the asset universe. They show that a small change in the mean can produce a large change in the optimal portfolio composition.
To highlight this problem we report the following example from Ceria and Stubbs [32]. Suppose a market with two assets and an investor who wants to maximize the expected return subject to a budget constraint that forces the full investment between the two assets, and a constraint that limits the portfolio risk to be no more than 10 per cent with respect to the benchmark portfolio (shown as point "M" in Figure 4.1). The feasible region of this example is illustrated in Figure 4.1 as the

intersection of the shaded ellipsoidal region and the budget constraint, i.e. the feasible region of this example is simply the line segment between the two points A and B. Suppose that asset 1 has an expected return equal to 2.4 per cent and a standard deviation equal to 0.42, whereas the same parameters for asset 2 are equal to 2.5 and 0.33, respectively. Assume that the correlation between the two assets is 0.7. Using these values, the optimal portfolio composition is at point A in Figure 4.1. Let us now assume that the expected rates of return for assets 1 and 2 are equal to 2.5 and 2.4, respectively. Using these values, the optimal portfolio is at point B. To conclude, a very small change in the data estimates of the model determines a dramatic change in the optimal portfolio composition. From this example, it is clear why practitioners often find "optimized" portfolios to be counter intuitive and impractical.

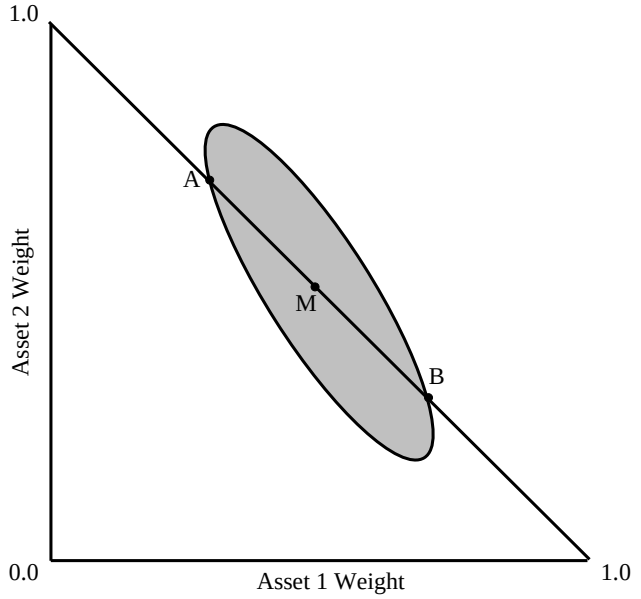

Figure 4.1: Feasible region from the example in Ceria and Stubbs [32].

Lobo *et al.* [107] recast the Markowitz optimization problem as a SOCP problem using robust optimization techniques. Ben-Tal and Nemirovski [11] are the first authors to suggest replacing an uncertain LP problem with its robust counterpart and illustrate their robust methodology with a simple portfolio selection problem. Ben-Tal *et al.* [9] apply the robust counterpart approach to multi-stage asset allocation problem and compare it with a multi-stage stochastic programming model. Costa and Paiva [44] study the optimal portfolio selection problem for tracking error when the expected returns of the risky and risk-free assets, as well as the covariance matrix, are assumed to belong to a convex polytope defined by some known elements. El-Ghaoui *et al.* [59] investigate the robust portfolio optimization problem using worst-case VaR as risk measure, when only partial information about the distribution is known. Goldfarb and Iyengar [74] formulate the robust counterpart of the Mean-Variance portfolio selection problem, of the maximum Sharpe ratio portfolio selection problem, and of the Value-at-Risk portfolio selection problem introducing ellipsoidal uncertainty sets. Additionally, they show that for natural classes of uncertainty sets all robust portfolio allocation problems are SOCP. Bertsimas and Sim [14] apply their approach to a portfolio optimization problem reporting numerical results. Tütüncü and Koenig [146] present an algorithm for generating robust efficient portfolios using a variant of an interior-point method

for saddle-point problems, and discuss an implementation of the algorithm. Zhu and Fukushima [152] introduce the concept of worst-case CVaR and present the application of it to robust portfolio optimization, together with some numerical examples. Natarajan *et al.* [123] propose an approximation method for minimizing the VaR of a portfolio based on robust optimization techniques. The method results in the optimization of a modified VaR measure, called Asymmetry-Robust VaR, that takes into consideration asymmetries in the distributions of returns and that is coherent. Bienstock [19] applies the already mentioned cutting plane algorithm to the robust counterpart of the Mean-Variance problem assuming two uncertainty models for shortfall in asset returns. Fabozzi *et al.* [64] discuss some of the major trends and innovations in the management of financial portfolios today, focusing on state-of-the-art robust methodologies for portfolio risk and return estimation, optimization, trading, and general management. Particularly, they provide an overview on the major approaches for optimization under uncertainty, including robust optimization, and claiming that robust optimization "...has shown a great potential and usability for portfolio management and optimization applications". The same authors explain the main ideas behind the robust optimization approach, and discuss the relation between robust optimization and other robust methods for portfolio management in [65]. In the latter paper, the authors also describe some of the latest developments in robust optimization applications, and discuss future directions in robust portfolio management.

An excellent introduction to the theory of robust optimization along with examples of robust optimization formulations for some financial optimization problems can be found in the book by Cornuejols and Tütüncü [43].

Other authors use a different approach to deal with the uncertainty, substituting the concept of uncertainty set with the one of *ambiguity set*. The key point of this approach is to consider the probability distribution function of the random variables to be imprecisely known. As defined in Ellsberg [60] we refer today to an uncertain problem if the probabilistic model is known but the outcomes of the random variables are unknown, whereas we refer to an ambiguity problem if both the probabilistic model and the realizations of the random variables are unknown. Modeling ambiguity has received attention in several different fields, e.g., economics, finance and stochastic programming. The approach consists of replacing the basic model by its ambiguity extension. To this end, the decision maker has to define an ambiguity set, i.e. a set of probability distribution to which the modeler is indifferent. Once defined a nominal distribution, for example inferring from historical data, every probability distribution function that differs in distance from the nominal one no more than some ϵ belongs to the ambiguity set. Therefore, the ambiguity extension results in a combination of a robust and a stochastic problem: while assuming that the outcomes of the random variables come from an uncertain probability distribution function, the model allows to vary the distribution within the ambiguity set (see [131]). Recently, Erdŏgan and Iyengar [63] studied the ambiguous chance constrained problem, employing the Prohorov distance to describe the ambiguity set. An application of the ambiguity approach to the portfolio optimization problem can be found in Pflug and Wozabal [131]. The authors use the Kantorovich distance to define the set of admissible distributions and their approach is "robust" in the sense that the selected portfolio is slightly sub-optimal for the given nominal probability distribution, but performs also well under neighboring distribution functions. They use as risk measure the function (1 - A), where A is the average Value-at-Risk. Calafiore [28] adopts a similar approach employing the Kullback-Leibler divergence function as distance measure among distributions, and develops solution methods to compute the worst-case optimal portfolios using the variance and the absolute deviation as risk

measures.
Other approaches incorporate uncertainty directly into the computation of the optimal solution. Stochastic programming methods, for instance, are based on the generation of several scenarios representing the uncertain data. Traditional stochastic linear programming models find the optimal solution with the best average objective function value over all the scenarios. More advanced stochastic programming methods include risk considerations such as penalties for constraint violation and probabilistic guarantees. Particularly, under the assumption that only the objective function is affected by the uncertainty, the simplest way to make the objective well-defined is to optimize it on average, where the expectation is computed over the uncertain data. On the other hand, if the uncertainty is present in the constraints, one can formulate such problems by introducing penalties for the violation, thus defining a mean-risk objective function (see, for instance [121]). An alternative approach is to require that the constraints are satisfied for all possible (in particular, the worst-case) values of the uncertain parameters. (It is worth highlighting that this is a stochastic programming method whose philosophy overlaps with the philosophy of robust optimization.) Finally, one may impose the requirement that the constraints are satisfied with a high probability. This leads to the so-called stochastic programming formulation with chance constraints. For a survey of stochastic programming applications in financial optimization see [88] and references therein.
It is worth citing [83] where the authors present three approaches for generating scenario trees for financial portfolio problems. These methods are based on simulation, optimization and hybrid simulation/optimization. In the simulation approach, scenarios are clustered from simulations generated sequentially or in parallel, while in the optimization approach scenarios are obtained by solving a non-linear optimization problem at each node or a large non-linear programming problem. In the hybrid approach, the optimization problem is reduced in size by fixing price variables to values obtained by simulation. Typically, in order to estimate random parameters one would regress against historical data. As with many estimation techniques based on historical data, the quality of the estimates or forecasts are therefore critically dependent on the applicability of the given data set for forecasting over the given period, which is difficult to determine. Therefore, the imprecise nature of the moment forecasts needs to be tackled to reduce the risk of decision-making on the wrong scenario. The mis-forecasting problem has been investigated through forecast pooling. See, between all, Lawrence *et al.* [101] that empirically examine the improvement in accuracy which can be gained from combining judgmental forecasts either with other judgmental or with quantitatively derived forecasts. The pooling has been often achieved using stochastic programming approaches where the probability of each scenario is considered and their weighted average is evaluated (e.g., see Makridakis and Winkler [109]). In the domain of stochastic programming, several authors claim that there are certain situations where the worst-case scenarios might have a disastrous impact on the portfolio should they materialize. Sometimes one might not even know which scenario is the worst-case. One technique to assess the worst-case effect is the so-called minimax approach, where the decision-maker minimize (maximize) the objective function with respect to the worst possible outcome of the uncertain parameters. The advantage of worst-case analysis is to provide a guaranteed performance that will improve if any scenario other than the worst-case occurs. The application of worst-case decisions to portfolio optimization in finance have been investigated by a number of researchers (e.g., see Gulpinar and Rustem [82]). In the book by Rustem and Howe [137] concepts and algorithms for computing the best decision in view of the worst-case scenario are provided, along with examples of applications to financial problems.

Dynamic programming methods are designed to deal with stochastic uncertain systems over multiple stages. It can be used to address similar types of problems as multi-stage stochastic programming, but the philosophy of solving problems is different. Most generally, instead of considering one large optimization problem, in dynamic programming one solves multiple optimization problems sequentially, starting at the last stage and proceeding backwards, thus keeping track only of the optimal paths from any given time period onwards. Unfortunately, they suffer from the curse of dimensionality: The problem size increases exponentially as new scenarios or states are added. The differences between dynamic programming and stochastic programming methodologies are not clear cut, however, because dynamic programming-type methodologies are used in some decomposition algorithms for solving stochastic programming problems. An excellent description of dynamic programming with applications in engineering, finance and operations research is provided in the book by Bertsekas [13].
In the following section we provide more information about the two robust approaches analyzed and tested in this chapter.

4.2.2 The CVaR(β) Robust Formulations

Consider problem (4.2) and assume that all, and only, the components $\tilde{r}_j$ of vector $\tilde{r}$ are affected by the uncertainty (an analogous choice has been done by Ceria and Stubbs [32]). Furthermore, assume that the objective function $\varrho : \mathcal{R}^n \to \mathcal{R}$ to be minimized is linear. No underlying probabilistic model of the uncertain data is assumed to be known although the uncertainty is supposed to be *unknown-but-bounded*, i.e. $\tilde{r}$ belongs to a given convex and closed *uncertainty set* $\mathcal{U}$. Moreover, the constraints are hard, i.e. they must be satisfied for all the meaningful actual realizations of the random parameters. Under these assumptions, a given point $\mathbf{x}^*$ is a feasible solution of problem (4.2) if and only if for every possible realization of the random parameters the constraints are satisfied. Thereby, we can consider the following problem

$$\begin{aligned} \min \quad & \varrho(\mathbf{x}) \\ & \tilde{r}^T\mathbf{x} \geq \mu_0 \\ & \tilde{r} \in \mathcal{U} \\ & \mathbf{x} \in X. \end{aligned} \tag{4.3}$$

Problem (4.3) is called the *robust counterpart* of the uncertain LP problem (4.2) (see [11] for a formal definition of robust counterpart).
Consider the uncertain return $\tilde{r}_j$, $j = 1, \ldots, n$, as a symmetric and bounded random variable with nominal value equal to its mean $\bar{r}_j$, and that takes values in the interval $[\bar{r}_j - \sigma_j, \bar{r}_j + \sigma_j]$. Ben-Tal and Nemirovski [11] assume the former model of data uncertainty and point out that when applying the robust counterpart approach the decision-maker is not obliged to include in the uncertainty set all which may happen. In [11] they introduce a simple portfolio optimization problem, and suggest the use of ellipsoidal uncertainty sets, i.e. sets given by convex quadratic inequalities

$$\mathcal{U}_\theta(\tilde{r}) := \{r \in \mathcal{R}^n | \sum_{j=1}^{n} \sigma_j^{-2}(r_j - \bar{r}_j)^2 \leq \theta^2\}.$$

The value of the parameter θ is chosen by the decision-maker and reflects her/his attitude toward risk: The larger the value of θ, the more risk averse the decision-maker is. Therefore, uncertainty sets of this shape allow the decision-maker to adjust the level of conservatism of the robust solutions. Under the former model of data uncertainty and assuming ellipsoidal uncertainty sets the robust counterpart of problem (4.2) is the following non-linear problem

$$\begin{aligned} \min \quad & \varrho(\mathbf{x}) \\ & \bar{r}^T\mathbf{x} - \theta\hat{V}(\mathbf{x})^{\frac{1}{2}} \geq \mu_0 \\ & \mathbf{x} \in X, \end{aligned} \tag{4.4}$$

where $\hat{V}(\mathbf{x}) = \sum_{j=1}^{n} \sigma_j^2 x_j^2$, and vector $\bar{r} = [\bar{r}_j]$ $(j = 1, \ldots, n)$ is the vector containing the nominal values. A valuable advantage of this robust approach is that the probability of constraint violation is bounded. Considering our application, the probability bound is the following: $\mathbb{P}(\tilde{r}^T\mathbf{x} < \mu_0) \leq \exp\{-\theta^2/2\}$ (see [12] for a proof). The main practical drawback of this approach is that it gives rise to a second-order cone programming problem, that is more computationally expensive than an LP model. In fact, despite polynomial-time algorithms like interior point methods can be used to solve an SOCP, the benefits deriving from the use of a linear formulation are evident when solving large-scale problems including real-life features, e.g. formulations including binary variables needed to model the payment of fixed transaction costs.

Bertsimas and Sim [15] assume the same model of data uncertainty and propose a robust formulation that allows to flexibly adjust the level of conservatism of the solution. They introduce the parameter $\Gamma \in [0, n]$, not necessarily integer, that adjusts the robustness of the method against the level of conservatism of the solution. Actually, the parameter Γ is called *budget of uncertainty* and, for a general optimization problem, has to be chosen for each constraint involving uncertain parameters. Considering our problem, the approach guarantees deterministically the feasibility of the robust solution if at most $\lfloor\Gamma\rfloor$ of the uncertain coefficients $\tilde{r}_j$ change from their nominal value and another uncertain coefficient $\tilde{r}_{\hat{j}}$ changes by $(\Gamma - \lfloor\Gamma\rfloor)\sigma_{\hat{j}}$. Applying the Bertsimas and Sim approach, the robust counterpart of problem (4.2) turns out to be the following non-linear problem

$$\begin{aligned} \min \quad & \varrho(\mathbf{x}) \\ & \bar{r}^T\mathbf{x} - \beta(\mathbf{x}, \Gamma) \geq \mu_0 \\ & \mathbf{x} \in X, \end{aligned} \tag{4.5}$$

where the *protection function* $\beta(\mathbf{x}, \Gamma)$ takes the following form

$$\beta(\mathbf{x}, \Gamma) = \max_{\{S \cup \{\hat{j}\} | S \subseteq N, |S| = \lfloor\Gamma\rfloor, \hat{j} \in N \setminus S\}} \{\sum_{j \in S} \sigma_j x_j + (\Gamma - \lfloor\Gamma\rfloor)\sigma_{\hat{j}} x_{\hat{j}}\}. \tag{4.6}$$

It is worth noting that given a vector $\mathbf{x}^*$, the protection function (4.6) equals the objective function of the following linear optimization problem

$$\beta(\mathbf{x}^*, \Gamma) = \max \sum_{j=1}^{n} (\sigma_j x_j^*) w_j \tag{4.7}$$

$$\sum_{j=1}^{n} w_j \leq \Gamma \tag{4.8}$$

$$w_j \leq 1 \quad j = 1, \ldots, n \tag{4.9}$$

$$w_j \geq 0 \quad j = 1, \ldots, n. \tag{4.10}$$

The dual of problem (4.7)-(4.10) is the following LP model

$$\begin{aligned} \beta(\mathbf{x}^*, \Gamma) = \min \quad & z\Gamma + \sum_{j=1}^{n} p_j \\ z + p_j \geq \sigma_j x_j^* \quad & j = 1, \ldots, n \\ p_j \geq 0 \quad & j = 1, \ldots, n \\ z \geq & 0, \end{aligned} \tag{4.11}$$

where z is the dual variable corresponding to constraint (4.8), whereas variables p_j are the dual variables associated to each of constraints (4.9). Since primal problem (4.7) is feasible and bounded for all values of $\Gamma \in [0, n]$, by the strong duality theorem the dual problem (4.11) is also feasible and bounded, and the optimal objective function value of the two problems coincide. Therefore, one can substitute in problem (4.5) the objective function of problem (4.11) and add the relative constraints. In the end, the robust counterpart of problem (4.2) can be cast as the following LP problem

$$\min \quad \varrho(\mathbf{x})$$

$$\mathbf{x}^T \mu - (z\Gamma + \sum_{j=1}^{n} p_j) \geq \mu_0$$

$$z + p_j \geq \sigma_j x_j \quad j = 1, \ldots, n \tag{4.12}$$

$$p_j \geq 0 \quad j = 1, \ldots, n \tag{4.13}$$

$$z \geq 0 \tag{4.14}$$

$$\mathbf{x} \in X.$$

The authors show several probability bounds of constraint violation. One of these is $\mathbb{P}(\tilde{r}^T \mathbf{x} < \mu_0) \leq 1 - \Phi(\frac{\Gamma - 1}{\sqrt{n}})$, where $\Phi(\cdot)$ is the cumulative distribution function of a standard normal distribution (see [15] for a proof).

We assume the CVaR(β) optimization model (2.6)-(2.11) as *nominal problem*. We will call *nominal portfolio* the optimal portfolio selected solving the nominal problem.

Assuming the Ben-Tal and Nemirovski approach, the CVaR(β) robust formulation is the following SOCP problem, by simplicity referred to as BN-CVaR(β) model

BN-CVaR(β) Model

(2.6)

$$\sum_{j=1}^{n} \mu_j x_j - \theta \hat{V}(\mathbf{x})^{\frac{1}{2}} \geq \mu_0 \tag{4.15}$$

(2.7) and (2.9)-(2.11).

On the other side, the Bertsimas and Sim approach gives rise to the following CVaR(β) robust counterpart, henceforth referred to as BS-CVaR(β) model

BS-CVaR(β) Model

(2.6)

$$\sum_{j=1}^{n} \mu_j x_j - (z\Gamma + \sum_{j=1}^{n} p_j) \geq \mu_0 \tag{4.16}$$

(4.12)-(4.14), (2.7) and (2.9)-(2.11).

4.3 Experimental Analysis

In this section we compare the portfolios selected by the BN-CVaR(β) and BS-CVaR(β) models. Computational experiments have been conducted on a PC with a 2.89 GHz Athlon Dual Core processor and 3.4 Gb of RAM. The models have been implemented in C++ by means of Concert Technology 2.3 and solved with CPLEX 10.1. We first present the testing environment, then the results of the in-sample and finally those of the out-of-sample analysis.

4.3.1 Testing Environment

Historical data are represented by weekly rates of return, computed by using stock prices taken from the London Stock Exchange (FTSE), the largest stock market in Europe. The rate of return of security j at the end of week t has been computed as $r_{jt} = \frac{q_{jt}-q_{jt-1}}{q_{jt-1}}$, where q_{jt} is the closing price of security j at the end of week t. No dividends have been considered.
We have tested the models under different market behaviors. We have constructed four data sets corresponding to different in-sample and out-of-sample time periods. Each data set temporal positioning is shown in Figure 4.2. The first data set is characterized by an increasing market trend in the in-sample period as well as in the out-of-sample period (*up-up data set*), the second data set by an increasing trend in the in-sample period and by a decreasing one in the out-of-sample period (*up-down data set*), the third data set by a decreasing trend in the in-sample period and by an increasing one in the out-of-sample period (*down-up data set*) and, finally, the last set by a decreasing trend in both the in-sample and the out-of-sample periods (*down-down data set*). Observing Figure 4.2 one may notice that the end of the down-up in-sample period does not coincide with the negative peak. The choice of a different testing period is motivated by the infeasibility of both the CVaR(β) robust counterparts when asked to find a robust portfolio with a minimum rate of return at least non-negative.
Each of these data sets consists of 2 years of in-sample weekly observations (104 realizations) and 1 year of out-of-sample ones (52 realizations). We have considered the 100 securities composing the FTSE100 Index at the date of September 25th, 2005. The FTSE100 Index is the capitalization-weighted index of the 100 most capitalized companies traded on the London Stock Exchange, representing approximately about 80% of the entire UK market. It is recognized as the benchmark for all UK financial markets.

All the computational experiments have been run using an enlarged data set derived from the

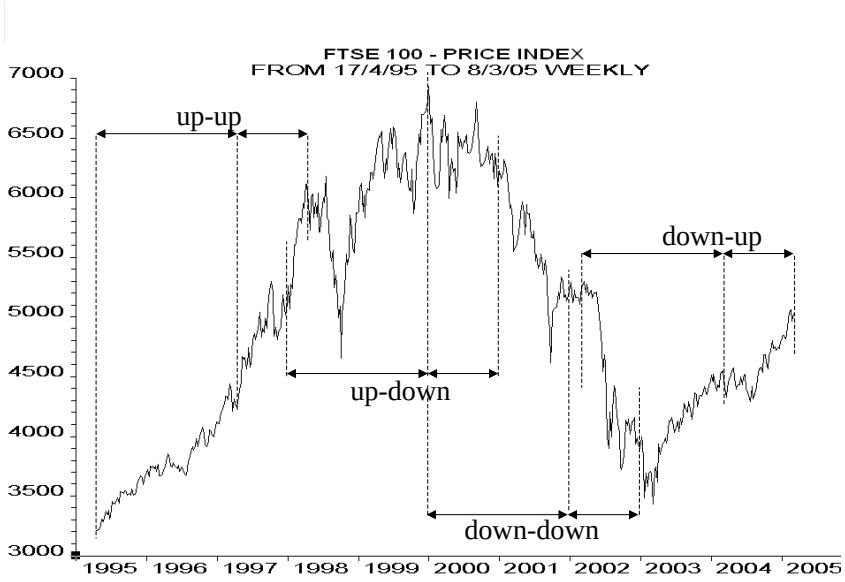

Figure 4.2: The four different market periods.

historical one. In particular, to each historical data set we applied a *bootstrapping* technique that samples with replacement 1000 times, i.e. we obtained four final data sets each of them composed by 1000 scenarios. In our application, one scenario represents an outcome of the random vector $\tilde{r} \in \mathcal{R}^{100}$ expressed on yearly basis, i.e. a correlated realization of the returns yielded by all the securities. In more details, we have computed one scenario in the following way. We sampled with replacement from the historical data set 52 samples. A sample corresponds to the joint realization of the rates of return for all securities as observed in a given time period. This allows us to preserve the historical correlation among securities. Subsequently, we computed the mean of these 52 samples and converted the mean from weekly to yearly basis using the equivalent rates formula. The procedure is repeated until 1000 scenarios are computed. Additionally, for each security, the average return and the standard deviation are computed over the 1000 scenarios (yearly returns). We refer to Efron and Tibshirani [56] for an explanation of the methodology. See Chapter 3 and [81] for an application of bootstrapping to portfolio optimization.
The former choice is motivated by the fact that no security composing our four data sets shows a standard deviation greater than the corresponding in-sample average weekly return. This implies that even for a low level of robustness, e.g. $\theta = 1.17$ and $\Gamma = 1$ corresponding approximately to a probability of constraint violation equal to 51%, the feasible domains of the two CVaR(β) robust formulations are empty sets. After running the bootstrapping, some securities show an average return greater than the standard deviation (both now computed on yearly basis). Furthermore, it is worth pointing out that if one sorts the list of securities with respect to the average return, or w.r.t. the standard deviation, before and after performing the bootstrap the two sorted lists are very similar, i.e. the bootstrapping technique does not modify heavily the order of the securities.
For each data set we solved the BN-CVaR(β) and BS-CVaR(β) models by considering four different levels of robustness. In particular, we solved the BN-CVaR(β) model choosing a value of θ equal to 0, 1.9187, 2.161, and 3.1, respectively. On the other hand, we solved the BS-CVaR(β)

formulation choosing a value of the budget of uncertainty Γ equal to 0, 11, 14, and 25, respectively. It is worth noting that solving both the previous models choosing a value of θ or Γ equal to 0 means solving the nominal problem. On the other hand, solving the robust counterparts assuming a value of $\theta = 1.9187$, as well as $\Gamma = 11$, corresponds to a probability bound of constraint violation, as introduced in Section 4.2.2, less than 16%. Assuming a value of $\theta = 2.161$, or equivalently $\Gamma = 14$, corresponds to a probability bound of constraint violation less than 10%. Finally, choosing a value of $\theta = 3.1$, or equivalently $\Gamma = 25$, corresponds to a probability bound of constraint violation less than 1%. We compare the portfolios selected using the two approaches by guaranteeing a similar probability of constraint violation.
For each data set we solved the two robust CVaR(β) formulations choosing six different values of the minimum required rate of return μ_0. We selected the six values as follows. We solved the BN-CVaR(β) formulation setting $\theta = 3.1$ seeking the minimum CVaR(β) portfolio, and computed the corresponding portfolio return. Subsequently, we found the return of the feasible portfolio that reaches the maximum yield. Finally, we divided that interval in five equally spaced sub-intervals. All the computational experiments are performed setting a value of β equal to 0.05.

4.3.2 In-Sample Analysis

This section is devoted to the presentation of the in-sample characteristics of the optimal portfolios selected by the two CVaR(β) robust counterparts formulated applying the described robust approaches. For the sake of clarity, we have decided to only comment the results concerning the optimal portfolios selected in the down-down and in the up-down data sets. As a matter of fact, the robust formulations are expected to provide evidence of their effectiveness especially when the market index is going down out-of-sample. Anyhow, the characteristics of the optimal portfolios selected in the remaining periods are similar to those commented in the present section. The complete set of results can be found in Guastaroba *et al.* [77].
Tables 4.1 and 4.2 show the characteristics of the optimal portfolios selected in the down-down and in the up-down period, respectively. Each table is divided into two parts providing the results for the BN and BS robust counterparts, respectively. Each part consists of five columns that show the objective function value (column obj.), the in-sample mean portfolio return (Mean Return), the number of securities selected (div.), the maximum (max) portfolio share and the in-sample portfolio variance (σ^2). We highlight that the portfolio robust return is, for any instance, equal to μ_0.

As a starting point of the analysis, we have compared the two robust formulations for the same level of robustness. The aim is to compare the "conservatism" of the two robust approaches. At this aim, a conclusion we feel we can draw is that the BN robust technique is more conservative than the BS technique. Two evidences support this claim. Firstly, the objective function value of the optimal portfolios selected by the BN-CVaR(β) formulation is often smaller than the value of the same parameter computed for the corresponding portfolios selected by the BS-CVaR(β). This is evident observing the results depicted in Table 4.2 for any value of μ_0 and of θ and Γ, and is also true in the down-down period for the highest level of robustness (compare in Table 4.1 the optimal solutions selected setting θ=3.1 and Γ=25, respectively). Secondly, considering the highest levels of robustness often the BN robust counterpart could not find any feasible solution, whereas the feasible domain of the BS robust formulation is not empty. This can be justified comparing the minimum required rate of return constraint of the BN and of the BS robust counterpart. Indeed, setting θ or Γ to a value greater than zero means to require in-sample a value of the robust

	BN-CVaR(β)					BS-CVaR(β)				
Expected Return (μ_0)	obj. 10^2	Mean Return	div.	max	σ^2	obj. 10^2	Mean Return	div.	max	σ^2
	θ=0					**Γ=0**				
$\mu_0 \leq 0.27800$	24.0466	0.782011	12	0.285617	0.115305	24.0466	0.782011	12	0.285617	0.115305
	θ=1.9187					**Γ=11**				
0.08450	24.0466	0.782011	12	0.285617	0.115305	24.0466	0.782011	12	0.285617	0.115305
0.12320	24.0466	0.782011	12	0.285617	0.115305	23.7572	0.778880	17	0.290301	0.115714
0.16190	24.0466	0.782011	12	0.285617	0.115305	23.0855	0.783586	21	0.278241	0.120997
0.20060	24.0466	0.782011	12	0.285617	0.115305	22.0379	0.740836	21	0.276451	0.135917
0.23930	23.5857	0.736647	18	0.249823	0.093166	20.3381	0.728956	23	0.257420	0.149175
0.27800	22.3340	0.692486	21	0.206307	0.076017	11.9213	1.007800	30	0.570190	0.383668
	θ=2.161					**Γ=14**				
0.08450	24.0466	0.782011	12	0.285617	0.115305	24.0459	0.783198	12	0.286098	0.115943
0.12320	24.0466	0.782011	12	0.285617	0.115305	23.5875	0.810153	21	0.300326	0.133563
0.16190	23.8403	0.750382	17	0.262614	0.099627	22.3711	0.827815	23	0.302511	0.152949
0.20060	23.0740	0.705666	20	0.228247	0.080167	20.4430	0.831226	25	0.317825	0.167267
0.23930	21.8134	0.668541	22	0.184495	0.068282	17.3154	0.836010	28	0.354071	0.187955
0.27800	15.8791	0.590878	31	0.102213	0.056084	10.6923	1.080970	31	0.571665	0.452455
	θ=3.1					**Γ=25**				
0.08450	21.0106	0.632249	23	0.166082	0.057785	24.0459	0.783198	12	0.286098	0.115943
0.12320	19.1513	0.591679	23	0.125217	0.049108	23.5536	0.841477	10	0.312625	0.152572
0.16190	14.1145	0.517648	43	0.075539	0.041779	22.1633	0.898911	8	0.345069	0.205208
0.20060			infeasible			19.6040	0.955508	8	0.397293	0.266443
0.23930			infeasible			16.0763	1.007654	6	0.453351	0.333818
0.27800			infeasible			10.5923	1.086432	3	0.559946	0.455071

Table 4.1: Down-down period: Optimal in-sample portfolio characteristics.

return greater than μ_0. The bigger the level of robustness, the higher the value of the robust return required. Subsequently, the robust return required for the highest level of robustness is often exceedingly high if the robust approach is the one proposed by Ben-Tal and Nemirovski.
The mean return of the portfolios selected by the two different robust formulations have opposite behavior. As a matter of fact, either increasing the value of μ_0, for the same value of $\theta\backslash\Gamma$, or increasing the value of $\theta\backslash\Gamma$, while keeping the same value of μ_0, the mean return of the portfolios selected by the BN robust counterpart is decreasing, whereas the same parameter computed for the portfolios selected by the BS robust counterpart goes uphill. Indeed, the computational experiments have shown that the BN-CVaR(β) model tends to select less yielding, and indeed less risky, securities as the value of θ and/or μ_0 increases. The opposite behavior has been shown by the BS robust counterpart. In fact, both approaches select approximately the same set of securities, especially in the up-down period (Table 4.2), but the BN robust model tends to weigh more stocks which show a low standard deviation with respect to the BS robust formulation. This behavior is confirmed by the portfolio variance values depicted in Tables 4.1 and 4.2.
Finally, it is worth noting that in the up-down period (see Table 4.2) the BS-CVaR(β) robust model selects the same robust portfolio for a given value of μ_0, any value of the budget of uncertainty Γ greater than zero is selected. The BS-CVaR(β) robust model shows the same behavior in the down-up and in the up-up periods as well.

4.3.3 Out-of-sample analysis

In the out-of-sample analysis we examine the behavior of the portfolios selected by the two CVaR(β) robust counterparts in the 52 weeks following the date of portfolio selection. Firstly,

	BN-CVaR(β)					BS-CVaR(β)				
Expected Return (μ_0)	obj. 10^2	Mean Return	div.	max	σ^2	obj. 10^2	Mean Return	div.	max	σ^2
	θ**=0**					Γ**=0**				
$\mu_0 \leq 0.54000$	42.0382	1.85646	11	0.446766	1.073437	42.0382	1.85646	11	0.446766	1.073437
	θ**=1.9187**					Γ**=11**				
0.31380	40.7757	1.65174	13	0.325824	0.717970	42.0382	1.85645	11	0.446766	1.073437
0.35904	40.1629	1.61455	13	0.303090	0.667832	42.0382	1.85645	11	0.446766	1.073437
0.40428	39.4546	1.57787	13	0.279114	0.618648	42.0376	1.86032	11	0.448168	1.078367
0.44952	38.5758	1.52245	15	0.248651	0.554714	42.0180	1.95798	10	0.494691	1.259690
0.49476	37.3849	1.46682	14	0.217222	0.492959	41.8775	2.05360	9	0.533098	1.422875
0.54000	35.7564	1.39449	15	0.178242	0.424527	41.4268	2.12529	8	0.569953	1.574237
	θ**=2.161**					Γ**=14**				
0.31380	38.6263	1.49336	15	0.243359	0.525230	42.0382	1.85645	11	0.446766	1.073437
0.35904	37.8128	1.42022	15	0.212502	0.450904	42.0382	1.85645	11	0.446766	1.073437
0.40428	36.6794	1.37621	15	0.186630	0.410584	42.0376	1.86032	11	0.448168	1.078367
0.44952	35.2697	1.33272	15	0.169969	0.372941	42.0180	1.95798	10	0.494691	1.259690
0.49476	33.0276	1.26292	16	0.157001	0.323838	41.8775	2.05360	9	0.533098	1.422875
0.54000	infeasible					41.4268	2.12529	8	0.569953	1.574237
	θ**=3.1**					Γ**=25**				
$\mu_0 \leq 0.35904$	infeasible					42.0382	1.85645	11	0.446766	1.073437
0.40428	infeasible					42.0376	1.86032	11	0.448168	1.078367
0.44952	infeasible					42.0180	1.95798	10	0.494691	1.259690
0.49476	infeasible					41.8775	2.05360	9	0.533098	1.422875
0.54000	infeasible					41.4268	2.12529	8	0.569953	1.574237

Table 4.2: Up-down period: Optimal in-sample portfolio characteristics.

we discuss the two approaches separately. Subsequently, we draw some conclusions comparing the two approaches. Finally, we compare the optimal portfolios with the market index.

To compare the out-of-sample optimal portfolio performances we have computed cumulative returns and the following six ex-post parameters: the number of times out of 52 the portfolio rate of return outperforms the corresponding required one in-sample (#), the average (r_{av}) and the median (r_{med}) of the rate of returns on yearly basis, the standard deviation (std), the semi-standard deviation (s-std) and the Sortino index. Both dispersion measures (std, s-std) and the Sortino index have been computed with respect to the minimum rate of return required in-sample. As for the in-sample analysis, we carefully describe only the computational results obtained in the down-down (Tables 4.3 and 4.4) and in the up-down (Tables 4.5 and 4.6) periods. The complete set of results can be found in Guastaroba *et al.* [77]

We point out that the ex-post characteristics of the portfolios selected in the down-up and in the up-up periods are quite similar to those selected in the down-down and in the up-down periods, respectively (and in the following we will mention any different behavior). Furthermore, this peculiarity allows us to draw some conclusions about the quality of the optimized portfolios based on in-sample trend of the market index.

In Table 4.3 we show all the resulting values for the performance measures and the dispersion parameters we have computed for the optimal portfolios selected in the down-down period by the BN-CVaR(β) model. For each value of μ_0 the market index behavior of the FTSE100 is also shown.

When the market index goes downhill both in-sample and out-of-sample, for a given value of μ_0 the higher the θ, the lower the average portfolio return. The same conclusion can be drawn considering an increasing value of μ_0 for a given value of θ. As far as the dispersion measures are concerned, it is worth noting that they show a slight tendency to increase when augmenting the value of θ and/or

Expected Return	#	r_{av}	r_{med}	std	s-std	Sortino index
μ_0 = **0.08450**						
$\theta \leq 2.161$	51.9231	0.050139	0.222666	0.023002	0.016973	-0.036525
$\theta = 3.1$	48.0769	0.025876	-0.001789	0.022963	0.017654	-0.060599
FTSE100	40.3846	-0.229322	-0.143048	0.034002	0.028679	-0.228668
μ_0= **0.12320**						
$\theta \leq 2.161$	50.0000	0.050139	0.222666	0.023036	0.017353	-0.074654
$\theta = 3.1$	44.2308	0.024495	0.026030	0.023226	0.018409	-0.096221
FTSE100	40.3846	-0.229322	-0.143048	0.034002	0.029041	-0.249077
μ_0= **0.16190**						
$\theta \leq 1.9187$	50.0000	0.050139	0.222666	0.023087	0.017726	-0.109931
$\theta = 2.161$	46.1538	0.044976	0.074272	0.022881	0.017563	-0.116353
$\theta = 3.1$	46.1538	0.005591	0.071876	0.024561	0.020343	-0.136787
FTSE100	40.3846	-0.229322	-0.143048	0.034002	0.029396	-0.268287
μ_0= **0.20060**						
$\theta \leq 1.9187$	50.0000	0.050139	0.222666	0.023153	0.018090	-0.142659
$\theta = 2.161$	46.1538	0.048948	-0.064229	0.022681	0.017946	-0.145021
$\theta = 3.1$			infeasible			
FTSE100	38.4615	-0.229322	-0.143048	0.034002	0.029744	-0.286403
μ_0 = **0.23930**						
$\theta = 0$	48.0769	0.050139	0.222666	0.023234	0.018447	-0.173102
$\theta = 1.9187$	44.2308	0.046920	0.012978	0.022935	0.018290	-0.177818
$\theta = 2.161$	44.2308	0.023641	-0.063026	0.023152	0.019005	-0.193894
$\theta = 3.1$			infeasible			
FTSE100	36.5385	-0.229322	-0.143048	0.034002	0.030085	-0.303516
μ_0 = **0.27800**						
$\theta = 0$	48.0769	0.050139	0.222666	0.023326	0.018796	-0.201488
$\theta = 1.9187$	46.1538	0.030562	-0.099351	0.023110	0.019170	-0.216449
$\theta = 2.161$	42.3077	0.009988	0.032619	0.024995	0.021255	-0.213470
$\theta = 3.1$			infeasible			
FTSE100	34.6154	-0.229322	-0.143048	0.034002	0.030419	-0.319706

Table 4.3: Out-of-sample statistics for the BN-CVaR(β) model: Down-down data set.

of μ_0, especially the semi-standard deviation. In other words, increasing the value of θ and/or of μ_0 the BN robust counterpart tends to select portfolios that show slightly poorer out-of-sample performances.

Similar conclusions can be drawn observing Table 4.4 that shows the same parameters computed for the optimal portfolios selected in the down-down period by the BS-CVaR(β) model. In fact, all the performance and dispersion measures tend to take worse values increasing the value of Γ and/or of μ_0. This is in particular true for the highest values of the required rate of return (see the figures concerning μ_0 equal to 0.23930 and to 0.27800 in Table 4.4).

As far as a comparison between the two approaches is taken into account, we highlight that the BN-CVaR(β) robust model often selects portfolios that outperform those selected by the BS robust counterpart in terms of average ex-post return. This is evident comparing the figures depicted in Table 4.3 with those in Table 4.4 for most values of θ and Γ and of μ_0, with the only exception of the instances solved setting θ equal to 3.1 (and, equivalently Γ =25) where the BN-CVaR(β) model is not able to find any feasible solution. Furthermore, considering the instances with high values of μ_0 the BN approach selects portfolios that show a lower downside risk than the corresponding portfolios selected by the BS approach. This result confirms the tendency showed in-sample by the BS-CVaR(β) model to select riskier portfolios as the value of μ_0, and of Γ, increases. Summarizing the results with the help of the Sortino index, in the down-down data set for most of the instances solved to optimality the BN robust model finds better portfolios than the BS robust counterpart.

Expected Return	#	r_{av}	r_{med}	std	s-std	Sortino index
μ_0 **= 0.08450**						
$\Gamma \leq 11$	51.9231	0.050139	0.222666	0.023002	0.016973	-0.036525
$\Gamma \geq 14$	51.9231	0.050161	0.225124	0.023005	0.016973	-0.036501
FTSE100	40.3846	-0.229322	-0.143048	0.034002	0.028679	-0.228668
μ_0**= 0.12320**						
Γ = 0	50.0000	0.050139	0.222666	0.023036	0.017353	-0.074654
Γ = 11	50.0000	0.036368	0.138476	0.023049	0.017391	-0.089099
Γ = 14	50.0000	0.039200	0.207067	0.022957	0.017285	-0.086612
Γ = 25	50.0000	0.046690	0.164126	0.023166	0.017293	-0.078576
FTSE100	40.3846	-0.229322	-0.143048	0.034002	0.029041	-0.249077
μ_0**= 0.16190**						
Γ = 0	50.0000	0.050139	0.222666	0.023087	0.017726	-0.109931
Γ = 11	46.1538	0.039186	0.153013	0.022540	0.017436	-0.123331
Γ = 14	46.1538	0.022742	0.085653	0.022872	0.017642	-0.139287
Γ = 25	42.3077	0.033626	0.031552	0.023953	0.017991	-0.125266
FTSE100	40.3846	-0.229322	-0.143048	0.034002	0.029396	-0.268287
μ_0**= 0.20060**						
Γ = 0	50.0000	0.050139	0.222666	0.023153	0.018090	-0.142659
Γ = 11	48.0769	0.008235	-0.035621	0.022953	0.018404	-0.182798
Γ = 14	44.2308	0.012761	-0.018807	0.022748	0.017953	-0.182599
Γ = 25	40.3846	0.013017	0.015289	0.025460	0.019369	-0.168994
FTSE100	38.4615	-0.229322	-0.143048	0.034002	0.029744	-0.286403
μ_0 **= 0.23930**						
Γ = 0	48.0769	0.050139	0.222666	0.023234	0.018447	-0.173102
Γ = 11	40.3846	-0.014065	0.026338	0.023139	0.018974	-0.232258
Γ = 14	44.2308	-0.025885	-0.091046	0.024229	0.019405	-0.239043
Γ = 25	42.3077	-0.001088	0.058555	0.027379	0.021194	-0.196060
FTSE100	36.5385	-0.229322	-0.143048	0.034002	0.030085	-0.303516
μ_0 **= 0.27800**						
Γ = 0	48.0769	0.050139	0.222666	0.023326	0.018796	-0.201488
Γ = 11	44.2308	0.040053	-0.037473	0.025445	0.019679	-0.201884
Γ = 14	40.3846	0.000905	-0.026040	0.028335	0.021939	-0.214728
Γ = 25	40.3846	-0.007948	-0.091614	0.028882	0.022413	-0.217815
FTSE100	34.6154	-0.229322	-0.143048	0.034002	0.030419	-0.319706

Table 4.4: Out-of-sample statistics for the BS-CVaR(β) model: Down-down data set.

When the market index goes uphill in-sample and decreases out-of-sample, the portfolios selected by the BN robust model yield higher average returns for higher level of robustness, considering the same value of μ_0 (see Table 4.5). The same conclusion can be drawn observing that the average portfolio return increases for higher values of μ_0, mantaining the same value of θ. As far as the dispersion measures are concerned, one should notice that both the standard deviation and the semi-standard deviation decrease for higher values of θ, choosing the same value of μ_0. The same trend is shown for higher values of μ_0, keeping the same level of robustness. The same portfolio characteristics are found in the up-up data set. To conclude, when the market index goes uphill in-sample and assuming the BN approach to deal with uncertainty, the investor should choose carefully the level of robustness of the solution, since higher values of θ provide better portfolios but too high values of this parameter determine the infeasibility of the model.
In Table 4.6 we show the ex-post parameters we computed for the BS optimal portfolios in the same data set. Even if the BN model frequently selects the same portfolio composition for different values of Γ, we feel we can draw the following conclusions. Considering the up-down data set, the average portfolio return tends to decrease for higher levels of robustness, as well as for higher values of μ_0. However, we have to highlight that in the up-up period the average portfolio return shows an uphill tendency for higher values of Γ, and of μ_0. When the volatility is taken

Expected Return	#	r_{av}	r_{med}	std	s-std	Sortino index
μ_0 **= 0.31380**						
θ = 0	38.4615	-0.344930	-0.566861	0.072165	0.055373	-0.241347
θ = 1.9187	38.4615	-0.305842	-0.430040	0.066290	0.051098	-0.239895
θ = 2.161	38.4615	-0.276790	-0.531126	0.062231	0.047896	-0.239578
θ = 3.1	40.3846	-0.104698	-0.229370	0.042362	0.032523	-0.227129
FTSE100	36.5385	-0.085385	0.187603	0.026850	0.022859	-0.305235
μ_0**= 0.35904**						
θ = 0	38.4615	-0.344911	-0.566906	0.072275	0.055790	-0.251269
θ = 1.9187	40.3846	-0.294101	-0.449356	0.065620	0.050671	-0.248511
θ = 2.161	38.4615	-0.273826	-0.511314	0.060157	0.046872	-0.257111
θ = 3.1			infeasible			
FTSE100	36.5385	-0.085385	0.187603	0.026850	0.023246	-0.328305
μ_0**= 0.40428**						
θ = 0	38.4615	-0.344934	-0.566879	0.072387	0.056196	-0.260742
θ = 1.9187	38.4615	-0.290912	-0.540147	0.065365	0.050789	-0.258718
θ = 2.161	38.4615	-0.260798	-0.492718	0.059935	0.046922	-0.263100
θ = 3.1			infeasible			
FTSE100	36.5385	-0.085385	0.187603	0.026850	0.023627	-0.349840
μ_0**= 0.44952**						
θ = 0	38.4615	-0.344931	-0.566863	0.072499	0.056589	-0.269780
θ = 1.9187	38.4615	-0.282384	-0.528264	0.063938	0.050000	-0.270512
θ = 2.161	38.4615	-0.261902	-0.442411	0.059134	0.046630	-0.278524
θ = 3.1			infeasible			
FTSE100	34.6154	-0.085385	0.187603	0.026850	0.024000	-0.369985
μ_0 **= 0.49476**						
θ = 0	36.5385	-0.344930	-0.566861	0.072613	0.056972	-0.278418
θ = 1.9187	38.4615	-0.283210	-0.518015	0.063528	0.050114	-0.282216
θ = 2.161	40.3846	-0.239990	-0.482185	0.058411	0.046052	-0.282799
θ = 3.1			infeasible			
FTSE100	34.6154	-0.085385	0.187603	0.026850	0.024366	-0.388867
μ_0 **= 0.54000**						
θ = 0	36.5385	-0.344930	-0.566863	0.072728	0.057345	-0.286687
θ = 1.9187	38.4615	-0.275224	-0.467158	0.062208	0.049351	-0.293999
θ = 2.161			infeasible			
θ = 3.1			infeasible			
FTSE100	34.6154	-0.085385	0.187603	0.026850	0.024725	-0.406599

Table 4.5: Out-of-sample statistics for the BN-CVaR(β) model: Up-down data set.

into account, one should notice that both dispersion measures we computed take higher values for higher level of robustness (the same behavior is shown for higher values of μ_0). The same conclusions can be drawn observing the portfolios selected in the up-up period. Summarizing, when the market trend increases in-sample and decreases out-of-sample, and under the assumption that the BS-CVaR(β) formulation is the reference model, the best ex-post performances are provided by the optimal portfolios selected by the nominal model, i.e. the model that does not guarantee any protection against parameter uncertainty. The reverse is true if the market index is increasing out-of-sample, i.e. the up-up data set. In fact, despite in this period the portfolio volatility increases for higher values of Γ and of μ_0, the best trade-off between risk and return is shown by the portfolios selected requiring a high value of μ_0 and setting a high level of robustness. When the two approaches are compared, the BN-CVaR(β) robust model usually selects better portfolios than those selected by the BS robust counterpart, both in terms of average return and of the dispersion measures we computed. The dominance of the BN approach is more and more evident increasing the value of μ_0 and the level of robustness, especially in the up-down period.
For sake of completeness, we have plotted six graphs (Figures 4.3-4.8) showing the cumulative returns yielded out-of-sample by the optimal portfolios. Specifically, Figures 4.3-4.6 offer a com-

Expected Return	#	r_{av}	r_{med}	std	s-std	Sortino index
μ_0 = **0.31380**						
$\Gamma \geq 0$	38.4615	-0.344930	-0.566861	0.072165	0.055373	-0.241347
FTSE100	36.5385	-0.085385	0.187603	0.026850	0.022859	-0.305235
μ_0= **0.35904**						
$\Gamma \geq 0$	38.4615	-0.344930	-0.566861	0.072275	0.055790	-0.251278
FTSE100	36.5385	-0.085385	0.187603	0.026850	0.023246	-0.328305
μ_0= **0.40428**						
Γ = 0	38.4615	-0.344930	-0.566861	0.072386	0.056195	-0.260744
$\Gamma \geq 11$	38.4615	-0.345603	-0.528947	0.072551	0.056330	-0.260470
FTSE100	36.5385	-0.085385	0.187603	0.026850	0.023627	-0.349840
μ_0= **0.44952**						
Γ = 0	38.4615	-0.344930	-0.566861	0.072499	0.056589	-0.269780
$\Gamma \geq 11$	34.6154	-0.345448	-0.550591	0.075710	0.058711	-0.260286
FTSE100	34.6154	-0.085385	0.187603	0.026850	0.024000	-0.369985
μ_0 = **0.49476**						
Γ = 0	36.5385	-0.344930	-0.566861	0.072613	0.056972	-0.278418
$\Gamma \geq 11$	34.6154	-0.391212	-0.583722	0.078096	0.061019	-0.282841
FTSE100	34.6154	-0.085385	0.187603	0.026850	0.024366	-0.388867
μ_0 = **0.54000**						
Γ = 0	36.5385	-0.344930	-0.566861	0.072728	0.057345	-0.286687
$\Gamma \geq 11$	34.6154	-0.385590	-0.515148	0.081076	0.063463	-0.278296
FTSE100	34.6154	-0.085385	0.187603	0.026850	0.024725	-0.406599

Table 4.6: Out-of-sample statistics for the BS-CVaR(β) model: Up-down data set.

parison of the cumulative returns yielded, in each of the data set investigated, by the optimized portfolios selected solving the nominal model (i.e., the Nom(μ_0) time series) and the two robust techniques for approximately the same level of robustness (i.e., the BN(μ_0, θ) and the BS(μ_0, Γ) times series). To maintain the graph readable we decided to plot only one portfolio trend for each optimization model we solved. To this aim, in each graph, we depicted three optimal portfolios selected requiring in-sample the same value of μ_0. On the other hand, Figures 4.7 and 4.8 compare, respectively, the optimal BN-CVaR(β) and BS-CVaR(β) portfolios selected in the same time period choosing different levels of robustness. In order to allow a comparison with the market index trend, in every graph the market index behavior (i.e., the FTSE100 time series) has been reported as well. We report cumulative returns since they allow us to evaluate how the portfolio value changes over time and provide an additional tool to compare the behavior of the portfolios selected by the nominal problem and the two robust counterparts.

It is worth observing that most of the conclusions we have drawn formerly can be confirmed observing Figures 4.3-4.6. When we compare the two robust optimization techniques, the BN robust portfolios dominate the BS robust ones in most of the tested data set. This can be noted by observing the cumulative returns yielded by the robust portfolios selected in all cases and especially in the down-down period (see Figure 4.3), in the down-up period (see Figure 4.4), and in the up-down period (see Figure 4.5) where the BN portfolio trends are almost ever better than the corresponding BS ones. The experimental analysis we conducted suggests that the robust portfolio optimization technique proposed by Ben-Tal and Nemirovski tends to select portfolios that have a better ex-post performance than those selected by the robust counterpart in the sense of Bertsimas and Sim.

Figures 4.3-4.6 also show the cumulative returns of the nominal portfolio, as it is interesting to evaluate the impact of introducing a robustness component in the nominal problem. The results suggest that robust portfolio optimization is primarily useful when the market trend is increasing in-sample, i.e. in the up-down and up-up periods (see Figures 4.5 and 4.6). Especially the

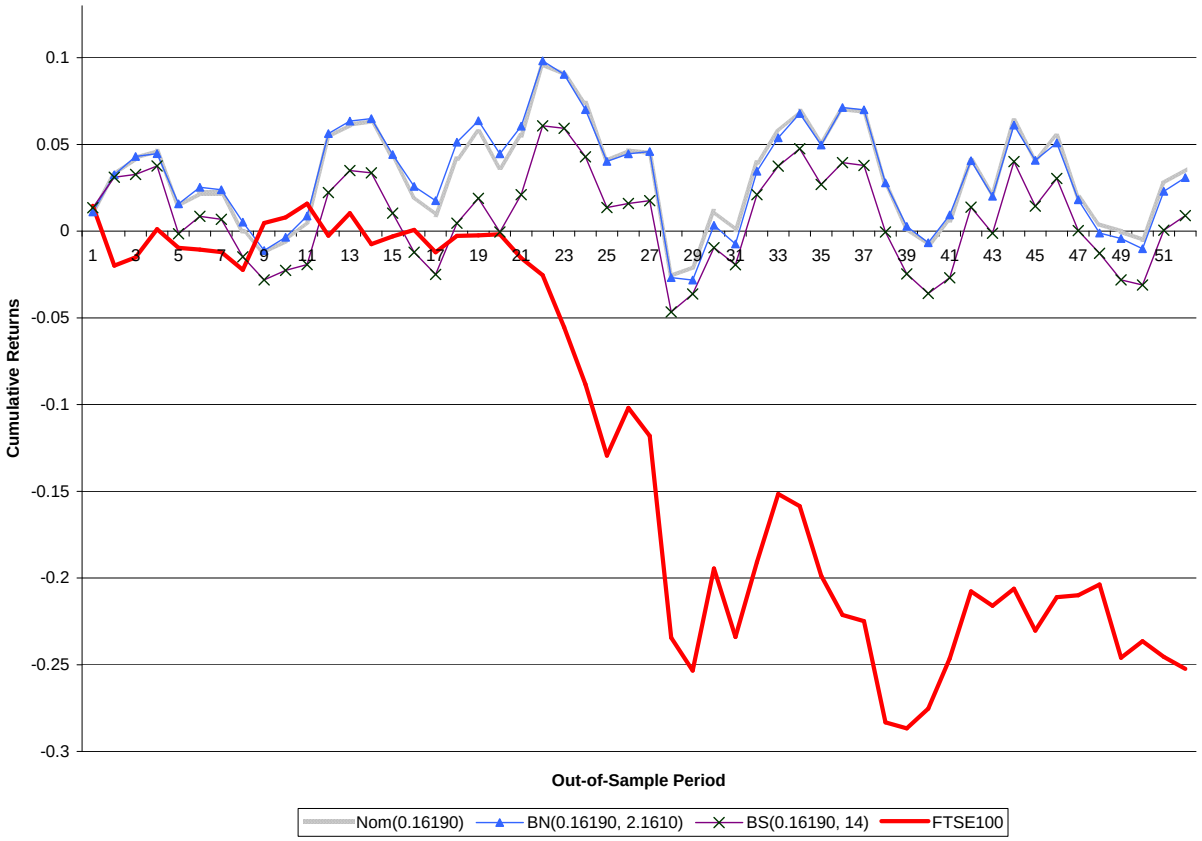

Figure 4.3: Cumulative returns (down-down data set): A comparison between robust techniques.

portfolios selected by the BN-CVaR(β) show better ex-post cumulative returns than the portfolios selected by the nominal problem. In the down-down period (see Figure 4.3) the nominal portfolio closely follows the Ben-Tal and Nemirovski robust portfolio, whereas, somehow surprisingly, in the down-up period (see Figure 4.4), the optimal portfolio selected by the nominal problem tends to perform better than the portfolios selected by the two robust counterparts. However, in this latter case the performance of the Ben-Tal and Nemirovski robust portfolio is close to that of the nominal portfolio.

Figures 4.3-4.6 also show, as an additional information element, the FTSE100 behavior. All the optimized portfolios, the nominal one and the robust counterparts, have an excellent performance with respect to the market index in the down-down period (see Figure 4.3), which is the most challenging situation for an investor. On the other hand, the performance of all the optimized portfolios is relatively unsatisfactory, when compared to the index, in the up-up period (see Figure 4.6). Although most of the time in the ex-post period of the latter data set the performance of the optimized portfolios is positive, the cumulative return remains constantly below the return of the index and there are times where the cumulative return of all the optimized portfolios is negative. The latter situation can be partially overcome by choosing a higher value of the expected return (see the Appendix). Indeed, only in this case there exists at least an optimal portfolio whose cumulative return is always positive during the entire ex-post period. However, even in this case, the

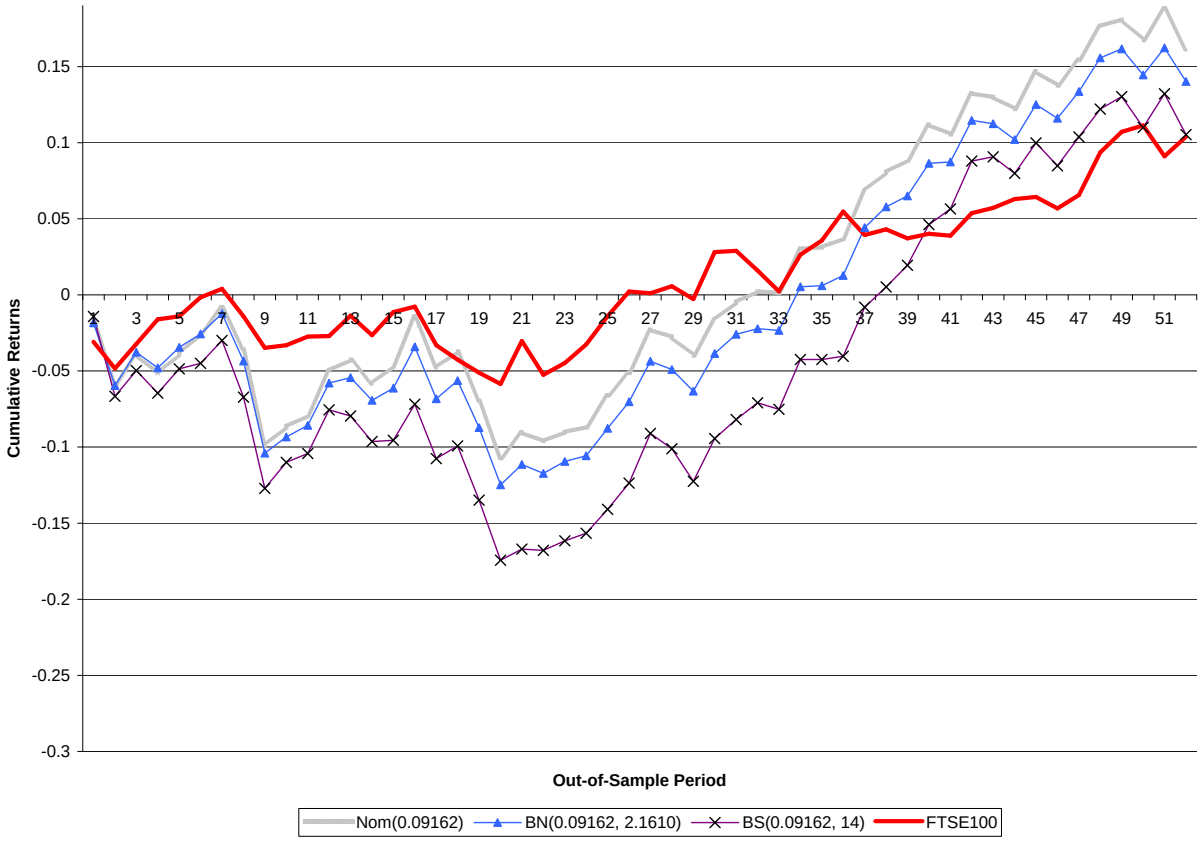

Figure 4.4: Cumulative returns (down-up data set): A comparison between robust techniques.

portfolio cumulative returns are frequently outperformed by the market index return. In the two periods that represent a change of market trend, the down-up and the up-down periods (see Figures 4.4 and 4.5), the ex-post behavior of the optimized portfolios with respect to the index depends on the evaluation time. Only in the first 3-4 months of the ex-post time range of the up-down period the optimized portfolios outperform the index (see Figure 4.5), whereas in the down-up period (see Figure 4.4) the good performance of the optimized portfolios can be observed only in last 3-4 months of the ex-post time range.
Finally, in Figures 4.7 and 4.8 we compare the ex-post performance of robust portfolios with different level of robustness. Figure 4.7 shows the optimal portfolios selected by the BN-CVaR(β) model in the up-down period, whereas Figure 4.8 shows the cumulative returns yielded by the optimal portfolios selected by the BS-CVaR(β) model in the down-down period. In both cases, the impact of the level of robustness on the portfolio performance seems to be marginal, especially when the performance is compared with the market index.

4.4 Conclusions

In this chapter we studied the impact of robust optimization techniques on a portfolio selection problem. Specifically, we formulated two robust counterparts according to the two best known ro-

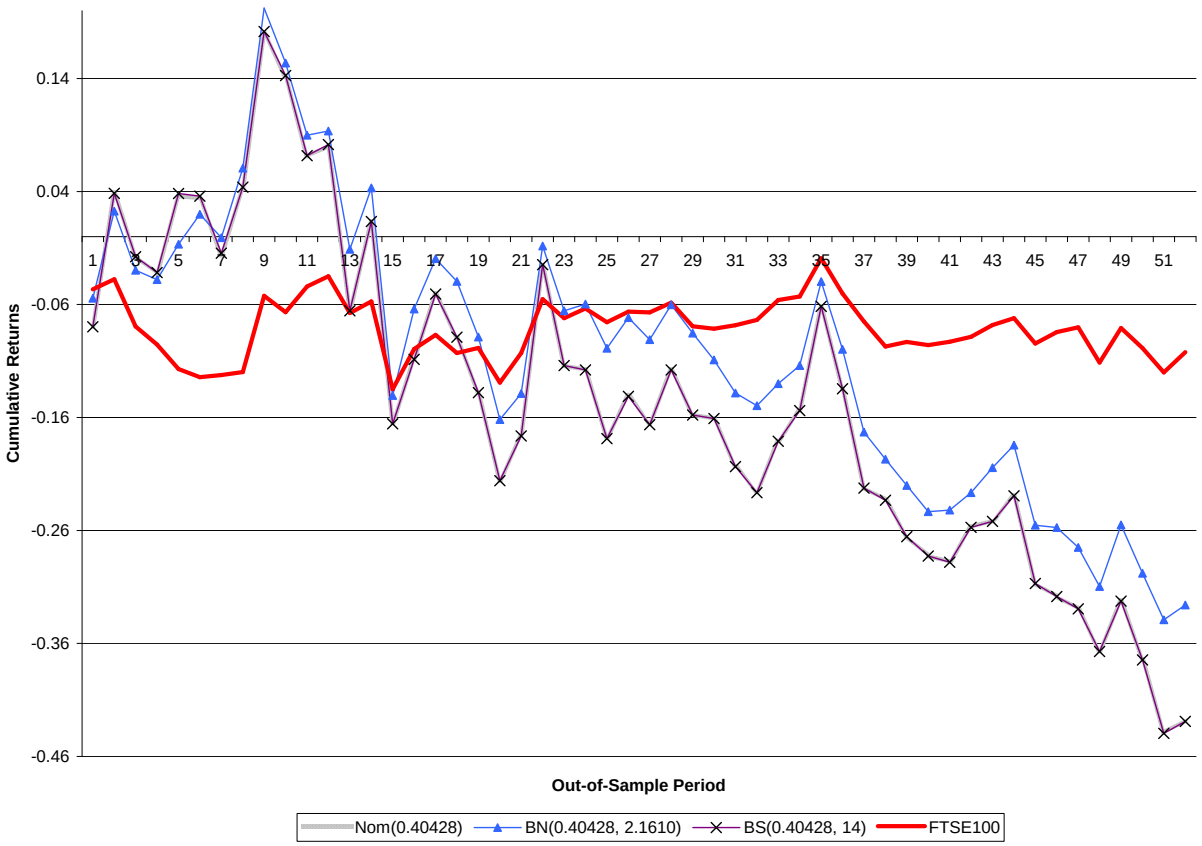

Figure 4.5: Cumulative returns (up-down data set): A comparison between robust techniques.

bust optimization techniques (due to Ben-Tal and Nemirovski [12] and to Bertsimas and Sim [15]) of a portfolio optimization model where the Conditional Value-at-Risk is the portfolio performance measure. The main goal of the research was to evaluate the effectiveness of robust optimization to support financial decisions. The extensive computational results we carried out on real-life data from the London Stock Exchange allowed us to draw some interesting conclusions on the performance of the nominal and of the robust portfolios, under different in-sample and out-of-sample scenarios. In general, the nominal and the robust portfolios tend to have a similar behavior, with a generally better performance of the Ben-Tal and Nemirovski robust portfolios.
Whereas robust optimization techniques are, without any doubt, theoretically valuable, the evaluation of their effectiveness on specific classes of problems remains an interesting research direction.

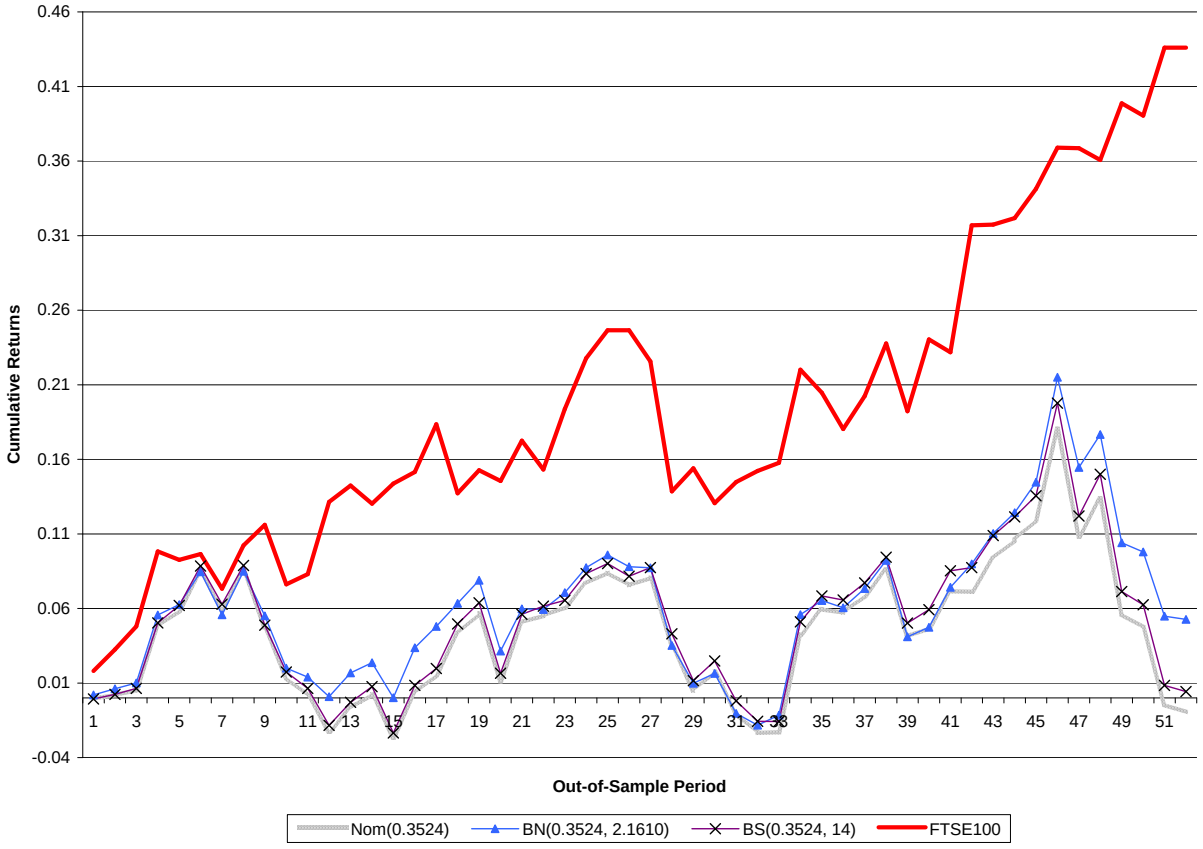

Figure 4.6: Cumulative returns (up-up data set): A comparison between robust techniques.

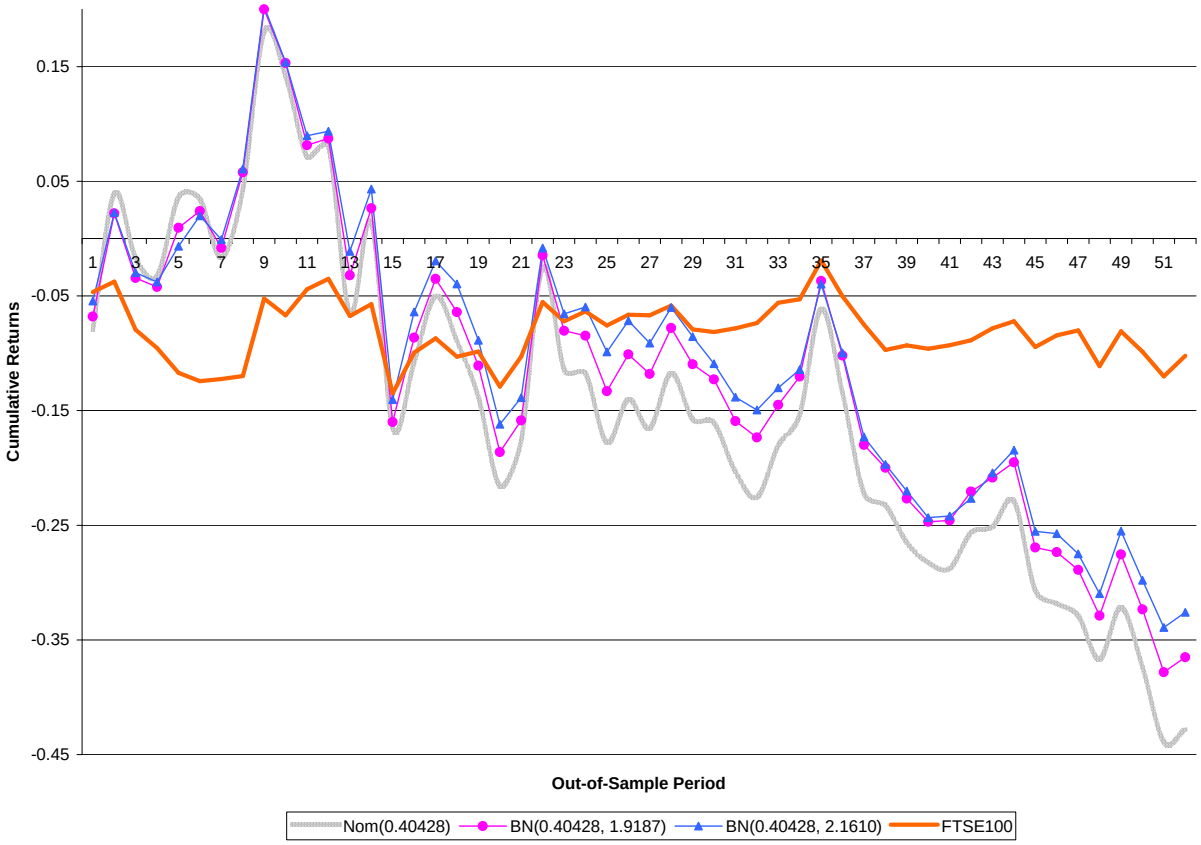

Figure 4.7: The BN-CVaR(β) model: A comparison between different level of robustness (up-down data set).

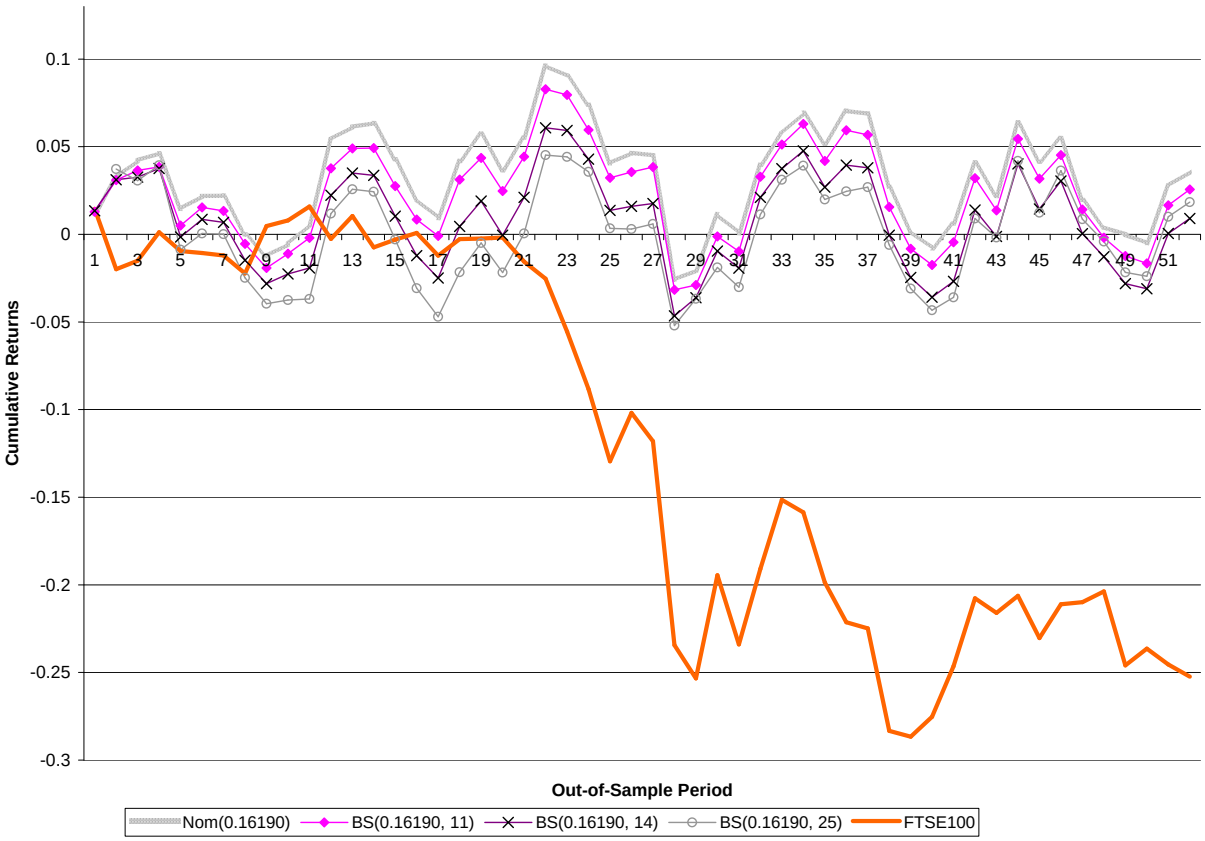

Figure 4.8: The BS-CVaR(β) model: A comparison between different level of robustness (down-down data set).

5

Models and Simulations for Portfolio Rebalancing

5.1 Introduction

The original mean-risk portfolio formulation introduced by Markowitz [117] has provided the fundamental basis for the development of a large part of the modern financial theory applied to the *single-period* portfolio optimization problem (also known as *buy-and-hold* portfolio strategy). One basic implication of Markowitz model is that investors hold well diversified portfolios of securities. Nevertheless, in practice, investors typically select portfolios consisting in a small number of securities (see Blume *et al.* [21]). There are many reasons for this fact, the most relevant of which is the presence of transaction costs. In financial markets, any movement of money among assets incurs in the payment of transaction costs. In many papers the emphasis has been put on properly modelling transaction costs. Patel and Subrahmanyam [130] first study the fixed transaction costs impact on portfolio diversification. Pogue [134] first deal with the proportional brokerage fees involved in revising an existing portfolio. Yoshimoto [150] analyze the problem of portfolio optimization with variable transaction costs adopting a Mean-Variance approach. Gennotte and Jung [69] examine the effect of proportional transaction costs on dynamic portfolio strategies. Finally, recent contributions on optimization models with transaction costs have emphasized the critical aspect of computational testing and the need for efficient algorithms to solve them. In Konno and Wijayanayake [96] and Mansini and Speranza [114] the authors propose exact solution algorithms for portfolio optimization problems with transaction costs, whereas different heuristic solution algorithms can be found in Kellerer *et al.* [91] and Chiodi *et al.* [34].
Transaction costs also affect portfolio optimization problems in a multi-period framework. Due to sudden changes in the market trend an investor might prefer to rebalance her/his portfolio composition to possibly reduce losses or better exploit returns growth. Such adjustments might be desirable even if implying additional costs. Markowitz [118] recognizes the importance of intermediate adjustment opportunities. Smith [142] introduces a model of portfolio revision with transaction costs. His approach modifies the Markowitz single-period formulation and aims at applying the solution period by period. In Li *et al.* [106] and Li *et al.* [105] the authors extend the portfolio optimization setting of the Mean-Variance approach proposed by Markowitz to the case with transaction costs formulated as a V-shaped function of differences between old and new portfolios avoiding short sales. They also propose solution algorithms to the studied problems. Elton and Gruber [61] show that, under some specific assumptions, even if no new information is expected to arise over time, changing the portfolio will increase the expected utility of the terminal wealth. Li and Ng [104]

and Steinbach [145] study extensions of the Markowitz Mean-Variance approach to multi-period portfolio optimization problems.
This research aims at studying different investment policies as practical tools for the portfolio management in a multi-period framework. In particular, we will analyze a rebalancing portfolio optimization model and show how optimization could represent a critical tool for financial managers.
Structure of the Chapter. In Section 5.2 we introduce the variant of the CVaR(β) optimization model considering portfolio rebalancing. Section 5.3 is devoted to the experimental analysis and to the comparison of different investment strategies. Alternative data sets are used to analyze most of the possible market trends. The main objectives are to provide evidence on the effectiveness of the rebalancing optimization model in leading financial decisions under different market conditions and to help decision-makers in choosing the proper portfolio rebalancing frequency. Finally, in Section 5.4 some concluding remarks are drawn.
The content of this chapter has been published in [80].

5.2 The Models

5.2.1 Notes on the CVaR(β) Model with Transaction Costs

We use the CVaR(β) measure of risk to model a real case situation where an investor optimally selects a portfolio and holds it until the end of the investment period. For each selected security the investor incurs a fixed and a proportional cost. Specifically, we consider the single-period CVaR(β) model with transaction costs (3.1)-(3.8) introduced in Section 3.2, and will refer to that model as the *basic model*.
Some comments are worthwhile on how the transaction costs can be modelled. The transaction costs have to be introduced in the minimum required return constraint (3.3) to guarantee a minimum net return rate of the portfolio. On the contrary, transaction costs may be included in the budget constraint (3.4) or not, depending on whether the capital C is used both to buy the securities and to pay the transaction costs or to buy the securities only. Both modelling alternatives may be interesting to a decision maker. We have chosen the latter that has the advantage of clearly dividing the capital invested in the portfolio and the money spent in transaction costs. On the other side, in the former alternative, constraint (3.4) should be modified into

$$\sum_{j=1}^{n}\{(1+c_j)q_jX_j + f_jz_j\} = C.$$

We carried out some computational experiments to compare the impact of the two modelling alternatives on the portfolio composition. The results have shown that the portfolios obtained in the two cases are practically identical.

5.2.2 The Portfolio Rebalancing CVaR(β) Model

The single-period CVaR(β) model with transaction costs described in the previous section assumes an investor who follows a buy-and-hold investment strategy. The investor allocates the capital

among n available securities by constructing a portfolio to be kept over the whole investment horizon. On the contrary, we now assume an investor who may decide to rebalance her/his portfolio composition to take into account new market information even if this implies additional transaction costs. Let us consider two different investment periods. At the end of the first period (time 0) the investor selects the portfolio and keeps it until the end of the second investment period (time 1) when she/he may decide to rebalance it. The decision variables X_j^1, $j \in N$, represent the fractional number of stocks selected in the portfolio after rebalancing. The parameters X_j^0, $j \in N$, represent the optimal portfolio selected at time 0 (portfolio composition before rebalancing). To correctly represent a portfolio rebalancing problem we need to consider both a fixed cost f_j for each security j sold or purchased and a proportional cost c_j. The latter is applied to the absolute difference between the amount invested in security j in the rebalanced portfolio and in the initial one

$$c_j q_j |X_j^1 - X_j^0|,$$

where q_j represents the quotation of security *j* at the rebalancing time. To linearize such expression, we introduce a non-negative variable, δ_j, for each security $j \in N$, and the two following constraints

$$\delta_j \geq (X_j^1 - X_j^0),$$

$$\delta_j \geq -(X_j^1 - X_j^0).$$

At the optimum $\delta_j, j \in N$, will take the value $|X_j^1 - X_j^0|$. Finally, to correctly model the fixed cost $f_j, j \in N$, we use the binary variable z_j, $j \in N$, that has here a sligthly different role with respect to the basic model. z_j takes value 1 if the investor buys or sells a fraction of security *j* and 0 otherwise, i.e.

$$z_j = \begin{cases} 1 & \text{if} \quad (X_j^1 - X_j^0) > 0 \quad \text{or} \quad (X_j^1 - X_j^0) < 0, \\ 0 & \text{otherwise.} \end{cases}$$

Notice that, with respect to the basic model, new upper bounds u_j^1 on the fractional number of stock units that can be purchased for each security $j, j \in N$, are defined. For sake of simplicity, only the decision variables X_j^1 and the upper bounds u_j^1 have been indexed at time 1, since they have a slightly different meaning than in the basic model. Anyway, one may notice that all the parameters, e.g. the set of probabilities p_t and the set of quotation q_j, and all the variables, e.g. non-negative variables d_t and δ_j, of the following optimization model are defined with respect to time 1.

The *rebalancing model* can be formulated as follows

CVaR(β) Rebalancing model

$$\max \quad \eta - \frac{1}{\beta} \sum_{t=1}^{T} p_t d_t \tag{5.1}$$

$$\text{subject to} \quad \eta - \sum_{j=1}^{n} r_{jt} q_j X_j^1 + \sum_{j=1}^{n} c_j q_j \delta_j + \sum_{j=1}^{n} f_j z_j \leq d_t \quad t = 1, ..., T \tag{5.2}$$

$$\sum_{j=1}^{n} r_j q_j X_j^1 - \sum_{j=1}^{n} c_j q_j \delta_j - \sum_{j=1}^{n} f_j z_j \geq \mu_0 \sum_{j=1}^{n} q_j X_j^1 \tag{5.3}$$

$$\sum_{j=1}^{n} q_j X_j^1 = \sum_{j=1}^{n} q_j X_j^0 + \alpha \tag{5.4}$$

$$z_j \geq \frac{(X_j^1 - X_j^0)}{u_j^1} \quad j = 1, ..., n \tag{5.5}$$

$$z_j \geq -\frac{(X_j^1 - X_j^0)}{u_j^1} \quad j = 1, ..., n \tag{5.6}$$

$$X_j^1 \leq u_j^1 z_j \quad j = 1, ..., n \tag{5.7}$$

$$\delta_j \geq (X_j^1 - X_j^0) \quad j = 1, ..., n \tag{5.8}$$

$$\delta_j \geq -(X_j^1 - X_j^0) \quad j = 1, ..., n \tag{5.9}$$

$$d_t \geq 0 \quad t = 1, ..., T \tag{5.10}$$

$$X_j^1 \geq 0 \qquad j = 1, ..., n \tag{5.11}$$

$$\delta_j \geq 0 \qquad j = 1, ..., n \tag{5.12}$$

$$z_j \in \{0, 1\} \quad j = 1, ..., n. \tag{5.13}$$

As for the basic model, the objective function (5.1), along with constraints (5.2) and (5.10), determines the maximization of the safety measure and the definition of the non-negative variables $d_t, t = 1, ..., T$. The variables d_t measure the deviation from the $\beta-$quantile. Constraint (5.3) defines the mean portfolio return based on "buy-and-sell" proportional and fixed transaction costs and imposes that the net portfolio return has to be greater than or equal to the required return $\mu_0 \sum_{j=1}^{n} q_j X_j^1$, where $\sum_{j=1}^{n} q_j X_j^1$ is the amount invested in the portfolio at time 1. The budget constraint (5.4) requires that the capital has to be equal to the value of the initial portfolio at time 1 ($\sum_{j=1}^{n} q_j X_j^0$) plus a non-negative additional fund α, if available. As already explained, constraints (5.8) and (5.9) allow each variable δ_j to take the value of the absolute difference $|X_j^1 - X_j^0|$. Finally, constraints (5.5) and (5.6) force variable z_j to take value 1 when $|X_j^1 - X_j^0| \neq 0$, i.e. when the position of the security j in the portfolio has been changed. Note that if $X_j^1 - X_j^0 > 0$ (the investor increases her/his position in security j - buying case) then $\frac{(X_j^1 - X_j^0)}{u_j^1}$ is a positive value in the range $(0, 1]$ since u_j^1 is an upper bound on X_j^1 and X_j^0. Thus, constraints (5.5) force z_j to 1, whereas constraints (5.6) do not impose any real restriction requiring z_j to be larger than a negative quantity. On the contrary, if $X_j^1 - X_j^0 < 0$ (the investor reduces her/his position in security j - selling case) then constraints (5.6) force z_j to 1 whereas constraints (5.5) are not binding. In both cases where $z_j = 1$ the fixed cost f_j is included in the net mean return expression (constraint (5.3)) and in the objective function through constraints (5.2). When $X_j^1 = X_j^0$ then constraints (5.5) and (5.6) let z_j free to take any value. In such case, given the presence of the fixed cost in the objective function, z_j will take value 0.

5.3 Experimental Analysis

In this section we describe the computational results and compare the basic and the rebalancing models. Computational experiments have been conducted on a PC with a 2,400 MHz Intel Pentium processor and 512 Mb of RAM. The models have been implemented in C++ by means of the Concert Technology 2.0 and solved with CPLEX 9.0.
The results discussion is organized as follows. We first describe the data sets, then we introduce a static investment strategy and analyze its behavior versus different dynamic strategies in both in-sample and out-of-sample periods.

5.3.1 Testing Environment

Historical data are represented by daily rates of return computed by using stock prices taken from the German Stock Exchange Market (XETRA), the largest stock market in the EURO area. The rate of return of security j under scenario t has been computed as $r_{jt} = \frac{q_{jt}-q_{jt-1}}{q_{jt-1}}$, where q_{jt} is the closing price of security j in period t. No dividends have been considered. In order to consider all possible market trends we have constructed four data sets corresponding to different in-sample and out-of-sample time periods selected as follows. The first data set is characterized by a market trend going up in the in-sample period as well as in the out-of-sample one (*up-up data set*), the second data set by a market increasing in the in-sample period and decreasing in the out-of-sample one (*up-down data set*), the third data set by a market going down in the in-sample period and going up in the out-of-sample period (*down-up data set*) and, finally, the last set by a market going down in both the in-sample and the out-of-sample periods (*down-down data set*). Each of these data sets consists of 6 months of in-sample daily observations and 6 months of out-of-sample ones. The time periods covered by the four data sets are summarized in Table 5.1, whereas their temporal positioning is shown in Figure 5.1.
We have considered the 100 securities composing the XETRA DAX100 index at the date of April 1st, 2005. A security is included in the set of available alternatives only if it has been quoted with continuity on the market during the analyzed period. This means that the security has not been suspended for a period longer than 5 successive working days, and that the total number of days of suspension has not been altogether greater than 30 over the in-sample and out-of-sample periods. To reach the number of 100 securities we have substituted the excluded securities with other securities quoted on the German Stock Exchange Market, chosen following the alphabetical order. The reader should be aware that restricting attention to companies that belong to the market index during the entire period of the analysis may give rise to the so called *survivorship bias*. This means that the selected portfolios performance may be overestimated due to the fact that only companies that were successful enough to survive until the end of the period are included.

5.3.2 Dynamic versus Static Strategies: A Computational Comparison

In this section we introduce the investment strategy of a *static investor* and compare it to the investment strategies of more *dynamic investors* who decide to periodically rebalance their portfolios. We define as *BS*, i.e. Basic Strategy, the buy-and-hold strategy implemented by the investor who is not interested in dynamically modify its portfolio when market trend changes but quietly waits for

	in-sample (130 daily observations)		*out-of-sample* (130 daily observations)	
data set	begin	end	begin	end
up-up	01/08/1996	30/01/1997	31/01/1997	31/07/1997
up-down	18/09/2001	19/03/2002	20/03/2002	17/09/2002
down-up	11/09/2002	12/03/2003	13/03/2003	10/09/2003
down-down	09/10/2000	09/04/2001	10/04/2001	08/10/2001

Table 5.1: The four data sets.

the end of the investment period. The main advantage of this static strategy is the lower amount of incurred transaction costs. The Basic Strategy implies only one optimization corresponding to solve the basic model at the date of portfolio selection.
On the contrary, a dynamic investor is more sensitive to market changes and accepts to pay additional transaction costs in exchange for a possibly higher portfolio return or a lower portfolio loss. We analyze different levels of dynamism according to the number of times the investor decides to rebalance her/his portfolio. We define as *RS*k the Rebalancing Strategy where the initial optimal portfolio composition is re-optimized k times. More precisely, in a RSk strategy the initial optimal portfolio is selected running the basic model, while each of the successive k re-optimizations is obtained by solving the rebalancing model at a different future date.
We have considered four different RSk strategies corresponding to $k = 1, 2, 3$ and 5. In the first strategy the investor creates the initial portfolio at time 0 running the basic model on the 6 months in-sample observations, and re-optimizes the portfolio with the rebalancing model only once ($k = 1$) three months after the selection of the initial portfolio. In the second strategy a more dynamic investor decides to rebalance her/his portfolio twice ($k = 2$) exactly every two months after the initial portfolio optimization. In the third strategy the investor rebalances three times ($k = 3$) after one month and a half, after 3 months and after 4 months and a half from the initial portfolio optimization. Finally, in the last strategy, the investor rebalances the portfolio every month, that is 5 times ($k = 5$). The rebalancing model is always solved taking into account only three months of in-sample realizations instead of the six moths realizations considered in the basic model. The choice is consistent with the behavior of a dynamic investor who pays more attention to the most recent information provided by the market. As an example, in Figure 5.2 we show the in-sample and the out-of-sample periods for the BS strategy and for the RS5 strategy with respect to the *down-up* data set.
In the following, we will denote as *BS(β)* the BS strategy used by the investor running the basic model with a quantile parameter equal to β. Similarly, we will denote as *RSk(β)* the RSk strategy where the initial optimal portfolio and the successive rebalancing optimizations are obtained running the models with the quantile parameter equal to β.
For each data set we have implemented and run the BS(β) strategy and the RSk(β) strategies by considering an initial capital equal to 100,000 Euros, three different levels of minimum required return μ_0 (0, 0.05 and 0.10 on yearly basis) and four different values of the quantile parameter β (0.01, 0.05, 0.10 and 0.25). We have solved each model considering a fixed cost equal to 12 Euros and a proportional cost for buying (in the basic model) and for buying/selling (in the rebalancing model) equal to the 0.195% of the amount invested. These are real conditions that Italian brokers

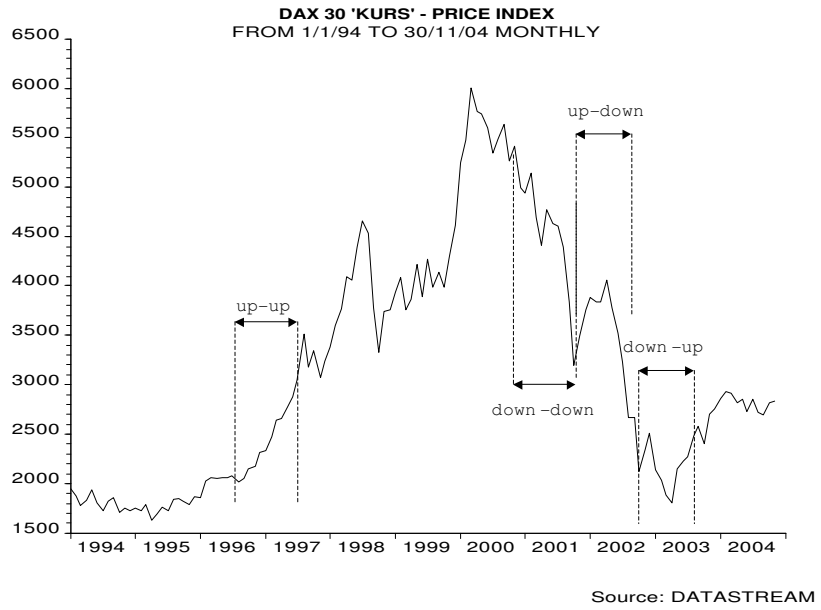

Figure 5.1: The four different market periods.

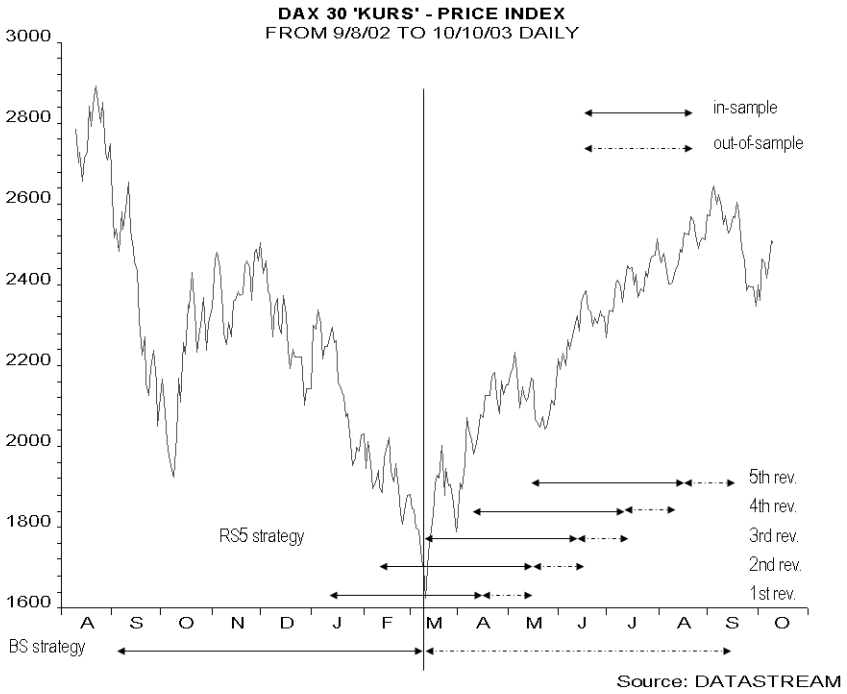

Figure 5.2: An in-sample and out-of-sample graphical comparison between the BS(β) and the RS5(β) strategies in the down-up data set.

apply for operations carried out on the German market. Moreover, in our experiments the upper bounds u_j for the basic model are calculated as

$$u_j = \frac{C}{q_j} \quad j = 1, \ldots, n. \tag{5.14}$$

The same quantities for the rebalancing model are calculated considering the capital available at the time of rebalancing

$$u_j^1 = \frac{\sum_{s=1}^{n} q_s X_s^0 + \alpha}{q_j} \quad j = 1, ..., n. \tag{5.15}$$

In our experiments we have assumed α=0.

5.3.3 In-Sample Analysis

In the following we present the characteristics of the portfolios selected by the investors using the BS(β) and the RSk(β) strategies. Since similar results have been obtained for different values of μ_0 and of β, we have decided to only report the results obtained by requiring a minimum rate of return equal to 5% on a yearly basis and a level of the quantile parameter equal to 0.01 and to 0.05. All the remaining results can be found in Guastaroba *et al.* [78].
Each of the following tables consists of nine columns showing the objective function value (obj.), the portfolio per cent average return on yearly basis including the transaction costs paid by the investor (net return) and without considering the transaction costs (gross return), the number of securities selected by the model (div.), the minimum and the maximum shares within the portfolio, the total transaction costs paid by the investor divided into proportional and fixed transaction costs. The tables for the RSk(β) strategies contain three additional columns (indicated as "# purchase", "# sale", "cumulative costs") representing the number of securities purchased and the number of securities sold in the current rebalancing, and the $cumulative$ transaction costs obtained as sum of the transaction costs paid in the current rebalancing and in all the previous ones.
In Table 5.2 we show the complete computational results obtained for the BS(β) strategy implemented in all the four different data sets. The large difference between net and gross portfolio rate

	obj.	net	gross		shares		prop.	fixed	total
instances	10^2	return	return	div.	min	max	costs	costs	costs
up-up									
BS(0.01)	-14.297	5.00	162.61	5	0.056	0.440	195	60	255
BS(0.05)	-12.106	5.00	186.24	7	0.072	0.256	195	84	279
up-down									
BS(0.01)	-14.033	5.00	255.01	12	0.012	0.249	195	144	339
BS(0.05)	-13.678	5.00	255.01	12	0.037	0.152	195	144	339
down-down									
BS(0.01)	-29.782	5.00	140.93	3	0.172	0.443	195	36	231
BS(0.05)	-27.166	5.00	151.54	4	0.125	0.332	195	48	243
down-up									
BS(0.01)	-16.736	5.00	198.84	8	0.014	0.405	195	96	291
BS(0.05)	-15.484	5.00	198.84	8	0.018	0.408	195	96	291

Table 5.2: BS(β) strategy with μ_0 = 5%: Optimal portfolio characteristics.

of return is due to the presence of transaction costs and to the formula adopted to convert the rates

of return on yearly basis. The number of securities in the portfolios (div.) is lower than in other experiments carried out on the same model (see for instance Mansini *et al.* [112]). This can be explained by the presence of fixed transaction costs that increase proportionally to the number of securities selected. One can also observe the higher incidence of proportional transaction costs with respect to fixed ones on the total transaction costs. In terms of computational times, the basic model requires only few seconds, rarely more than one minute to find the optimal solution.
In Tables 5.3-5.6 we show the results obtained for the RS$k(\beta)$ strategies. We compare the portfolios obtained by the different strategies at approximately the same date. At this aim we have chosen the unique revision of the dynamic investor who rebalances the portfolio once (after 3 months), the second revision of that who rebalances twice (this means after 4 months), the third revision of the investor who rebalances three times (this means after 4 months and a half), and the fourth of the investor who rebalances five times (this means after 4 months). One may notice that, whereas in the basic model the portfolio net return is always equal to the required one, in the rebalancing model the portfolio net rate of return may yield a larger value. This is especially true for the *up-up* and *down-up* data sets and for the investor who rebalances five times. This can be explained by the fact that the in-sample observations belong to a data set where the market was increasing. For the *down-up* data set and the RS$5(\beta)$ strategy see Figure 5.2. The number of securities selected seems to increase with the number of rebalances. In particular, when the market is increasing (i.e. *up-up* and *down-up* data sets) and when new information is provided by the market, the investor purchases more securities than she/he sells. For thoroughness, we have also shown the cumulative cost paid by the investor until the current rebalance. Obviously, this cost increases with the number of rebalances.
By comparing Tables 5.3-5.6, one can see that the objective function value usually increases with the number of rebalances. This is an expected result since to operate more rebalances means to benefit from newer information.

instances	obj. 10^2	net return	gross return	div.	shares min	shares max	# purchase	# sale	prop. costs	fixed costs	total costs	cumulative costs
up-up												
RS1(0.01)	-12.515	5.00	170.56	9	0.027	0.464	5	3	230.80	96	326.80	581.80
RS1(0.05)	-10.368	5.00	195.35	10	0.026	0.270	5	4	233.35	108	341.35	620.35
up-down												
RS1(0.01)	-21.642	5.00	99.91	9	0.010	0.268	2	4	94.79	72	166.79	505.79
RS1(0.05)	-22.375	5.00	169.43	8	0.036	0.268	2	6	137.03	96	233.03	572.03
down-down												
RS1(0.01)	-10.535	5.00	226.85	9	0.026	0.333	6	3	213.93	108	321.93	552.93
RS1(0.05)	-9.390	5.00	197.09	10	0.043	0.292	6	3	192.48	108	300.48	543.48
down-up												
RS1(0.01)	-7.948	91.88	569.14	14	0.001	0.406	9	6	343.78	180	523.78	814.78
RS1(0.05)	-7.001	55.00	424.82	15	0.004	0.409	8	6	249.46	168	417.46	708.46

Table 5.3: RS1(β) strategy with μ_0 = 5%: Optimal portfolio characteristics.

5.3.4 Out-of-Sample Analysis

In the ex-post analysis, we have examined the behavior of all the portfolios selected by using the BS(β) strategy and all the RS$k(\beta)$ strategies for the six months following the date of the initial

instances	obj. 10^2	net return	gross return	div.	shares min	shares max	# purchase	# sale	prop. costs	fixed costs	total costs	cumulative costs
up-up												
RS2(0.01)	-9.194	17.71	123.74	14	0.005	0.246	6	3	116.73	108	224.73	914.36
RS2(0.05)	-8.797	8.58	102.86	13	0.002	0.299	4	4	113.70	96	209.70	804.46
up-down												
RS2(0.01)	-25.421	5.00	270.47	13	0.003	0.186	3	7	165.58	120	285.58	757.41
RS2(0.05)	-24.980	5.00	117.28	10	0.017	0.141	2	4	89.88	72	161.88	650.07
down-down												
RS2(0.01)	-8.191	5.77	126.71	8	0.009	0.237	3	2	152.29	60	212.29	696.53
RS2(0.05)	-11.251	6.31	106.15	8	0.032	0.384	4	1	125.17	60	185.17	647.45
down-up												
RS2(0.01)	-8.007	5.00	299.59	15	0.003	0.351	6	7	290.36	156	446.36	1148.96
RS2(0.05)	-7.598	5.00	336.33	17	0.009	0.297	7	7	283.70	168	451.70	1056.36

Table 5.4: RS2(β) strategy with μ_0 = 5%: Optimal portfolio characteristics of the second revision.

instances	obj. 10^2	net return	gross return	div.	shares min	shares max	# purchase	# sale	prop. costs	fixed costs	total costs	cumulative costs
up-up												
RS3(0.01)	-7.396	14.23	147.09	18	0.003	0.220	5	6	154.23	132	286.23	1334.47
RS3(0.05)	-6.908	5.00	119.37	14	0.002	0.214	3	6	144.72	108	252.72	1155.13
up-down												
RS3(0.01)	-21.411	5.00	358.26	11	0.005	0.145	4	6	204.17	120	324.17	1022.18
RS3(0.05)	-20.751	5.00	273.81	13	0.011	0.169	4	5	171.19	108	279.19	957.98
down-down												
RS3(0.01)	-10.750	5.00	79.13	9	0.009	0.360	2	4	75.30	72	147.30	887.38
RS3(0.05)	-9.565	8.52	57.45	14	0.003	0.308	3	2	39.80	60	99.80	780.30
down-up												
RS3(0.01)	-7.653	10.96	132.14	21	0.003	0.197	7	3	150.21	120	270.21	1320.20
RS3(0.05)	-7.853	5.00	107.19	19	0.006	0.382	5	4	121.87	108	229.87	1237.89

Table 5.5: RS3(β) strategy with μ_0 = 5%: Optimal portfolio characteristics of the third revision.

portfolio optimization.

For each model and data set we have computed a table including nine ex-post parameters defined as follows: the number of times out of 130 the portfolio rate of return outperforms the corresponding required rate of return (#), the average portfolio rate of return (r_{av}), the median (r_{med}) of the returns, the standard deviation (std) and the semi-standard deviation (s-std), the mean absolute deviation (MAD) and the mean downside deviation (s-MAD), the maximum downside deviation (D-DEV) and the Sortino index. The latter provides a measure of the over-performance of the portfolio mean rate of return with respect to the required one per unit of downside risk (measured by the semi-standard deviation).

Such performance criteria are used to compare the out-of-sample behavior of the portfolios selected by using the BS(β) strategy and the RSk(β) strategies, respectively. The average return and the median are expressed on yearly basis. All the dispersion measures (std, s-std, MAD, s-MAD and D-DEV) and the Sortino index have been computed with respect to the minimum required rate of return to make them directly comparable in the different models.

In Figures 5.3–5.6 the portfolio performances have been analyzed and compared in terms of cumulative returns. This allows us to consider the evolution of the portfolio value over time, and allows a complete comparison between the portfolios selected by using the BS(β) strategy with respect to those selected by using the RSk(β) strategies.

We first comment Tables 5.7 and 5.8, and then Figures 5.3–5.6. In Tables 5.7 and 5.8 we show

instances	obj. 10^2	net return	gross return	div.	shares min	shares max	# purchase	# sale	prop. costs	fixed costs	total costs	cumulative costs
up-up												
RS5(0.01)	-6.935	103.03	146.81	13	0.001	0.204	3	1	14.97	48	62.97	1190.99
RS5(0.05)	-6.800	37.85	132.06	18	0.006	0.174	4	4	74.42	96	170.42	1347.27
up-down												
RS5(0.01)	-18.594	5.00	91.78	16	0.005	0.157	2	5	53.15	84	137.15	1103.99
RS5(0.05)	-20.899	5.00	42.60	20	0.003	0.148	1	2	33.64	36	69.64	817.44
down-down												
RS5(0.01)	-7.774	6.91	121.85	13	0.014	0.238	3	6	96.38	108	204.38	1090.93
RS5(0.05)	-8.484	9.95	113.60	11	0.009	0.367	3	5	89.72	96	185.72	1044.59
down-up												
RS5(0.01)	-7.140	39.14	360.91	20	0.010	0.256	7	5	269.30	144	413.30	1560.26
RS5(0.05)	-5.472	38.47	250.43	19	0.005	0.254	8	3	162.84	132	294.84	1406.32

Table 5.6: RS5(β) strategy with μ_0 = 5%: Optimal portfolio characteristics of the fourth revision.

all the performance measures and the dispersion parameters we have computed. In particular, we compare the BS(β) strategy with the RSk(β) strategies, and show their performances with respect to the market behavior of the DAX30 index. It is worth observing that when the market index is decreasing in the ex-post period (see the *up-down* and *down-down* data sets), in terms of average returns all the models dominate the market index. This is in particular true for the *up-down* data set where all the models show an average negative return significantly larger than the market index performance with the gap almost equal to 20% (see also Figure 5.4).

One may also notice that in the same data sets the *Sortino* index computed on DAX30 is greater than that obtained using the BS(β) and the RSk(β) strategies. Nevertheless, only following the *Sortino* index indications may be misleading. In fact, a strategy characterized by higher downside risk and worse negative average return may show a better *Sortino* index than another strategy that shows a better risk measure and a better average return. For this reason we mediate the *Sortino* index with the cumulative returns.

Tables 5.7 and 5.8 show that the downside risk incurred by the market index, in particular when the out-of-sample period is *down* (i.e. in the *down-down* and the *up-down* data sets), is greater than that incurred by using both the BS(β) and the RSk(β) strategies. This is also true when the out-of-sample period is *up* (i.e. in the *up-up* and the *down-up* data sets) and can be seen by comparing the risk incurred by the DAX30 and that incurred by the RSk(β) strategies. This confirms that portfolio optimization models allow a reduction of the risk incurred by the investor.

One may also notice that the risk, and in particular the downside risk represented by the s-std and the s-MAD columns, tends to decrease with the number of revisions. This result can be explained considering the attempt of the rebalancing model to reduce the downside risk incurred by the portfolio at each revision. We have also observed that, in the *up-down* data set, the RS5(β) strategy cannot find a feasible integer solution in the fifth revision. In Tables 5.7 and 5.8 we have used symbol "*" to indicate such occurrence.

In Figures 5.3–5.6 we present the *cumulative returns* in the four different market periods with a minimum required rate of return equal to 5% on yearly basis and a quantile parameter β = 0.05. In all the figures the DAX30 shows a cumulative return extremely unstable. See, for instance, Figure 5.3 where the market index reaches a cumulative return that is greater than that of all the portfolio optimization models in the first 27 ex-post realizations. Later it decreases reaching the worst cumulative return from period 27 to period 57. Then, from period 57 to 120 it yields a cumulative

model	#	$r_a v$	r med	std	s-std	MAD	s-MAD	D-DEV	Sortino index
up-up									
BS(0.01)	66	196.29	0	0.0191	0.0098	0.0128	0.0049	0.0516	0.0596
RS1(0.01)	66	149.18	10.83	0.0166	0.0080	0.0105	0.0041	0.0396	0.0872
RS2(0.01)	71	162.87	58.55	0.0157	0.0075	0.0096	0.0035	0.0396	0.1151
RS3(0.01)	68	187.58	33.04	0.0154	0.0069	0.0091	0.0032	0.0396	0.1650
RS5(0.01)	67	77.58	10.56	0.0109	0.0062	0.0078	0.0032	0.0326	0.0865
dax30	84	186.89	321.93	0.0117	0.0072	0.0091	0.0031	0.0390	0.1463
up-down									
BS(0.01)	55	-50.63	-79.91	0.0131	0.0102	0.0101	0.0061	0.0341	-0.0147
RS1(0.01)	55	-46.99	-58.17	0.0143	0.0108	0.0113	0.0066	0.0330	-0.0112
RS2(0.01)	56	-55.96	-57.91	0.0136	0.0106	0.0108	0.0066	0.0339	-0.0140
RS3(0.01)	55	-51.30	-51.43	0.0137	0.0104	0.0104	0.0063	0.0444	-0.0139
RS5(0.01)	*	*	*	*	*	*	*	*	*
dax30	49	-73.83	-85.18	0.0248	0.0185	0.0189	0.0114	0.0584	-0.0011
down-down									
BS(0.01)	58	-34.18	3192.81	0.0146	0.0118	0.0101	0.0057	0.0911	-0.0058
RS1(0.01)	57	-31.78	-37.70	0.0134	0.0104	0.0098	0.0055	0.0694	-0.0092
RS2(0.01)	62	-27.12	-7.55	0.0109	0.0083	0.0080	0.0045	0.0363	-0.0175
RS3(0.01)	60	-27.18	-19.74	0.0120	0.0094	0.0086	0.0048	0.0571	-0.0116
RS5(0.01)	62	-36.92	-1.70	0.0108	0.0088	0.0079	0.0046	0.0411	-0.0189
dax30	60	-49.23	-29.52	0.0187	0.0149	0.0133	0.0077	0.0850	-0.0026
down-up									
BS(0.01)	75	425.45	-99.46	0.0232	0.0135	0.0161	0.0058	0.0608	0.0334
RS1(0.01)	82	486.93	156.14	0.0149	0.0050	0.0095	0.0023	0.0224	0.9666
RS2(0.01)	77	167.91	120.30	0.0111	0.0058	0.0077	0.0026	0.0340	0.2334
RS3(0.01)	82	198.04	159.04	0.0105	0.0052	0.0072	0.0022	0.0339	0.3542
RS5(0.01)	79	146.60	142.39	0.0104	0.0057	0.0075	0.0025	0.0296	0.2073
dax30	70	273.33	98.95	0.0202	0.0119	0.0154	0.0059	0.0615	0.0390

Table 5.7: Out-of-sample statistics for $\mu_0 = 0.05$ and $\beta = 0.01$.

return only better than that yielded using the RS5(β) strategy and finally it has a strong increase but it does not even reach the final cumulative return obtained by the BS(β) strategy. The high volatility of index ex-post returns is also confirmed by the dispersion measures showed in Tables 5.7 and 5.8.

In Figure 5.6 we show the ex-post cumulative returns for the *down-up* data set. In this data set the market index reaches a cumulative return always better than that yielded by all the optimal portfolios. By comparing the figures in Table 5.8 it is evident that it incurs a higher level of downside risk. As a consequence, the DAX30 has the worst value for *Sortino* index, clearly lower than those associated to the optimal portfolios. In particular, still referring to the *down-up* data set, we have observed that through an appropriate choice of the quantile parameter value (i.e. $\beta = 0.01$) the cumulative return reached by the RS1(β) strategy is better than that reached by the market index, with a lower level of downside risk (see the average rate of return and the downside risk measures shown in Table 5.7).

All computational results show that the risk management obtained by means of a portfolio optimization model provides a good strategy when the market is increasing and becomes of crucial importance when the market is decreasing. Figures 5.4 and 5.5 show the cumulative returns associated to the market index and to the optimal portfolios in the *up-down* and in the *down-down* data sets, respectively. In particular, in Figure 5.4 the DAX30 has, on average, a cumulative return clearly worse than that obtained by the optimal portfolios. After the first 7 periods, where the portfolios selected by the optimization models and the DAX30 follow the same trend, the market index approximately yields the worst cumulative returns in all the remaining ex-post periods. This

model	#	r av	r med	std	s-std	MAD	s-MAD	D-DEV	Sortino index
up-up									
BS(0.05)	72	177.48	0	0.0134	0.0078	0.0099	0.0036	0.0402	0.1126
RS1(0.05)	72	144.17	103.59	0.0112	0.0066	0.0081	0.0029	0.0402	0.1443
RS2(0.05)	76	107.18	91.98	0.0106	0.0066	0.0079	0.0030	0.0402	0.1045
RS3(0.05)	77	125.69	130.43	0.0102	0.0062	0.0079	0.0029	0.0348	0.1470
RS5(0.05)	76	127.45	106.42	0.0105	0.0061	0.0073	0.0026	0.0385	0.1517
dax30	84	186.89	321.93	0.0117	0.0072	0.0091	0.0031	0.0390	0.1463
up-down									
BS(0.05)	48	-54.01	-70.41	0.0126	0.0099	0.0098	0.0060	0.0352	-0.0177
RS1(0.05)	53	-54.70	-50.97	0.0142	0.0110	0.0111	0.0067	0.0326	-0.0118
RS2(0.05)	54	-52.75	-49.02	0.0136	0.0104	0.0105	0.0064	0.0435	-0.0141
RS3(0.05)	51	-43.83	-50.84	0.0140	0.0102	0.0104	0.0061	0.0424	-0.0130
RS5(0.05)	*	*	*	*	*	*	*	*	*
dax30	49	-73.83	-85.18	0.0248	0.0185	0.0189	0.0114	0.0584	-0.0011
down-down									
BS(0.05)	60	-31.75	69731.25	0.0129	0.0100	0.0087	0.0049	0.0631	-0.0105
RS1(0.05)	58	-39.33	-21.31	0.0128	0.0098	0.0090	0.0052	0.0473	-0.0137
RS2(0.05)	64	-28.01	0	0.0117	0.0088	0.0083	0.0047	0.0406	-0.0145
RS3(0.05)	60	-33.07	-20.39	0.0110	0.0083	0.0082	0.0047	0.0403	-0.0203
RS5(0.05)	58	-44.09	-37.47	0.0127	0.0104	0.0090	0.0054	0.0625	-0.0120
dax30	60	-49.23	-29.52	0.0187	0.0149	0.0133	0.0077	0.0850	-0.0026
down-up									
BS(0.05)	77	143.32	-90.16	0.0135	0.0084	0.0098	0.0038	0.0406	0.0723
RS1(0.05)	78	165.68	115.90	0.0096	0.0045	0.0069	0.0022	0.0175	0.4025
RS2(0.05)	80	113.14	94.11	0.0097	0.0056	0.0072	0.0026	0.0316	0.1653
RS3(0.05)	82	139.27	119.23	0.0091	0.0051	0.0064	0.0021	0.0371	0.2548
RS5(0.05)	78	87.36	82.29	0.0084	0.0052	0.0060	0.0022	0.0362	0.1500
dax30	70	273.33	98.95	0.0202	0.0119	0.0154	0.0059	0.0615	0.0390

Table 5.8: Out-of-sample statistics for $\mu_0 = 0.05$ and $\beta = 0.05$.

is also true for all the other experiments carried out in the *up-down* and in the *down-down* data sets (Figure 5.5), for different values of μ_0 and of β. The DAX30 shows not only worse cumulative returns than those associated to the portfolios selected by using both the BS(β) and the RSk(β) strategies, but also a higher volatility in the ex-post periodic returns. This can be noticed by analyzing Figures 5.4 and 5.5 and also by comparing Tables 5.7 and 5.8.

The first aim of this analysis is to understand if the portfolio optimization management represents a valid tool for financial decisions, and then to understand if any *dynamic strategy* (i.e. RSk(β) strategy) is better than the *static strategy* (i.e. BS(β) strategy). The second aim is to provide some general guidelines about the best strategy to follow on the basis of the information available in the in-sample period.

As a conclusion, we can say that it is evident that portfolio optimization models represent valuable tools for a rational risk averse investor. Moreover, in the *up-down* and in the *down-down* data sets, the downside risk incurred by the market index is clearly higher than that of all the optimal portfolios, with the worst average return and the worst cumulative returns.

Similar conclusions can be drawn for the *up-up* and the *down-up* data sets. The use of a portfolio optimization model, and in particular the use of a RSk(β) strategy, determines a strong reduction of the instability of the portfolio return. Only in some experiments the cumulative return associated to the market index is better than that reached by an optimal portfolio. If the quantile parameter β is set equal to 0.01 the cumulative return associated to the BS(β) and the RS1(β) strategies are, in almost all the ex-post periods, better than that yielded by the market index.

With respect to a comparison between *static* and *dynamic* strategies we can draw some conclusions

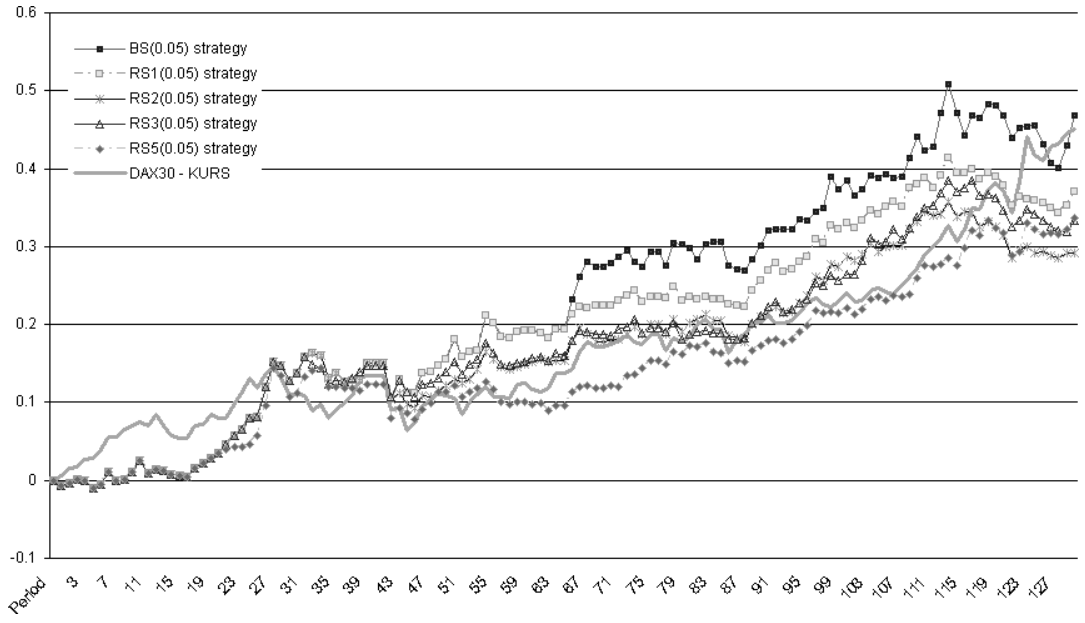

Figure 5.3: Cumulative returns (up-up data set): A comparison between portfolio optimization models and the market index.

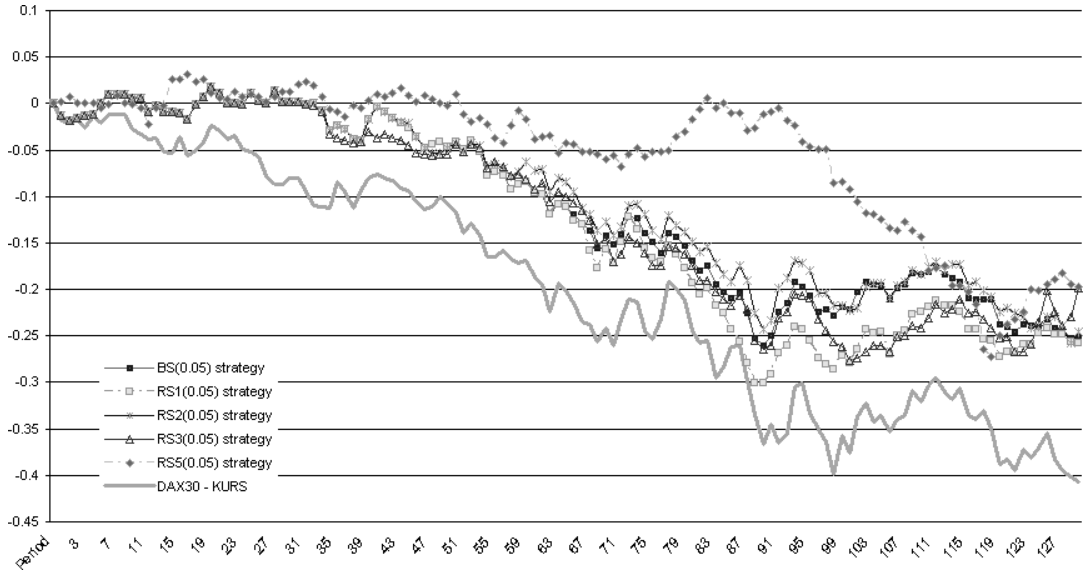

Figure 5.4: Cumulative returns (up-down data set): A comparison between portfolio optimization models and the market index.

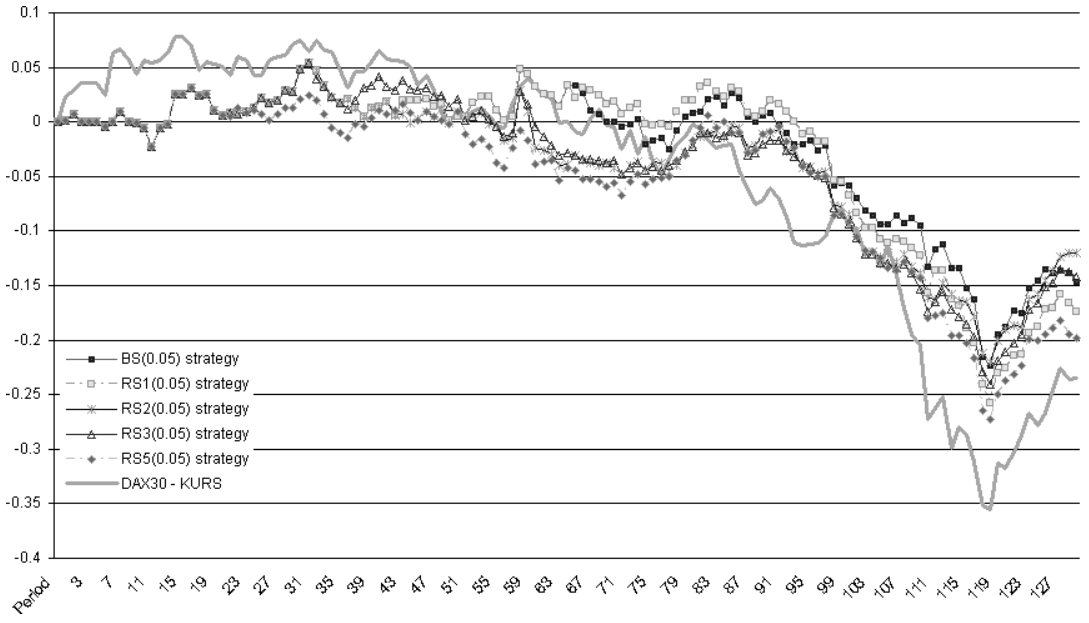

Figure 5.5: Cumulative returns (down-down data set): A comparison between portfolio optimization models and the market index.

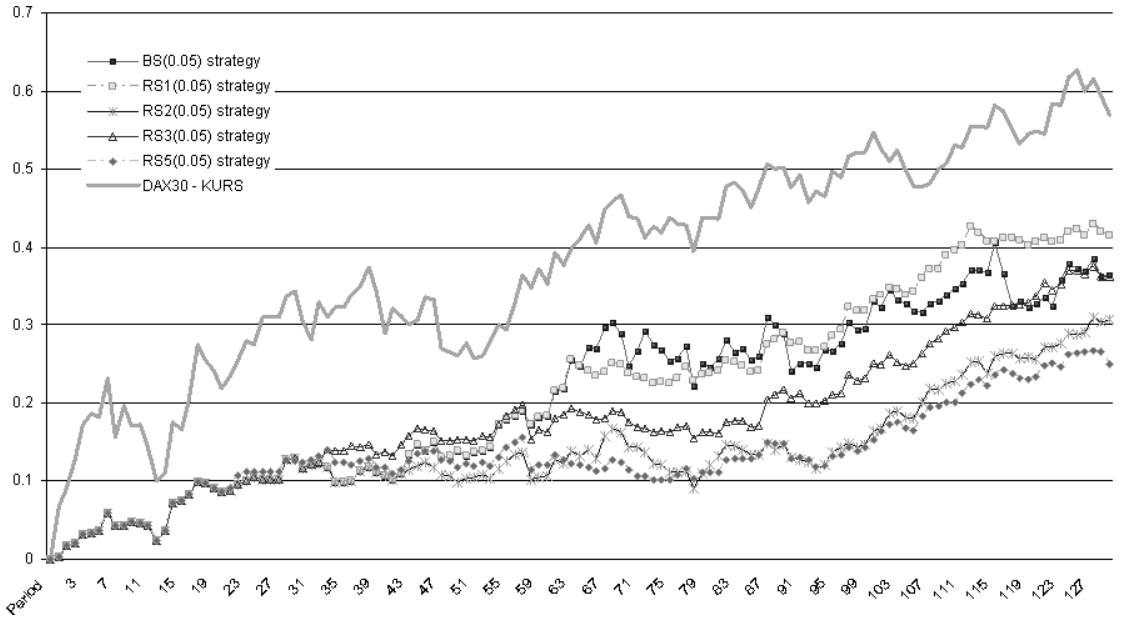

Figure 5.6: Cumulative returns (down-up data set): A comparison between portfolio optimization models and the market index.

by observing the market trend and the portfolio performances. If the aim of the investor is to minimize the portfolio return instability, rebalancing is the best strategy to follow. By increasing the number of rebalances the investor tends to reduce the incurred downside risk. This is the correct strategy for a *very risk averse* investor. If the investor is *less risk averse* and willing to take more risk to possibly yield a higher rate of return we can draw some other conclusions by observing the market trend. If the market is going *up* the best choice is to implement a BS(β) or RS1(β) strategy (i.e. see Figures 5.3 and 5.6). On the contrary, when the market is going *down* the investor best choice is to implement a RSk(β) strategy, and in particular the RS2(β) or the RS3(β) strategy (i.e. see Figures 5.4 and 5.5).
Obviously, nobody knows what will happen in the future at the time the portfolio is chosen. For this reason we try to draw some guidelines about the strategy to follow for the investor on the basis of the information available in the in-sample period. If the market is increasing in the in-sample period, the *very risk averse* investor would like to hedge her/his investments from a possibly dramatic fall in market quotations during the out-of-sample period. Thus, as already pointed out, this investor should choose to implement the RS2(β) or the RS3(β) strategy in the *up* period and also in the *down* period being the two strategies which can better hedge the investment in case of a possible market fall. On the contrary, a *less risk averse* investor is willing to take more risk in order to possibly gain from an increasing market trend, thus she/he should implement a BS(β) or RS1(β) strategy because these are the two choices that yield the higher average and cumulative returns.

5.3.5 Impact of the Amount of Transaction Costs

In order to evaluate the impact of the amount of the transaction costs on the portfolio, we have compared the portfolios obtained with different amounts of the transaction costs. We have considered a foreign investor and a local one, who incur in different transaction costs. The former pays the transaction costs considered in Section 5.3.2, whereas the latter pays half of that amount (i.e. fixed costs equal to 6 Euros and proportional costs equal to 0.195%). Both investors implement a rebalancing strategy where the initial portfolio is re-optimized 5 times, with $\beta = 0.05$ and $\mu_0 = 5\%$. The test has been carried out on the down-down data set.
The first row of Table 5.9 corresponds to the initial portfolio, while the successive rows to the re-optimized portfolios. The two central parts of Table 5.9 report information on the portfolios obtained by the foreign investor and by the local one. For each of the two cases, the number of securities selected (div.) and the total amount of fixed plus proportional transaction costs (total costs) are shown. In the far right column of Table 5.9 the number of securities selected by both investors (# common securities) is shown. Comparison of the portfolios obtained by the two investors and the small number of common securities show that the portfolio is very sensitive to the amount of the transaction costs.

In Figure 5.7 we have compared the out-of-sample net cumulative returns of the portfolios obtained by the two investors and the market index. As expected, the portfolio obtained with lower transaction costs has a better out-of-sample behavior.

	transaction costs		*(transaction costs)/2*		
	div.	total costs	div.	total costs	# common securities
BS(0.05)	4	243.00	5	127.50	2
1^{st} rev. RS5(0.05)	7	216.70	6	110.12	4
2^{nd} rev. RS5(0.05)	8	134.53	7	78.79	4
3^{rd} rev. RS5(0.05)	12	264.64	9	195.16	3
4^{th} rev. RS5(0.05)	11	185.72	12	153.97	4
5^{th} rev. RS5(0.05)	13	120.54	12	160.47	4
cumulative costs		1165.13		826.01	

Table 5.9: Impact of the amount of transaction costs (down-down data set).

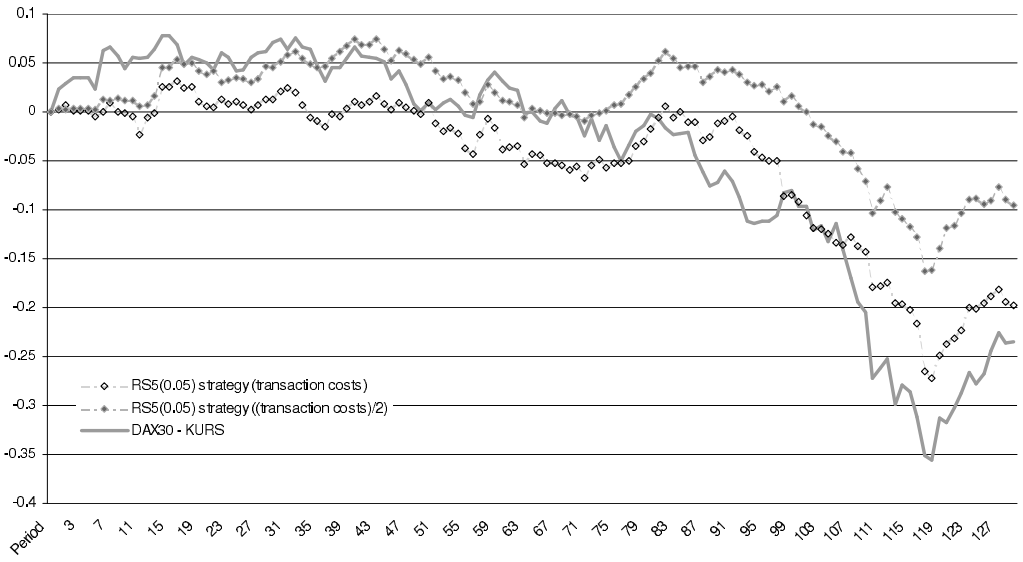

Figure 5.7: Cumulative returns (down-down data set): Impact of the amount of transaction costs.

5.4 Conclusions

In this chapter we have shown how to modify a buy-and-hold optimization model to provide dynamic portfolio optimization strategies. Extensive computational results based on real-life data from German Stock Exchange Market have allowed us to draw some interesting conclusions, the most important of which is that portfolio optimization models represent effective tools to help the investor in making financial decisions. More precisely, different models can be used according to the level of risk aversion of the investor. For a *very risk averse* investor the best choice in order to hedge the portfolio is to implement a RS$k(\beta)$ strategy and to rebalance two or three times in 6 months. On the other side, a *less risk averse* investor should prefer to invest through the BS(β) or the RS1(β) strategy.
Finally, in order to evaluate how side-constraints affect the portfolio optimization, we have shown that fixed transaction costs determine a reduced number of securities selected in the portfolio with respect to the case in which such costs are ignored.

6

Index Tracking and Enhanced Index Tracking

6.1 Replicating and Out-Performing a Market Index: Literature Review

Fund management involves the investment of capital on equities issued by companies quoted in stock markets. The aim of a fund manager is to significantly make the initial capital grow over the medium/long term. Even if the possible alternatives are many, the strategies implemented by most of the fund managers can be basically classified as follows.

a) *Passive management.* The fund manager has low degree of flexibility and is expected to conform to a set of criteria. A common criterion is that the fund should reproduce (*track*) the performances of a stock market index selecting properly a subset of the stocks composing the index. This strategy is referred to as *index tracking*.

b) *Active management.* The fund manager has a high degree of flexibility in choosing the portfolio composition. The fundamental assumption is that the expertise and the judgment of the fund manager permit to select the most promising stocks and/or the best timing of stocks purchase and sale. The aim is usually that of out-performing a stock market index. Particularly, a common goal is to track a fictitious index whose return is, over time, above the return achieved by the market index. This strategy is referred to as *enhanced index tracking*.

The previous broad classification considers only pure strategies. However, mixed strategies, whereby a portion of the fund is managed passively and the remaining part is managed actively, are also possible. Moreover, pure passive and active management have strengths and weaknesses. Specifically, passive management usually implies low fixed management costs (associated with the payment to the management team) and low transaction costs, but it has the disadvantage that if the market index drops, and so most of the securities that compose the index, inevitably the fund return drops as well. Conversely, active management usually implies high fixed management and transaction costs. High transaction costs are due to the frequent trading implied by this strategy. However, if the fund is successfully managed, these costs are offset by the return achieved. Furthermore, in passive management the portfolio is only exposed to market risk, whereas in active management the portfolio can be exposed to both market and idiosyncratic risk, i.e. company risk. The more intuitive way to track an index is full replication, i.e. building a portfolio composed

of the same stocks that constitute the market index in the exact same proportion. However, full replication has several disadvantages. Firstly, certain stocks are present in very small quantities in the market index, and therefore in the managed portfolio. The purchase of these small quantities can imply the payment of high transaction costs. Secondly, when the underlying composition of the market index is revised, the transaction costs implied by the purchase and sale of stocks, done in order to replicate again the index, may be high. For instance, index revision can occur when the fall of the market capitalization of a stock is so sharp that another stock merits the inclusion in the index. Mergers are another reason why an index is revised. Thirdly, when indices are composed by many stocks, the stocks issued by small companies are usually relatively illiquid, and this implies high transaction costs for buying and selling. Finally, transaction costs need to be considered in the stocks selection model. On the contrary, full replication provides no means of limiting transaction costs so as to prevent them becoming excessive. Because of all the previous disadvantages, many passively managed funds, especially those that track indices constituted by many stocks, hold fewer stocks than those included in the index they are tracking [37], and a number of fund managers is implementing an active management strategy. Therefore, though currently passive management is the most used strategy in fund management business, and index tracking is in particular (see [62]), an increasing number of funds are managed by means of an active management strategy (see [76] and [108]). Typical examples of benchmarks that fund managers usually seek to track or beat are the FTSE100 for equity funds, the Standard and Poors 500 Index (S&P500) for mutual funds, the Goldman Sachs Commodity Index for commodities funds.

In the domain of index tracking models, it is worth mentioning the contributions by Worzel *et al.* [147] who develop an integrated simulation and optimization approach for tracking fixed-income indices. To this aim, and drawing on the MAD portfolio optimization model [97], the authors introduce an LP formulation that penalizes downside deviations of the portfolio return from the index. An analogous model is implemented and tested in Consiglio and Zenios [41] tracking a composite index of the international bond market. Finally, we mention the model introduced by Konno and Wijayanayake in [95]. To this model we dedicate Section 6.2.1.

Despite several papers concerning index tracking appeared in the literature (see [29] and references therein), the area of enhanced index tracking is still relatively unexplored. Enhanced index tracking is sometimes referred to as enhanced indexation, or as "index-plus-alpha". The latter phrase is based on a linear regression based view of enhanced index tracking, where a linear regression of the *enhanced portfolio* return, i.e. the enhanced index tracking portfolio return, is performed against the return achieved by the market index. The alpha represents the regression intercept. Therefore, the higher the alpha, the better the enhanced portfolio return.

Among the principal contributions on enhanced index tracking, Beasley *et al.* [8] propose a generalized formulation for the index tracking problem, where the decision maker is allowed to control the trade-off between index tracking and excess return, i.e. enhanced index tracking. The trade-off is managed through a parameter in the objective function that weighs one objective against the other. Their formulation explicitly includes transaction costs (associated with buying or selling stocks), a limit on the total transaction cost that can be incurred, and a cardinality constraint on the number of stocks that can be purchased. A similar idea is developed in Fang and Wang [67] where the authors propose a bi-objective programming problem. In their paper, the index tracking error is minimized whereas the excess return is maximized. Furthermore, based on fuzzy decision theory, the authors propose an index tracking portfolio selection model and give a numerical example to illustrate its behavior. Dose and Cincotti [54] propose a mathematical formulation similar to the

model introduced in [8], and adopt a two-step solving approach. In the first step, the selection process limits the number of stocks in the tracking portfolio by considering a subset of stocks designed to statistically represent the index. Secondly, the weight of each stock in the portfolio is determined as the result of an optimization process. The selection process is performed by means of a clustering technique that relies on a distance function defined between time series of stock prices. The optimization process makes use of the allowed list of stocks and builds the tracking portfolio as a result of the proposed objective function minimization. Wu *et al.* [148] formulate the enhanced index tracking problem as a dual-criteria goal programming problem. One goal relates to track a benchmark index closely, whereas the other goal concerns the use of risk-controlled strategies to add modest value to the index. Canakgoz and Beasley [29] consider both problems of tracking and out-performing a market index. The authors consider two mixed-integer linear programming formulations that adopt a regression based view of index tracking and enhanced indexation, respectively. Then, they propose a three-stage solution procedure for the index tracking problem, and a two-stage solution approach for enhanced indexation.

In the remainder of this section, we survey the main contributions appeared in the literature on algorithms proposed to solve single-period portfolio optimization problems.

Chang *et al.* [33] extend the standard Markowitz model to include a cardinality constraint and limits on the proportion of the portfolio held in a given asset. The authors present three heuristics, based upon Genetic, Tabu Search and Simulated Annealing metaheuristics, for finding the efficient frontier. Konno and Wijayanayake [95] consider the minimal cost index tracking problem under non-linear and non-convex transaction costs and propose a branch-and-bound algorithm to solve this class of problems. Beasley *et al.* [8] present an evolutionary heuristic, i.e. a Genetic Algorithm, for solving the index tracking problem with a limit on the total transaction cost that can be paid and a constraint limiting the number of stocks that can be held in portfolio. Schaerf [138] adds to the basic Mean-Variance model additional constraints on the cardinality of the portfolio and on the quantity of individual shares. The author uses Tabu Search algorithms to solve the problem. Gilli and Këllezi [70] investigate the performance of the Threshold Accepting heuristic for the index tracking problem. Derigs and Nickel [48] describe the concept of designing a metaheuristic based decision support system generator for portfolio optimization problems, and, among them, for an index tracking problem. The metaheuristic considered is a Simulated Annealing algorithm. Maringer and Kellerer [115] consider a variant of the Markowitz model that include a cardinality constraint on the number of securities composing the portfolio. The authors suggest a hybrid local search algorithm which combines principles of Simulated Annealing and Genetic Algorithms. Derigs and Nickel [49] describe and implement a 2-phase Simulated Annealing heuristic approach for the problem of optimizing a stock fund with respect to tracking error and transaction costs over time subject to a set of constraints. Oh *et al.* [128] propose a Genetic Algorithm to solve the index tracking problem and show an application to the Korean stock market. Konno and Hatagi [94] propose a scheme to construct an index-plus-alpha portfolio with minimal transaction costs. The problem is formulated as a concave minimization under linear constraints, which is solved by means of a branch-and-bound algorithm. Maringer and Oyewumi [116] consider the index tracking problem with a cardinality constraint and lower and upper limits on the portfolio weights to avoid excessive fragmentation of single assets. They present an empirical study based on the solution of the problem by means of a differential evolution metaheuristic. Finally, an overview on the use of metaheuristics for the portfolio selection problems can be found in di Tollo and Roli [51].

It is worthwhile mentioning the contribution by Di Gaspero *et al.* [50] where exact methods are

used as components of heuristics for the Markowitz model with a cardinality constraint and a minimal and maximal share limiting each stock included in the portfolio. The authors employ a quadratic programming solver in a problem decomposition approach in which a metaheuristic searches in the space of assets only for a subset of assets to be included in the portfolio, and at each iteration the solver determines the optimal allocation over that subset. The heuristics that are implemented and compared are Steepest Descent, First Descent and Tabu Search. A similar idea is developed in Moral-Escudero *et al.* [120] where a Genetic Algorithm is implemented as metaheuristic.
The idea of developing heuristics identifying a small/moderate-size subset of variables in order to intensify the search in a promising region of the solution space has also been used in other contexts. In the Knapsack Problem family, for instance, Balas and Zemel [7] propose the idea of selecting a small subset of items (called the core) and to solve exactly a restricted problem on that subset. The most evident weakness of the former and other related methods concerns the fact that the core composition is not known in advance. To partially overcome this problem, Pisinger [133] proposes the use of an expanding method to modify the size of the core during the algorithm execution. In the domain of routing problems, MILP problems considering only a subset of the decision variables are solved to optimality by De Franceschi *et al.* [46] and Archetti *et al.* [4]. Finally, drawing on the idea of exhaustively exploring promising portions of the solution space, Angelelli *et al.* [3] propose a heuristic framework, called *Kernel Search*, for the solution of MILP problems.
The promising results obtained by these works indicate that identifying good portions of the solution space, and then thoroughly exploring them, can improve performances in terms of both computational time and solution quality.
Structure of the Chapter. In Section 6.2 we briefly recall the index tracking model formulation proposed by Konno and Wijayanayake [95]. Subsequently, and drawing on their model, we propose the mathematical formulations for the index tracking and the enhanced index tracking problems. Section 6.3 introduces an enhanced version of the kernel search framework when applied to the index tracking problem. Section 6.4 provides the computational analysis of the heuristics implemented solving the index tracking problem. Finally, in Section 6.5 some conclusions are drawn.

6.2 The Models

6.2.1 The Konno and Wijayanayake Model

Konno and Wijayanayake [95] formulate the minimal transaction costs index tracking problem as a linearly constrained concave minimization problem. Let $\tilde{q}_j$, $j = 1, \ldots, n$, be the random variable representing the price of asset j. Let us assume that each random variable $\tilde{q}_j$ is distributed over a set of finitely many points of cardinality T. In the following, we denote as q_{jt}, $j = 1, \ldots, n$, the realization of the random variable $\tilde{q}_j$ under scenario t, where $t = 1, \ldots, T$. Moreover, let I_t, $t = 1, \ldots, T$, represents the value of the benchmark index under scenario t. Let X_j, $j = 1, \ldots, n$, be the units of asset j included in the portfolio, restricted to be non-negative, i.e. $X_j \geq 0$, $j = 1, \ldots, n$. Therefore, the value of the portfolio under scenario t is given by $\sum_{j=1}^{n} q_{jt} X_j$. Assume that the investor has an initial capital available for the investment equal to C under scenario T, i.e. the time of portfolio selection. Moreover, the authors assume that under scenario T the value

of the portfolio should be equal to the total investment C, i.e. $\sum_{j=1}^{n} q_{jT}X_j = C$. Konno and Wijayanayake measure the tracking error between the portfolio and the market index using the absolute deviation

$$\sum_{t=1}^{T} |\theta I_t - \sum_{j=1}^{n} q_{jt}X_j|, \tag{6.1}$$

where $\theta = \frac{C}{I_T}$. Let $x_j = q_{jT}\frac{X_j}{C}$ be the investment in security j at the time of portfolio construction, and denote as $c(x_j)$ the transaction costs associated. Specifically, the authors define the transaction cost as a non-linear and non-convex function that they subsequently approximate by a piecewise linear function.
Their model minimizes the total transaction costs incurred while keeping the tracking error within some given tolerance ε. The mathematical formulation proposed is the following non-linear model

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c(x_j) \\ \text{subject to} \quad & \sum_{t=1}^{T} |\theta I_t - \sum_{j=1}^{n} q_{jt}X_j| \leq \varepsilon C \\ & \sum_{j=1}^{n} q_{jT}X_j = C \\ & X_j \geq 0 \quad j = 1, \ldots, n. \end{aligned} \tag{6.2}$$

Subsequently, model (6.2) is linearized by means of the introduction of two non-negative variables Y_t and Z_t, $t = 1, \ldots, T$, satisfying the following conditions

$$Y_t - Z_t = \theta I_t - \sum_{j=1}^{n} q_{jt}X_j \quad t = 1, \ldots, T$$

$$Y_t Z_t = 0, Y_t \geq 0, Z_t \geq 0 \quad t = 1, \ldots, T.$$

The introduction of the previous decision variables allows the following representation

$$Y_t + Z_t = |\theta I_t - \sum_{j=1}^{n} q_{jt}X_j| \quad t = 1, \ldots, T.$$

Finally, the authors prove that model (6.2) is equivalent to the following linearly constrained concave minimization problem

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c(x_j) \\ \text{subject to} \quad & \sum_{t=1}^{T} (Y_t + Z_t) \leq \varepsilon C \end{aligned}$$

$$Y_t - Z_t = \theta I_t - \sum_{j=1}^{n} q_{jt} X_j \quad t = 1, \ldots, T \tag{6.3}$$

$$\sum_{j=1}^{n} q_{jT} X_j = C$$

$$X_j \geq 0 \quad j = 1, \ldots, n$$

$$Y_t \geq 0, Z_t \geq 0 \quad t = 1, \ldots, T.$$

In a similar way, in Konno and Hatagi [94] the enhanced index tracking problem is modelled. Drawing on the former formulations, and specifically the way of modelling the tracking error, we designed the model for index tracking that is introduced in the next section. Subsequently, the model for enhanced index tracking is described.

6.2.2 The Index Tracking Model

Let us assume that the investor holds a portfolio of securities (hereafter referred to as the *current portfolio*), and let the parameters X_j^0, $j = 1, \ldots, n$, represent the units of asset j included in the current portfolio. Let us assume that we computed T scenarios. In each scenario t, $t = 1, \ldots, T$, the value (quotation) of security j, denoted as q_{jt}, as well as the value of the index we are tracking, denoted as I_t, are computed. We are interested in selecting at scenario T the optimal portfolio composition that minimizes the in-sample tracking error, rebalancing the current portfolio composition. The decision variables X_j^1, $j = 1, \ldots, n$, represent the fractional number of stocks selected in the portfolio after rebalancing. We assume that short sales are not allowed, i.e. both X_j^0 and X_j^1, $j = 1, \ldots, n$, are constrained to be non-negative.
Let C be the capital available for investment at scenario T. It is equal to the value of the current portfolio ($W_T = \sum_{j=1}^{n} q_{jT} X_j^0$) plus cash change ($\tau$), that can be either an additional fund or a withdrawal, i.e. $C = \sum_{j=1}^{n} q_{jT} X_j^0 + \tau$. The investor is allowed to either purchase or sell stocks in order to rebalance the portfolio composition. In case the investor buys $X_j^1 - X_j^0$ units of security j, i.e. $X_j^1 - X_j^0 > 0$, the investor incurs in a transaction cost that is proportional to the value of security j that has been bought, i.e. $c_j^b (X_j^1 - X_j^0) q_{jT}$ where c_j^b is the proportional transaction cost paid when buying security j. In the opposite case, i.e. $X_j^0 - X_j^1 > 0$, the investor incurs in a transaction cost that is proportional to the value of security j that has been sold, i.e. $c_j^s (X_j^0 - X_j^1) q_{jT}$ where c_j^s is the proportional transaction cost paid when selling security j. Additionally, in our formulation each time a security is traded (either bought or sold with respect to the current portfolio) a fixed transaction cost, denoted as f_j, $j = 1, \ldots, n$, is paid by the investor. Let the non-negative variable G_j, $j = 1, \ldots, n$, represent the transaction costs incurred in purchasing or selling security j. Furthermore, we assume that if security j is selected in the optimal solution, its proportional value with respect to the capital available for the investment must be within a lower bound, denoted as ϵ_j, and an upper bound, denoted as δ_j. To this aim, we introduce a binary variable z_j, $j = 1, \ldots, n$, that takes value 1 if the investor holds any fraction of security j and 0 otherwise.
Our approach focuses on the normalized trajectory $I_t / I_{\bar{t}}$, $t = 1, \ldots, T$, of market index values where $\bar{t}$ denotes some reference time period. The former trajectory is then compared with the corresponding trajectory of portfolio values $W_t = \sum_{j=1}^{n} q_{jt} X_j^1$, $t = 1, \ldots, T$ to compute the

tracking error. Specifically, we set $\bar{t} = T$ and employ the absolute deviation to measure the tracking error between the market index and the new portfolio

$$TrE(\mathbf{X}) = \sum_{t=1}^{T} |\theta I_t - \sum_{j=1}^{n} q_{jt} X_j^1|$$

where the parameter $\theta = C/I_T$ is used to scale the value of the index to be tracked. Intuitively, the parameter θ represents the number of stocks the investor can buy at scenario T of a fictitious security whose quotation is equal to I_T.
Our index tracking model aims at minimizing function $TrE(\mathbf{X})$. Without any loss of generality we can introduce and minimize the non-negative variable ε, and constrain function $TrE(\mathbf{X})$ to be less than or equal to εC, i.e. $TrE(\mathbf{X}) \leq \varepsilon C$.
A first step that will lead to the linearization of the tracking error function is the introduction of two non-negative variables d_t and u_t, $t = 1, \ldots, T$, satisfying the following conditions

1. $d_t - u_t = (\theta I_t - \sum_{j=1}^{n} q_{jt} X_j^1)$, $t = 1, \ldots, T$,
2. $d_t u_t = 0$, $t = 1, \ldots, T$,
3. $d_t \geq 0$, $u_t \geq 0$, $t = 1, \ldots, T$,

that lead to the following representation

$$d_t + u_t = |\theta I_t - \sum_{j=1}^{n} q_{jt} X_j^1|, t = 1, \ldots, T. \tag{6.4}$$

Let us now consider variables G_j, $j = 1, \ldots, n$. To deal with a proportional transaction cost for buying securities different from that for selling securities, we need to introduce two further non-negative variables b_j and s_j, $j = 1, \ldots, n$, that satisfy the following conditions

$$b_j = \begin{cases} (X_j^1 - X_j^0) q_{jT} & \text{if} \quad X_j^1 - X_j^0 \geq 0, \\ 0 \quad \text{otherwise} \end{cases}$$

and

$$s_j = \begin{cases} (X_j^0 - X_j^1) q_{jT} & \text{if} \quad X_j^1 - X_j^0 < 0, \\ 0 \quad \text{otherwise.} \end{cases}$$

Let us constrain the two variables b_j and s_j, $j = 1, \ldots, n$, to satisfy the following conditions

1. $b_j - s_j = (X_j^1 - X_j^0) q_{jT}$, $j = 1, \ldots, n$,
2. $b_j s_j = 0$, $j = 1, \ldots, n$,
3. $b_j \geq 0$, $s_j \geq 0$, $j = 1, \ldots, n$.

Since either b_j or s_j can take a value greater than zero, or none of the two (condition 2), if the investor buys new stocks of security j, i.e. $X_j^1 - X_j^0 > 0$, then b_j takes value equal to the value of stocks purchased. Conversely, if the investor sells any stock of security j, i.e. $X_j^1 - X_j^0 < 0$, then s_j takes value equal to the value of stocks sold.
To correctly model the fixed cost, we introduce the binary variable w_j, $j = 1, \ldots, n$. Variable w_j takes value 1 if the investor buys or sells any fraction of security j, and 0 otherwise, i.e.

$$w_j = \begin{cases} 1 & \text{if} \quad (X_j^1 - X_j^0) > 0 \text{ or } (X_j^1 - X_j^0) < 0 \\ 0 & \text{otherwise.} \end{cases}$$

For each security j, the variable G_j, can be defined as follows

$$G_j = c_j^b b_j + c_j^s s_j + f_j w_j. \tag{6.5}$$

Moreover, we constrain the total transaction costs paid by the investor to be less than or equal to a proportion γ of the capital available for investment under scenario T

$$\sum_{j=1}^{n} G_j \leq \gamma C. \tag{6.6}$$

In order to maintain a low portfolio cardinality, we constrain the number of securities selected in portfolio to be less than or equal to a certain threshold k. Actually, even the cardinality constraint requires the introduction of a set of binary decision variables. As we mentioned before, we assume that z_j, $j = 1, \ldots, n$, takes value 1 if security j is included in the optimal portfolio, and 0 otherwise, i.e.

$$z_j = \begin{cases} 1 & \text{if} \quad X_j^1 > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Accordingly, the cardinality constraint can be formulated as follows

$$\sum_{j=1}^{n} z_j \leq k. \tag{6.7}$$

Consider the following (non-linear) formulation for the index tracking problem

(NL)IT Model

$$\text{minimize} \quad \varepsilon$$

$$\text{subject to} \quad \sum_{j=1}^{n} q_{jT} X_j^1 \leq C$$

$$\epsilon_j z_j \leq X_j^1 q_{jT}/C \leq \delta_j z_j \quad j = 1, \ldots, n$$

$$d_t - u_t = (\theta I_t - \sum_{j=1}^{n} q_{jt} X_j^1) \quad t = 1, \ldots, T$$

$$d_t u_t = 0 \quad t = 1, \ldots, T$$
$$\sum_{t=1}^{T} (d_t + u_t) \leq \varepsilon C$$
$$b_j - s_j = (X_j^1 - X_j^0) q_{jT} \quad j = 1, \ldots, n$$
$$b_j s_j = 0 \quad j = 1, \ldots, n$$
$$b_j + s_j \leq \delta_j C w_j \quad j = 1, \ldots, n$$
$$(6.5) \quad - \quad (6.7)$$
$$X_j^1, b_j, s_j, G_j \geq 0 \quad j = 1, \ldots, n$$
$$d_t, u_t \geq 0 \quad t = 1, \ldots, T$$
$$z_j \in \{0, 1\} \quad j = 1, \ldots, n$$
$$w_j \in \{0, 1\} \quad j = 1, \ldots, n.$$

Theorem 1 *Problem (NL)IT can be equivalently formulated as the following MILP problem*

IT Model

$$\textit{minimize} \quad \varepsilon \tag{6.8}$$
$$\textit{subject to} \quad \sum_{j=1}^{n} q_{jT} X_j^1 \leq C \tag{6.9}$$
$$\epsilon_j z_j \leq X_j^1 q_{jT} / C \leq \delta_j z_j \quad j = 1, \ldots, n \tag{6.10}$$
$$d_t - u_t = (\theta I_t - \sum_{j=1}^{n} q_{jt} X_j^1) \quad t = 1, \ldots, T \tag{6.11}$$
$$\sum_{t=1}^{T} (d_t + u_t) \leq \varepsilon C \tag{6.12}$$
$$b_j - s_j = (X_j^1 - X_j^0) q_{jT} \quad j = 1, \ldots, n \tag{6.13}$$
$$b_j + s_j \leq \delta_j C w_j \quad j = 1, \ldots, n \tag{6.14}$$
$$G_j = c_j^b b_j + c_j^s s_j + f_j w_j \quad j = 1, \ldots, n \tag{6.15}$$
$$\sum_{j=1}^{n} G_j \leq \gamma C \tag{6.16}$$
$$\sum_{j=1}^{n} z_j \leq k \tag{6.17}$$
$$X_j^1, b_j, s_j, G_j \geq 0 \quad j = 1, \ldots, n \tag{6.18}$$
$$d_t, u_t \geq 0 \quad t = 1, \ldots, T \tag{6.19}$$
$$z_j \in \{0, 1\} \quad j = 1, \ldots, n \tag{6.20}$$
$$w_j \in \{0, 1\} \quad j = 1, \ldots, n. \tag{6.21}$$

Proof. The minimization of the objective function (6.8) along with constraint (6.12) force either variable d_t or variable u_t to take the value of the tracking error under scenario t in constraint (6.11). Indeed, since the two variables are constrained to be non-negative by means of constraints (6.19), if under any scenario t the market index takes a value greater than the portfolio, i.e. $\theta I_t > \sum_{j=1}^{n} q_{jt} X_j^1$, then $d_t = \theta I_t - \sum_{i=1}^{n} q_{jt} X_j^1$ by means of constraint (6.11). Conversely, if under any scenario t the portfolio takes a value greater than the index tracked, i.e. $\theta I_t < \sum_{j=1}^{n} q_{jt} X_j^1$, then $u_t = \sum_{j=1}^{n} q_{jt} X_j^1 - \theta I_t$.
Let $(X_1^{1*}, \ldots, X_n^{1*}, b_1^*, \ldots, b_j^*, \ldots, b_n^*, s_1^*, \ldots, s_j^*, \ldots, s_n^*, G_1^*, \ldots, G_n^*, z_1^*, \ldots, z_n^*, d_1^*, \ldots, d_T^*, u_1^*, \ldots, u_T^*)$ be an optimal solution of problem (6.8)-(6.21). Let us assume that there exists a security j such that $b_j^* s_j^* > 0$, then it is always possible finding a feasible solution of problem (6.8)-(6.21) such that $b_j^* s_j^* = 0$. Let $(\check{b}_j, \check{s}_j) = (b_j^* - s_j^*, 0)$ if $b_j^* \geq s_j^*$, and let $(\check{b}_j, \check{s}_j) = (0, s_j^* - b_j^*)$ if $s_j^* > b_j^*$. Then $(X_1^{1*}, \ldots, X_n^{1*}, b_1^*, \ldots, \check{b}_j, \ldots, b_n^*, s_1^*, \ldots, \check{s}_j, \ldots, s_n^*, G_1^*, \ldots, G_n^*, z_1^*, \ldots, z_n^*, d_1^*, \ldots, d_T^*, u_1^*, \ldots, u_T^*)$ is a feasible solution of problem (6.8)-(6.21). Moreover, it is an optimal solution since the objective function value has not changed. □

In more details, and without any loss of generality, let us assume that only security $\underline{j} \in N$ is selected in the optimal solution of problem (6.8)-(6.21). Furthermore, let us assume that at the optimum $b_{\underline{j}}^* s_{\underline{j}}^* > 0$ and that $b_{\underline{j}}^* > s_{\underline{j}}^*$. We want to show that the solution obtained substituting values $b_{\underline{j}}^*$ and $s_{\underline{j}}^*$ with values $\check{b}_{\underline{j}} := b_{\underline{j}}^* - s_{\underline{j}}^*$ and $\check{s}_{\underline{j}} := 0$, respectively, is still a feasible solution for problem (6.8)-(6.21). As a matter of fact, variables $b_{\underline{j}}$ and $s_{\underline{j}}$ appear in constraints (6.13), (6.14) and (6.15). Additionally, they indirectly affect constraint (6.16). Specifically, after the substitution, variable $w_{\underline{j}}$ in constraint (6.14) keeps taking value equal to 1. Constraint (6.13) is still satisfied, but now solely one of the two variables of the left hand side of the equation is greater than zero, specifically $\check{b}_{\underline{j}}$. This implies, by constraint (6.15), that $\check{G}_{\underline{j}} = c_{\underline{j}}^b \check{b}_{\underline{j}} + c_{\underline{j}}^s \check{s}_{\underline{j}} + f_{\underline{j}} w_{\underline{j}}^*$ takes a value smaller than $G_{\underline{j}}^* = c_{\underline{j}}^b b_{\underline{j}}^* + c_{\underline{j}}^s s_{\underline{j}}^* + f_{\underline{j}} w_{\underline{j}}^*$. As a consequence, constraint (6.16) is surely satisfied.
Constraint (6.9) is the budget constraint imposing that the capital invested is less than or equal to the principal available for the investment under scenario T. Constraints (6.10) ensure that if security j is selected in the rebalanced portfolio, i.e. $X_j^1 > 0$, then the associated binary variable z_j takes value 1. If this is the case, the proportional value of security j in the optimal portfolio is constrained to be within the lower and upper bounds defined. Additionally, if security j is not selected in the rebalanced portfolio, i.e. $X_j^1 = 0$, then variable z_j takes value equal to 0.
Finally, it is worth pointing out that a different mode of considering transaction costs could have been alternatively used. In our formulation, the transaction costs paid are constrained to be less than or equal to some threshold by constraint (6.16). Conversely, one could drop constraint (6.16) and introduce the sum of the transaction cost in the budget constraint (6.9), that would turn out to be

$$\sum_{j=1}^{n} q_{jT} X_j^1 \leq C - \sum_{j=1}^{n} G_j. \tag{6.22}$$

Another option, that has been considered by some authors, would be maintaining the transaction costs constraint (6.16) and introducing constraint (6.22).
The rational of our choice, similar to what we have assumed in Section 5.2, is motivated by the advantage of clearly dividing the principal invested in the portfolio and the capital spent in transaction costs.

The Index Tracking Model: Reduced Formulation

As a matter of fact, the specific structure of model (6.8)-(6.21) for index tracking allows several reductions in terms of the variables and constraints involved. Indeed, model (6.8)-(6.21) is equivalent to the following MILP formulation

IT Reduced Model

$$\text{minimize} \quad \sum_{t=1}^{T}(\theta I_t - \sum_{j=1}^{n} q_{jt}X_j^1 + 2u_t) \tag{6.23}$$

$$\text{subject to} \quad \sum_{j=1}^{n} q_{jT}X_j^1 \leq C \tag{6.24}$$

$$\epsilon_j z_j \leq X_j^1 q_{jT}/C \leq \delta_j z_j \quad j = 1, \ldots, n \tag{6.25}$$

$$(X_j^1 - X_j^0)q_{jT} + 2s_j \leq \delta_j C w_j \quad j = 1, \ldots, n \tag{6.26}$$

$$\sum_{j=1}^{n}[(X_j^1 - X_j^0)c_j^b q_{jT} + (c_j^b + c_j^s)s_j + f_j w_j] \leq \gamma C \tag{6.27}$$

$$\theta I_t - \sum_{j=1}^{n} q_{jt}X_j^1 + u_t \geq 0 \quad t = 1, \ldots, T \tag{6.28}$$

$$(X_j^1 - X_j^0)q_{jT} + s_j \geq 0 \quad j = 1, \ldots, n \tag{6.29}$$

$$\sum_{j=1}^{n} z_j \leq k \tag{6.30}$$

$$X_j^1, s_j \geq 0 \quad j = 1, \ldots, n \tag{6.31}$$

$$u_t \geq 0 \quad t = 1, \ldots, T \tag{6.32}$$

$$z_j \in \{0, 1\} \quad j = 1, \ldots, n \tag{6.33}$$

$$w_j \in \{0, 1\} \quad j = 1, \ldots, n. \tag{6.34}$$

In order to formulate model (6.23)-(6.34) we proceed as follows. Firstly, we utilize equations (6.11) to obtain the following explicit formulation for variables d_t

$$d_t = (\theta I_t - \sum_{j=1}^{n} q_{jt}X_j^1) + u_t \quad t = 1, \ldots, T.$$

Subsequently, we replace variables d_t in constraint (6.12) obtaining

$$\sum_{t=1}^{T}(\theta I_t - \sum_{j=1}^{n} q_{jt}X_j^1 + 2u_t) \leq \varepsilon C. \tag{6.35}$$

Note that this substitution allows the removal of T variables d_t and the set of T constraints (6.11), but implies the introduction of T new constraints (6.28) guaranteeing the non-negativity of the

expression replacing variables d_t. Then, instead of minimizing objective function (6.8), we equivalently minimize the left-hand side of inequality (6.35). This choice allows the removal of the latter constraint.
Furthermore, we utilize equation (6.13) to obtain the following representation for variable b_j

$$b_j = (X_j^1 - X_j^0)q_{jT} + s_j \quad j = 1, \ldots, n.$$

We then replace variables b_j in constraints (6.14) obtaining constraints (6.26). We also replace variables b_j in constraints (6.15), obtaining

$$G_j = (X_j^1 - X_j^0)c_j^b q_{jT} + (c_j^b + c_j^s)s_j + f_j w_j \quad j = 1, \ldots, n. \tag{6.36}$$

The former two substitutions allow the removal of n variables b_j, and n constraints (6.13), but require to introduce n new constraints (6.29).
Finally, we utilize equation (6.36) to replace variables G_j in constraint (6.16), obtaining inequality (6.27). This substitution allows the removal of n variables G_j and n constraints (6.15). It can be trivially proved that the remotion of variables G_j does not need the introduction of any further constraint.

6.2.3 The Enhanced Index Tracking Model

The non-linear formulation (NL)IT that models the index tracking problem can be modified in order to consider enhanced indexation, i.e. the problem of yielding excess return with respect to the return achieved by the market index chosen as benchmark. The enhanced index tracking problem can be formulated as the following non-linear model

(NL)EIT Model

$$\text{maximize} \quad \alpha$$

$$\text{subject to} \quad \sum_{j=1}^{n} q_{jT} X_j^1 \leq C$$

$$\epsilon_j z_j \leq X_j^1 q_{jT}/C \leq \delta_j z_j \quad j = 1, \ldots, n$$

$$d_t - u_t = (\theta I_t(1+\alpha) - \sum_{j=1}^{n} q_{jt} X_j^1) \quad t = 1, \ldots, T$$

$$d_t u_t = 0 \quad t = 1, \ldots, T$$

$$\sum_{t=1}^{T} (d_t + u_t) \leq \varepsilon C$$

$$b_j - s_j = (X_j^1 - X_j^0)q_{jT} \quad j = 1, \ldots, n$$

$$b_j s_j = 0 \quad j = 1, \ldots, n$$

$$b_j + s_j \leq \delta_j C w_j \quad j = 1, \ldots, n$$

$$G_j = c_j^b b_j + c_j^s s_j + f_j w_j \quad j = 1, \ldots, n$$

$$\sum_{j=1}^{n} G_j \leq \gamma C$$

$$\sum_{j=1}^{n} z_j \leq k$$

$$X_j^1, b_j, s_j, G_j \geq 0 \quad j = 1, \ldots, n$$

$$d_t, u_t \geq 0 \quad t = 1, \ldots, T$$

$$z_j \in \{0, 1\} \quad j = 1, \ldots, n$$

$$w_j \in \{0, 1\} \quad j = 1, \ldots, n.$$

The basic idea of the (NL)EIT model is to maximize the free variable α representing the excess return with respect to the market index, i.e. the yield above the market index return in each scenario $t = 1, \ldots, T$. A graphical explanation of this idea is provided in Figure 6.1. An important modification with respect to model (6.8)-(6.21) is that in formulation (NL)EIT ε becomes a parameter chosen by the decision-maker. Speaking intuitively, the enhanced index tracking model aims at maximizing the return above the market index return while maintaining the tracking error from the index-plus-alpha portfolio within a given tolerance ε.

A similar approach to that adopted for the index tracking problem can be used to formulate a

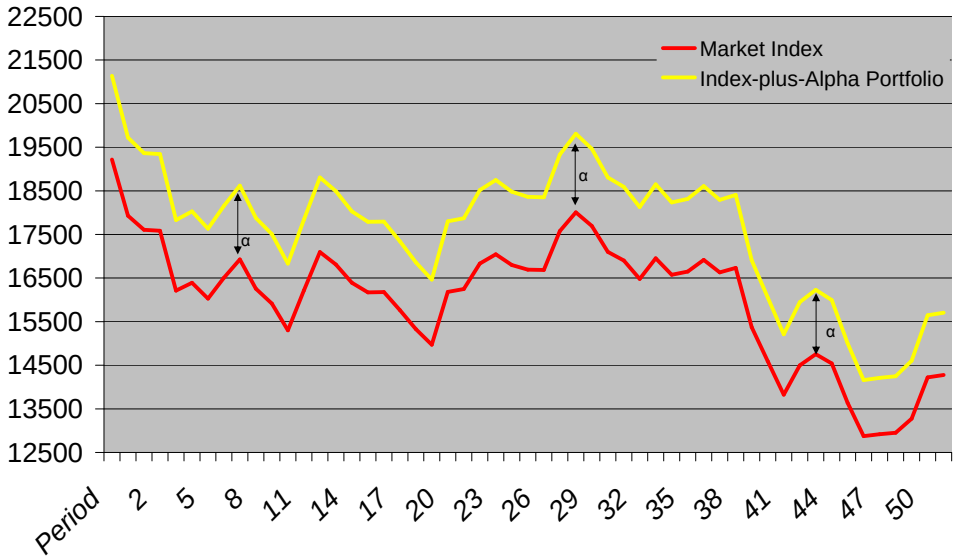

Figure 6.1: A comparison between the market index and the index-plus-alpha portfolio.

mixed integer linear programming formulation equivalent to model (NL)EIT. The MILP formulation for the enhanced index tracking is the following

EIT Model

$$\text{maximize} \quad \alpha \tag{6.37}$$

$$\text{subject to} \quad \sum_{j=1}^{n} q_{jT} X_j^1 \leq C \tag{6.38}$$

$$\epsilon_j z_j \leq X_j^1 q_{jT}/C \leq \delta_j z_j \quad j = 1, \ldots, n \tag{6.39}$$

$$d_t - u_t = (\theta I_t(1+\alpha) - \sum_{j=1}^{n} q_{jt} X_j^1) \quad t = 1, \ldots, T \tag{6.40}$$

$$\sum_{t=1}^{T} (d_t + u_t) \leq \varepsilon C \tag{6.41}$$

$$b_j - s_j = (X_j^1 - X_j^0) q_{jT} \quad j = 1, \ldots, n \tag{6.42}$$

$$b_j + s_j \leq \delta_j C w_j \quad j = 1, \ldots, n \tag{6.43}$$

$$G_j = c_j^b b_j + c_j^s s_j + f_j w_j \quad j = 1, \ldots, n \tag{6.44}$$

$$\sum_{j=1}^{n} G_j \leq \gamma C \tag{6.45}$$

$$\sum_{j=1}^{n} z_j \leq k \tag{6.46}$$

$$X_j^1, b_j, s_j, G_j \geq 0 \quad j = 1, \ldots, n \tag{6.47}$$

$$d_t, u_t \geq 0 \quad t = 1, \ldots, T \tag{6.48}$$

$$z_j \in \{0, 1\} \quad j = 1, \ldots, n \tag{6.49}$$

$$w_j \in \{0, 1\} \quad j = 1, \ldots, n. \tag{6.50}$$

In fact, let $(X_1^{1*}, \ldots, X_n^{1*}, b_1^*, \ldots, b_n^*, s_1^*, \ldots, s_n^*, G_1^*, \ldots, G_n^*, z_1^*, \ldots, z_n^*, d_1^*, \ldots, d_t^*, \ldots, d_T^*, u_1^*, \ldots, u_t^*, \ldots, u_T^*)$ be an optimal solution of problem (6.37)-(6.50) and assume that there exists a scenario t such that $d_t^* u_t^* > 0$. Then, an equivalent feasible solution is obtained substituting either $(\check{d}_t, \check{u}_t) = (d_t^* - u_t^*, 0)$ if $d_t^* \geq u_t^*$, or $(\check{d}_t, \check{u}_t) = (0, u_t^* - d_t^*)$ if $u_t^* > d_t^*$. As a consequence, solution $(X_1^{1*}, \ldots, X_n^{1*}, b_1^*, \ldots, b_n^*, s_1^*, \ldots, s_n^*, G_1^*, \ldots, G_n^*, z_1^*, \ldots, z_n^*, d_1^*, \ldots, \check{d}_t, \ldots, d_T^*, u_1^*, \ldots, \check{u}_t, \ldots, u_T^*)$ is feasible and also optimal since the objective function value has not changed. Analogously it can be proved that constraints $b_j s_j = 0$, $j = 1, \ldots, n$, are no longer necessary.

The Enhanced Index Tracking Model: Reduced Formulation

Similarly to the index tracking model, it is also possible modelling the enhanced index tracking problem by means of an equivalent reduced formulation. The reduced model is the following MILP formulation

EIT Reduced Model

$$\text{maximize} \quad \alpha$$

$$\text{subject to} \quad \sum_{j=1}^{n} q_{jT} X_j^1 \leq C$$

$$\epsilon_j z_j \leq X_j^1 q_{jT}/C \leq \delta_j z_j \quad j = 1, \ldots, n \tag{6.51}$$

$$\sum_{t=1}^{T} [\theta I_t (1+\alpha) - \sum_{j=1}^{n} q_{jt} X_j^1 + 2u_t] \leq \varepsilon C \tag{6.52}$$

$$(X_j^1 - X_j^0) q_{jT} + 2s_j \leq \delta_j C w_j \quad j = 1, \ldots, n \tag{6.53}$$

$$\sum_{j=1}^{n} [(X_j^1 - X_j^0) c_j^b q_{jT} + (c_j^b + c_j^s) s_j + f_j w_j] \leq \gamma C \tag{6.54}$$

$$\theta I_t (1+\alpha) - \sum_{j=1}^{n} q_{jt} X_j^1 + u_t \geq 0 \quad t = 1, \ldots, T \tag{6.55}$$

$$(X_j^1 - X_j^0) q_{jT} + s_j \geq 0 \quad j = 1, \ldots, n \tag{6.56}$$

$$\sum_{j=1}^{n} z_j \leq k$$

$$X_j^1, s_j \geq 0 \quad j = 1, \ldots, n$$

$$u_t \geq 0 \quad t = 1, \ldots, T$$

$$z_j \in \{0, 1\} \quad j = 1, \ldots, n$$

$$w_j \in \{0, 1\} \quad j = 1, \ldots, n.$$

Firstly, from equation (6.40) we obtain the following expression for variables d_t

$$d_t = \theta I_t (1+\alpha) - \sum_{j=1}^{n} q_{jt} X_j^1 + u_t \quad t = 1, \ldots, T,$$

that, then, is substituted in constraint (6.41) leading to costraint (6.52). We point out that this substitution permits the removal of T variables d_t and T constraints (6.40), but implies the introduction of T constraints (6.55) ensuring the non-negativity of the expression substituting variables d_t.
Subsequently, we utilize equation (6.42) to obtain the following expression for variables b_j

$$b_j = (X_j^1 - X_j^0) q_{jT} + s_j \quad j = 1, \ldots, n,$$

that is substituted, firstly, in constraints (6.43) obtaining constraints (6.53), and, secondly, in constraints (6.44) leading to the following expression

$$G_j = (X_j^1 - X_j^0)c_j^b q_{jT} + (c_j^b + c_j^s)s_j + f_j w_j \quad j = 1, \ldots, n$$

that is, subsequently, substituted in constraint (6.45) obtaining constraint (6.54).
The latter substitutions allow the removal of $2n$ variables b_j and G_j, other than $2n$ constraints (6.42) and (6.44). However, the first of the two substitutions requires the introduction of set of constraints (6.56).

6.3 A Heuristic Framework for MILP problems with binary variables

Recently, Angelelli *et al.* [3] proposed a new heuristic framework, referred to by the authors as Kernel Search, solving any MILP problem with binary variables. The algorithm is based on the idea of exhaustively exploring restricted portions of the feasible region of problems where the crucial decision is the selection of items from a given (typically of large size) set. Such decisions are usually modelled through the introduction of binary variables, e.g. variables z_j in model (6.23)-(6.34) that drive the selection of securities to be included in the portfolio, leading to MILP formulations.
In few words, the method is based on the detection of restricted sets of promising items, called kernels, and on the subsequent optimal solution of a sequence of restricted MILP problems that consider a set each. At the beginning of the algorithm, the initial kernel is built exploiting information provided by the solution of the linear programming relaxation of the original problem. All the remaining items, i.e. those not included in the initial kernel, are separated into groups, called buckets, according to a specific criterion. Subsequently, a sequence of small/moderate size MILP problems are solved and new promising items are possibly added to the kernel. The first MILP problem solved in the sequence is restricted to the initial kernel and provides a first upper (lower) bound for the problem. Any other MILP problem in the sequence is restricted to the previous kernel plus the items belonging to a bucket. Indeed, the solution of the current MILP problem may identify some new items not yet included in the kernel but that are worth considering. If this is the case, such new items are added *permanently* to the kernel. Additionally, the solution of any MILP problem in the sequence provides an upper (lower) bound for all the successive ones. The possibly updated kernel is then forwarded to the next MILP problem of the sequence. We point out that in [3] the kernel cardinality increases in a monotonic way, i.e. no item is discharged at any time. The procedure stops when the last bucket has been analyzed. The authors called the former framework the Basic Kernel Search. In [3] an Iterative Kernel Search is also proposed where the procedure to extend the initial kernel is restarted from the first bucket after the last one in the sequence has been analyzed. Such variant stops when no new items are added to the kernel.
The proposed framework allows implementing heuristics that have two major features from a practical point of view. Firstly, they require only little implementation effort because the most demanding part of the search is carried out by a (any) general-purpose LP and MILP solver. Secondly, the same heuristic is applicable to a large set of problems.
Drawing on the formerly described heuristic framework, we designed an enhanced version of the

kernel search aiming at improving its efficiency and its effectiveness. In the following, we will describe more formally the heuristic framework we designed pointing out the differences with respect to what described in [3].

6.3.1 Enhanced Kernel Search: An Application to the Index Tracking Problem

In this chapter we consider the IT Reduced Model (6.23)-(6.34). However, the reader should be aware that the framework we will introduce can be easily generalized to any combinatorial optimization problem modelled as a MILP problem with binary variables.
We recall that in the IT Reduced Model we are given a set N of securities to be selected. Moreover, one binary variable z_j, $j \in N$, is associated to each security j belonging to set N. Specifically, the binary variables drive the selection of securities. Furthermore, we are also given a set of additional continuous variables X_j^1, $j \in N$, that indicate the number of stocks of the selected security. We will denote as MILP(N) the problem that takes into consideration the whole set N of securities. We called the framework *Enhanced Kernel Search*.
The basic idea of the enhanced kernel search is to select a set K of promising securities, hereafter referred to as *kernel*, and solve to optimality the restricted MILP problem that considers only the securities belonging to set K, i.e. problem MILP(K). The cardinality of set K is of crucial importance. As a matter of fact, it affects the trade-off between efficiency of the procedure and quality of the solution. Set K should be small enough to find the optimal solution of problem MILP(K) quickly (efficiency) and, at the same time, large enough to contain the promising securities that most likely are selected in the optimal solution of problem MILP(N) (solution quality). The *initial kernel* K is built upon information provided by the solution of the linear programming relaxation on the whole set N of securities, from now on called LP(N). Subsequently, the kernel is possibly extended or narrowed according to the solution of a sequence of MILP(K) problems.
The enhanced kernel search is composed of two main steps. In the first phase, hereafter called the *initialization phase*, the LP(N) problem is solved and all the securities are sorted according to a predefined criterion. The sorting criterion should reflect the likelihood of a security to be included in the optimal solution of MILP(N), as inferred from the solution of LP(N). Subsequently, kernel K is initialized by selecting the first C securities of the list, where C is a given parameter, whereas all the remaining securities are separated into other sets called *buckets*. Finally, a first upper bound is obtained solving the MILP(K) problem that considers the initial kernel composition. In the second step, from now on referred to as *execution phase*, the initial kernel composition is iteratively modified taking into consideration one bucket at a time and the information provided by the solution of a MILP problem. Specifically, at each iteration the securities belonging to the current bucket are temporarily added to kernel K providing set $K' \supset K$. Problem MILP(K') is solved, after the introduction of two additional constraints aiming at reducing the computational time required by the solver to find the optimal solution. If some securities from the current bucket are selected in the optimal solution of problem MILP(K'), then they are added to kernel K. On the contrary, in the case a security belonging to kernel K has not been selected in the optimal solution of the current MILP(K') and in b of the sub-problems solved since it has been included in the kernel, where b is a given parameter, then the security is removed from kernel K. Actually, this is one of the features of our framework that improves upon that introduced in [3]. Indeed, in order

to keep under control the trade-off between efficiency and quality of the solution, the cardinality of the kernel is let increasing when new promising securities are found, but also decreasing if a security previously included seems to be no more promising.
The procedure is then iterated on a new bucket. When the last bucket has been analyzed the algorithm stops. We call the previous variant of the enhanced kernel search the *Basic Enhanced Kernel Search*.
We also propose an *Improved Enhanced Kernel Search* where information about the optimal solutions of the MILP sub-problems solved during the execution of the basic variant is collected and then analyzed. Specifically, at the end of the basic procedure an empty set, say $Z = \emptyset$, is created. Then, for each security j the ratio $\frac{\#select_j}{\#consider_j}$ is computed, where $\#select_j$ counts the number of sub-problems where security j is selected in the optimal (or feasible) solution, whereas $\#consider_j$ counts the number of sub-problems *solved*, not necessarily to optimality, considering security j as belonging to set K'. Specifically, both parameters consider the number of sub-problems the solver found either a feasible or an optimal solution, i.e. in the counting only the infeasible or unbounded sub-problems are excluded. Trivially, if the sub-problem is infeasible or unbounded we are not able to retrieve any useful information about the desirability of a security. On the other hand, considering also sub-problems where the solver is able to find a feasible but not proved optimal solution becomes a necessary choice when solving large-scale instances. When dealing with this sort of instances, in fact, it is usually opportune setting a threshold on the computational time allocated to solve the sub-problems. However, if the time limit is too tight, the solver would usually terminate its execution providing a feasible but not proved optimal solution. In our opinion, also this solution can be useful to identify the promising securities. Once the ratio $\frac{\#select_j}{\#consider_j}$ for each security has been computed, all the securities whose ratio is at least equal than g, where $g \in [0, 1]$ is a given parameter, are inserted into set Z.
In the initialization phase of the improved enhanced kernel search a linear relaxation problem LP(N, Z) is, subsequently, solved. Particularly, if a security j belongs to set Z, its inclusion in the optimal solution of problem LP(N, Z) is forced by setting the corresponding binary variable z_j equal to one. In case the cardinality of set Z is larger than k, the right-hand side in cardinality constraint (6.30), the securities are sorted according to a predefined criterion and the binary variable z_j is set equal to one for each of the first k securities in the list. Once the LP(N, Z) problem is solved a new initial kernel and new buckets are created upon the solution found. Subsequently, an execution phase, equal to that designed for the basic version, is run on the new kernel and the new buckets.
The rationale for the improved variant is that if a security is frequently selected in the optimal solution of the sub-problems MILP(K), it is most likely selected also in the optimal solution of the MILP(N).
In Algorithm 1 a pseudo-code for the basic enhanced kernel search is provided, whereas a sketch for the improved enhanced kernel search can be found in Algorithm 2.

In the basic enhanced kernel search the initialization phase is devoted the construction of the initial kernel K and of the sequence of buckets $\{B_i\}_{i=1,\dots,NB}$. In Step 1 the linear programming relaxation of the original problem is solved considering the whole set N of available securities. In case no feasible solution of problem LP(N) is found, then the algorithm terminates stating that no feasible solution for problem MILP(N) exists. Different sorting criteria, that lead to different implementations of the general framework, can be used in Step 2 to create list L. The basic idea is

Algorithm 1 Procedure: Basic Enhanced Kernel Search.

Input: A set N of securities.

Output: A feasible solution ($\mathbf{x}^*$, $\mathbf{z}^*$) and the corresponding objective function value w^* or *failure*=TRUE.

/* *Initialization phase*: Build the initial kernel K and the sequence of buckets $\{B_i\}_{i=1,\dots,NB}$ */

1. Set *failure*:=FALSE.
2. Solve the linear programming relaxation LP(N) and store its solution ($\mathbf{x}_{LP}$, $\mathbf{z}_{LP}$) if any. Otherwise, set *failure*:=TRUE and STOP.
3. Sort the securities in set N according to a predefined criterion that exploits information from the solution ($\mathbf{x}_{LP}$, $\mathbf{z}_{LP}$) and add them to list L.
4. Consider list L and
   - build the initial kernel K by taking the first C securities of L;
   - consider the last $|N| - C$ securities in list L and build a sequence $\{B_i\}_{i=1,\dots,NB}$ of buckets according to a predefined criterion;
   - let NB be the number of buckets generated.
5. Solve problem MILP(K). Let ($\mathbf{x}^*$, $\mathbf{z}^*$) be its optimal solution and w^* the corresponding optimal value. Otherwise, if no feasible solution of MILP(K) exists, set *failure*:=TRUE.

/* *Execution phase*: Modify the kernel composition according to the solutions of a sequence of MILP problems */

1. **for** i=1 **to** NB **do**
2. Create set $K' := K \bigcup B_i$.
3. Solve MILP(K') adding the following two constraints
   (a) $\sum_{j \in B_i} z_j \geq 1$;
   (b) set w^* as upper bound to the objective function value.
4. **if** MILP(K') is feasible, ($\mathbf{x}'$, $\mathbf{z}'$) is its optimal solution and w' is its optimal value **then**
5. **if** *failure*=TRUE **then** set *failure*:=FALSE **end if**;
6. set $\mathbf{x}^* := \mathbf{x}'$, $\mathbf{z}^* := \mathbf{z}'$ and $w^* := w'$;
7. let K_i^+ be the set of securities belonging to bucket B_i selected in solution ($\mathbf{x}'$, $\mathbf{z}'$);
8. let K_i^- be the set of securities belonging to kernel K that have not been selected b times since they have been introduced in kernel K;
9. set $K := K \bigcup K_i^+ \backslash K_i^-$.
10. **end if**
11. **end for**
12. STOP.

that the solution ($\mathbf{x}_{LP}$, $\mathbf{z}_{LP}$) of the linear programming relaxation can provide useful information about the most promising securities that are hopefully included in the optimal solution of problem MILP(N). Drawing on this idea, the goal is to design a sorting criterion such that more promising securities come first. In Step 3, the initial kernel K and the sequence of buckets $\{B_i\}_{i=1,\dots,NB}$ are

created. Specifically, the initial kernel K is constructed by selecting the first C elements in list L. The remaining elements, i.e. the securities starting from position $C+1$ to $|N|$ where $|N|$ is the length of list L, are separated into groups. Different rules can be used to generate the sequence of buckets, differing from the number NB of buckets created and/or their cardinality (called bucket length and referred to as *lbuck*). Finally, in Step 4, a MILP problem considering only the securities in the initial kernel K is solved providing a first upper bound on the value of the optimal cost.

In the execution phase a sequence of MILP problems is solved. The solution of each problem

Algorithm 2 Procedure: Improved Enhanced Kernel Search.

Input: A feasible solution ($\mathbf{x}^*$, $\mathbf{z}^*$) and the corresponding objective function value w^*.
Output: A feasible solution ($\mathbf{x}^*$, $\mathbf{z}^*$) and the corresponding objective function value w^*.

/* *Initialization phase*: Build a new initial kernel K and the sequence of buckets $\{B_i\}_{i=1,\dots,NB}$ utilizing the information collected performing the basic enhanced kernel search */

1. Let $Z \subseteq N$ be the set of securities such that ratio $\frac{\#select_j}{\#consider_j}$ is greater or equal than g.
2. **if** $|Z| \geq k$ **then**
3. sort the securities in set Z according to a predefined criterion and add the first k securities to set Z';
4. set $K := Z'$ and $z_j = 1$ for each security j in set K, then execute steps 11 to 14;
5. STOP.
6. **end if**
7. Set $z_j = 1$ for each security j in set Z.
8. Solve the linear programming relaxation LP(N, Z) and store its solution ($\mathbf{x}_{LP}$, $\mathbf{z}_{LP}$) if any.
9. Sort the securities in set N according to a predefined criterion that exploits information from the solution ($\mathbf{x}_{LP}$, $\mathbf{z}_{LP}$) and add them to list L.
10. Consider list L and
    - build the initial kernel K by taking the first C securities of list L;
    - consider the last $|N| - C$ securities in list L and build a sequence $\{B_i\}_{i=1,\dots,NB}$ of buckets according to a predefined criterion;
    - let NB be the number of buckets generated.
11. Solve problem MILP(K) adding the following two constraints
    (a) $\sum_{j \in B_i} z_j \geq 1$;
    (b) set w^* as upper bound to the objective function value.
12. **if** MILP(K) is feasible, ($\mathbf{x}$, $\mathbf{z}$) is its optimal solution and w its optimal value **then**
13. set $\mathbf{x}^* := \mathbf{x}$, $\mathbf{z}^* := \mathbf{z}$ and $w^* := w$.
14. **end if**

/* *Execution phase*: Repeat steps 1 to 10 of the execution phase in Algorithm 1 */

potentially provides information that is utilized to modify the kernel composition. Specifically, at each iteration i, $i = 1, \dots, NB$, a new, temporary, kernel $K' := K \bigcup B_i$ is created in Step 2, by

taking into consideration bucket B_i. In Step 3, the corresponding MILP(K') problem is solved after constraints (a) and (b) have been added. The rationale for the insertion of the two additional constraints is saving computational time. Indeed, the solutions we are interested to find are only those improving upon the former upper bound w^* using at least one new security from the current bucket B_i. In case the MILP(K') problem is feasible, its optimal solution is better than the best known one. If that is the case, in Steps 4-9, the best known solution is updated and substituted with the new one. In more details, in Step 6 set K_i^+ containing the securities in the current bucket that have been selected in the optimal solution of MILP(K') is created. Possibly, in Step 7, set K_i^- is created that contains the securities that have not been selected in b of the MILP problems solved since the single security has been introduced in kernel K. Finally, kernel K is updated in Step 8.
The basic enhanced kernel search fails only when no feasible solution for any MILP sub-problem is found. We point out that this could mean that either no feasible solution for MILP(N) problem exists or that the enhanced kernel search has not been able to find any of them (the latter outcome being, however, quite unlikely).
In the improved enhanced kernel search the initialization phase is dedicated to the construction of a *new* initial kernel K, hopefully different from that created performing the basic enhanced kernel search. The new initial kernel K is created utilizing the information about the desirability of the securities collected during the execution of the basic procedure. More precisely, when the basic procedure terminates, set $Z \subseteq N$ is created in Step 1.
In the case the cardinality of set Z is greater or equal than k, then those securities are sorted according to a predefined criterion in Step 3. Subsequently, an empty set Z' is created, and the first k securities in the list are added to it. Set Z' constitutes the kernel K, Step 4, and MILP(K) problem is then solved forcing the inclusion in the solution of each security by setting the corresponding binary variable equal to 1. Once problem MILP(K) is solved, either to optimality or proving its infeasibility, the algorithm stops in Step 5.
In the opposite case, once set Z has been created, in Step 8 the linear programming relaxation of a modified version of the original problem is solved. Indeed, we consider the original formulation and force the inclusion in the solution of each of the *most promising* securities in Step 7, i.e. we set $z_j = 1$ for each security j that is in set Z. From this point on, the improved enhanced kernel search behaves similarly to the basic variant.

6.4 Experimental Analysis

This section is devoted to the computational experiments we carried out. The computational experiments have been conducted on a PC Intel Core 2 with 2.40 GHz processor and 3.12 Gb of RAM. The IT Reduced Model and all the heuristics have been implemented in Java by means of the Concert Technology 2.0, and solved with CPLEX 10.1.
The present section is organized as follows. In Section 6.4.1 we briefly describe the instances that have been used. In the subsequent section, we validate the proposed formulation for the index tracking problem providing an analysis of the out-of-sample behaviors of the optimal portfolios selected by the model. In Section 6.4.3 we describe different possible implementations of the Enhanced Kernel Search framework for the IT Reduced Model. Attention is paid to the heuristics we implemented. Finally, Section 6.4.4 provides the computational results of the implemented heuristics.

6.4.1 Testing Environment

The data set we used are derived from the benchmark instances for index tracking problems belonging to the OR-Library [1]. There are currently 8 benchmark instances. Each instance corresponds to the securities composing a specific market index. Specifically, the following market indices are considered: Hang Seng (Hong Kong), DAX100 (Germany), FTSE100 (United Kingdom), S&P100 (USA), Nikkei225 (Japan), S&P500 (USA), Russell2000 (USA) and Russell3000 (USA). The number of securities considered in each instance ranges from 31, i.e. composing the Hang Seng market index, to 2151, composing the Russell3000 index. For each security, 291 weekly prices are provided. We considered the first 104 weekly prices as in-sample observations, i.e. T=104, and the following 52 weeks as out-of-sample period to validate the model. The parameter k, the right-hand side in cardinality constraint (6.30), has been set accordingly to the number of securities composing each market index. Table 6.1 summarizes the characteristics of each instance.

Instance	Market Index	N	in-sample (weeks)	out-of-sample (weeks)	k
indtrack1	Hang Seng	31	104	52	10
indtrack2	DAX100	85	104	52	10
indtrack3	FTSE100	89	104	52	10
indtrack4	S&P100	98	104	52	10
indtrack5	Nikkei225	225	104	52	10
indtrack6	S&P500	457	104	52	40
indtrack7	Russell2000	1318	104	52	70
indtrack8	Russell3000	2151	104	52	90

Table 6.1: The eight instances.

In all the computational experiments, we assumed that the investor does not hold any current portfolio, i.e. $X_j^0 = 0$, $j = 1, \ldots, n$. We considered a budget available for the investment equal to 10^5. Furthermore, we used $\epsilon_j = 0.01$ and $\delta_j = 0.1$, $j = 1, \ldots, n$. We set $c_j^b = c_j^s = 0.01$ and $f_j = 12$, $j = 1, \ldots, n$. Finally, we used $\gamma = 0.01$. All the instances have been solved setting a threshold on the computational time equal to 3600 seconds.

6.4.2 Index Tracking Formulation: Model Validation

We now turn our attention to the validation of the IT Reduced Model for tracking a market index. Assuming the investor prespective, we are interested in studying the performances of the optimal portfolios selected by the IT Reduced Model under the realized market behavior after the date of portfolio selection, i.e. during the out-of-sample period.
To this aim, the model validation proceeds along the following general lines. We solve the IT Reduced Model using the 104 in-sample weekly prices and assuming week 104 as date of the portfolio constitution, i.e. $T = 104$. We then observe the out-of-sample behavior of the optimal portfolio in the following 52 weeks.
Within the given time limit, CPLEX has been able to find a proved optimal solution only for the four smallest instances. For the remaining four instances, after one hour CPLEX returned a feasible but not proved optimal solution. A summary of the computational results is reported in Table 6.2.

[1] The benchmark instances are publicly available at http://people.brunel.ac.uk/~mastjjb/jeb/orlib/indtrackinfo.html.

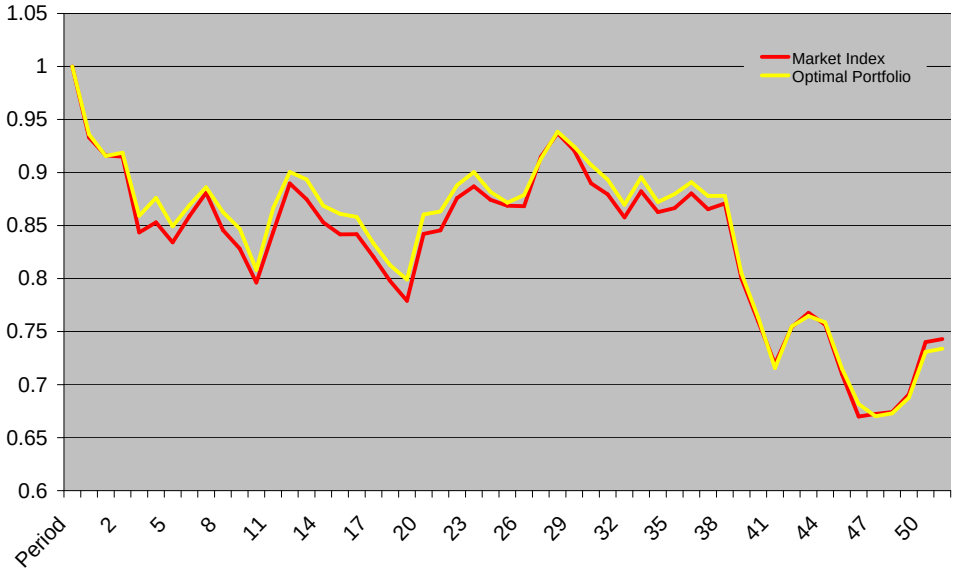

Figure 6.2: A comparison between the out-of-sample cumulative returns: Instance indtrack1.

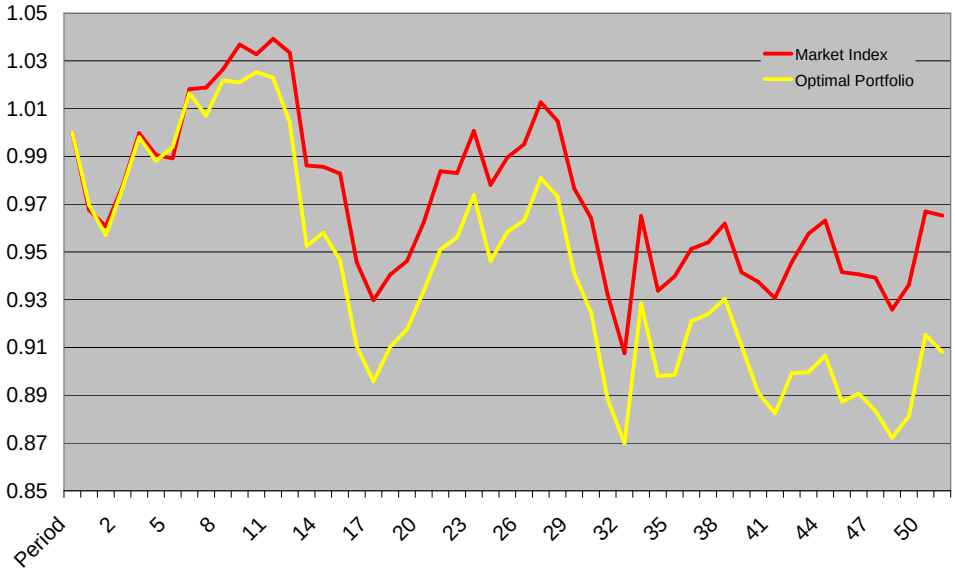

Figure 6.3: A comparison between the out-of-sample cumulative returns: Instance indtrack2.

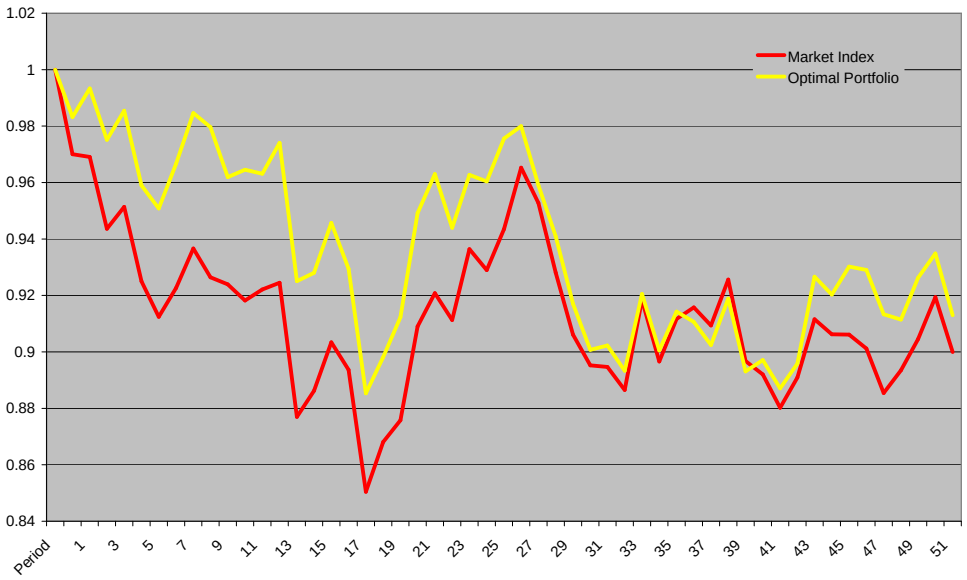

Figure 6.4: A comparison between the out-of-sample cumulative returns: Instance indtrack3.

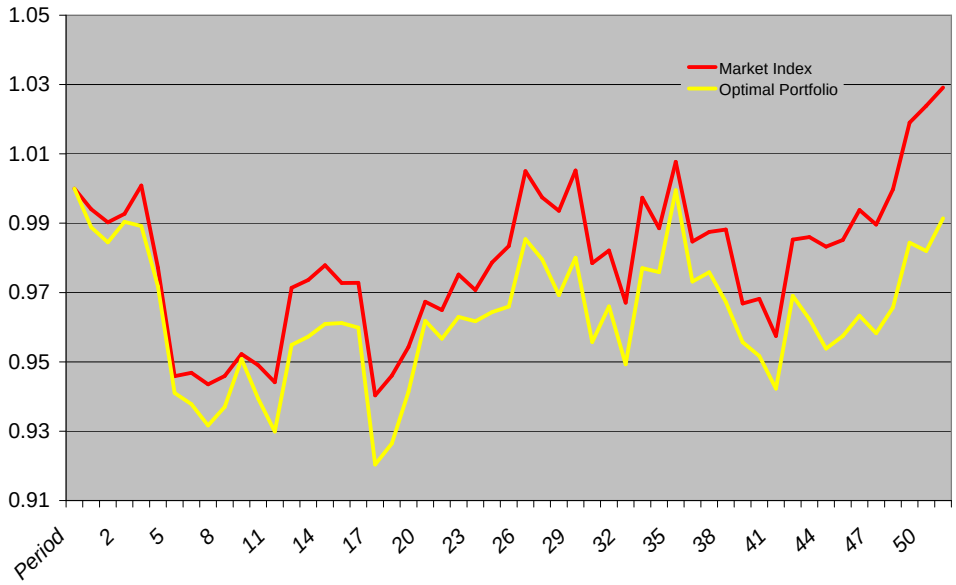

Figure 6.5: A comparison between the out-of-sample cumulative returns: Instance indtrack4.

Instance	Obj.	CPU (sec.)	CPLEX Status
indtrack1	363907.3803	1.547	Optimal
indtrack2	292558.2738	151.750	Optimal
indtrack3	383184.9924	301.375	Optimal
indtrack4	368210.1220	244.922	Optimal
indtrack5	414065.3268	3600.000	Feasible
indtrack6	102266.9404	3600.000	Feasible
indtrack7	168679.9121	3600.000	Feasible
indtrack8	129678.8511	3600.000	Feasible

Table 6.2: Solving the eight instances to optimality with CPLEX.

As we do not know the optimal solution of all the instances, in Figures 6.2-6.5 we report the ex-post cumulative returns for the first four instances only. Specifically, in each picture we compare the cumulative returns yielded by the market index with those reached by the optimal portfolios. It is worth pointing out that in any of the tested instances the optimal portfolio mimics quite closely the market index behavior for most part of the out-of-sample period. This happens especially for instance indtrack1 (see Figure 6.2) where the trends of the market index and of the optimal portfolio are hugely similar over the entire ex-post period. Furthermore, for instances indtrack2 and indtrack4 (see Figure 6.3 and 6.5, respectively) the optimal portfolios succeeded tracking the market indices in most of the out-of-sample period, and experienced an increasing tracking error only in the very last weeks. This behavior should suggest considering to rebalance the weights of the securities in order to re-optimize the portfolios.

6.4.3 Enhanced Kernel Search: The Heuristics Implemented

The Enhanced Kernel Search framework leads to different algorithms according to the criterion used to sort the securities, the number of securities that compose the initial kernel (parameter C), the type of buckets to be created (e.g., with fixed or variable length), the number of securities composing each bucket (parameter *lbuck*), the number of buckets to be generated (parameter *NB*), the number of iterations a security can be part of the kernel without being selected in the solutions of the sub-problems before being removed (parameter b). Furthermore, the Basic or Improved version has to be selected. Finally, if the Improved version is selected, the minimum value for ratio $\frac{\#select_j}{\#consider_j}$ that security j has to fulfill in order to be included in set Z has to be set (parameter g).
In this section we introduce the options we have implemented and tested for each parametric feature of the framework. It is worth pointing out that most of the options we have considered are independent from the specific optimization model, and can be applied (directly or easily adapted) to a large class of optimization problems. On the other hand, we made only few problem-dependent choices, specifically the criterion to sort the securities, and in these cases other options that do not rely on the problem specific characteristics have also been proposed.

Sorting criterion. We denote as $\hat{c}_j$ the reduced cost of variable X_j^1 in the optimal solution of the linear programming relaxation LP(N). We have considered the following criterion

- non-increasing order of the value of each security, i.e. $q_{jT}X_j^1$, and for those securities not selected in the optimal portfolio, non-decreasing order of the values $\hat{c}_j \frac{\epsilon_j C}{q_{jT}}$.

The rationale for this sorting criterion is to exploit information coming from the solution of the linear programming relaxation LP(N). Indeed, the criterion sort the securities, firstly, according to their value in the optimal solution, i.e. the securities with larger values are in the first positions of the list. Once all the securities with positive value in the portfolio have been sorted, the remaining securities are sorted utilizing the definition of reduced cost and a specific feature of the problem. Indeed, the reduced cost $\hat{c}_j$ represents the sensitivity of the objective function value (i.e., the increase in the problem under analysis) to the introduction of security j in the solution. We have chosen the reduced cost of variable X_j^1, and not for example that of variable z_j or w_j, because it is the variable that affects the objective function most. Furthermore, any security j can be introduced in the solution only at a minimum value, i.e. $X_j^1 \geq \frac{\epsilon_j C}{q_{jT}}$ given by constraint (6.25). Therefore, a non-basis variable X_j^1 increases the objective function value of at least $\hat{c}_j \frac{\epsilon_j C}{q_{jT}}$ if selected. Pointedly, we consider more interesting those securities whose introduction imply the lower increase of the objective function value.
In the execution of preliminary computational experiments, we implemented and tested also two problem-independent sorting criteria. The first considers non-increasing order of the value of each security, and for those securities not selected in the optimal portfolio, non-decreasing order of the values $\hat{c}_j$. The second considers non-increasing order of the value of each security, and a random order for those securities not selected in the optimal portfolio, i.e. it considers as promising only the securities selected in the optimal solution of LP(N) and all the remaining securities have the same importance. After extensive preliminary experiments and considering also the poor results reported in [3], we decided to not further consider these two criteria.

Kernel initialization. In all the tested heuristics we set the parameter C equal to the number of securities with positive value in the solution of problem LP(N). This means that all the securities that are selected in the solution of LP(N) compose the initial kernel. Obviously, different other choices can be alternatively made.

Number and conformation of the buckets. Several modes of constructing the sequence of buckets have been identified and tested in [3]. Drawing on their computational experience, we have chosen, between those, the most promising rules to build the buckets. Specifically, we implemented the rules to build *fixed* or variable length buckets. After extensive preliminary computational experiments, we decided to not further consider heuristics that build buckets with variable length.
Therefore, we consider only heuristics based on the generation of disjoint buckets all with the same length. Two are the parameters that can be set a priori, the bucket length $lbuck$ and/or the number of buckets to generate NB. Particularly, one can set a priori both parameters $lbuck$ and NB. In this case, if $NB < \left\lceil \frac{|N|-C}{lbuck} \right\rceil$, then we set $NB := \left\lceil \frac{|N|-C}{lbuck} \right\rceil$. Moreover, one can also set a priori parameter $lbuck$, then parameter NB will come out as a result of the bucket construction, i.e. $NB := \left\lceil \frac{|N|-C}{lbuck} \right\rceil$. The last option is to set a priori parameter NB, then parameter $lbuck$ is determined on the basis of the bucket construction, i.e. $lbuck := \left\lceil \frac{|N|-C}{NB} \right\rceil$. Whatever option is chosen, all the generated buckets will have a cardinality equal to $lbuck$, but possibly the last one may contain a number of securities less than $lbuck$. Furthermore, we tested several values for parameters $lbuck$ and NB. To the sake of brevity, we report only those values that better allow a comparison between the heuristics tested in [3] and the new heuristics we propose.
Parameter b has to be set accordingly to parameter NB. Particularly, parameter b drives the trade-

off between efficiency of the procedure, i.e. $b << NB$ in order to eliminate rapidly securities not selected in the sub-problems, and quality of the solution, i.e. $b \approx NB$ in order to consider the interactions among most of the securities.

Basic and Improved variants. For each heuristic characterized by a number of buckets greater than or equal to one, we have implemented both the Basic and the Improved variants. For the Improved Enhanced Kernel Search, we set $g := 0.75$. Furthermore, if the number of securities in set Z is larger than k, then those securities are sorted considering the non-increasing order of the average value of each security, i.e. $\frac{\sum_{i=0}^{NB} q_{jT} X_j^{1(i)}}{\#consider_j}$, $j = 1, \ldots, n$, where $X_j^{1(i)}$ is the value taken by variable X_j^1 in the solution of sub-problem number i, $i = 0, \ldots, NB$.

Time limit. After some preliminary tests, and considering the difficulties experienced when solving problem MILP(N) (see Section 6.4.2), we realized that also the sub-problems are often quite difficult to be solved to optimality. For this reason, we decided to implement all the heuristics imposing a computational time threshold on the solution of the sub-problems. Analogously to what we have done in 6.4.2, we have set to 3600 seconds the computational time allowed for each instance. This means that, at the start of the algorithm for each sub-problem a time slot equal to $time = 3600/n_p$ is allocated, where n_p is the number of sub-problems still to be solved (in the beginning equal to NB+1). If the solver does not terminate within the allowed time, then the computation is stopped and the best solution found so far, if any, is used to update the kernel and the upper bound of the problem. On the other hand, if the solver terminates solving a sub-problem before the time allocated is elapsed, the time left is re-allocated among the remaining sub-problems.

Heuristic	Variant	Bucket Length	$lbuck$	NB	b
Fixed-Bucket(0, 0, 1)	Basic	Fixed	0	0	1
Fixed-Bucket(C, 2, 3)	Basic	Fixed	C	2	3
Fixed-Bucket($\left\lceil \frac{\lvert N\rvert - C}{12} \right\rceil$, 12, 2)	Basic	Fixed	$\left\lceil \frac{\lvert N\rvert - C}{12} \right\rceil$	12	2
I-Fixed-Bucket(C, 2, 3)	Improved	Fixed	C	2+2	3
I-Fixed-Bucket($\left\lceil \frac{\lvert N\rvert - C}{10} \right\rceil$, 10, 3)	Improved	Fixed	$\left\lceil \frac{\lvert N\rvert - C}{10} \right\rceil$	10+10	3

Table 6.3: The tested heuristics: A detail of the chosen parameters.

Basic Enhanced Kernel Search heuristics	
Fixed-Bucket(0, 0, 1)	One sub-problem is solved considering the securities with positive value in the solution of LP(N).
Fixed-Bucket(C, 2, 3)	Three sub-problems are solved. Two buckets are created with length equal to C. No security is discarded.
Fixed-Bucket($\left\lceil \frac{\lvert N\rvert - C}{12} \right\rceil$, 12, 2)	13 sub-problems are solved. 12 buckets are created with length equal to $\left\lceil \frac{\lvert N\rvert - C}{12} \right\rceil$. A security is discarded after that it has not been selected in 2 sub-problems since it has been introduced in the kernel.

Improved Enhanced Kernel Search heuristics	
I-Fixed-Bucket(C, 2, 3)	Improved version of Fixed-Bucket(C, 2, 3).
I-Fixed-Bucket($\left\lceil \frac{\lvert N\rvert - C}{10} \right\rceil$, 10, 3)	Improved version of Fixed-Bucket($\left\lceil \frac{\lvert N\rvert - C}{10} \right\rceil$, 10, 3). 22 sub-problems are possibly solved (11 for the basic and 11 for the improved). 20 buckets are possibly created with length equal to $\left\lceil \frac{\lvert N\rvert - C}{10} \right\rceil$. A security is discarded after that it has not been selected in 3 sub-problems since it has been introduced in the kernel.

Table 6.4: The tested heuristics: A brief explanation.

A list of the tested heuristics along with the specifics of the parameters is provided in Table 6.3. Basically, each basic enhanced kernel search implementation is referred to as Fixed-Bucket($lbuck$, NB, b). The keyword Fixed-Bucket identifies the conformation of the buckets, i.e. with fixed length. The parameters listed between parenthesis defines the length of the buckets, the number of buckets generated and the number of iterations a security can be part of the kernel without being selected in the solutions of the sub-problems before being removed. The improved versions we report are referred to as *I*-Fixed-Bucket($lbuck$, NB, b).
A brief explanation of the heuristics is reported in Table 6.4.

6.4.4 Enhanced Kernel Search: Computational Experiments

In this section we provide and comment the foremost results of the computational experiments we carried out. Particularly, in Tables 6.5 and 6.6 we compare the performances of the selected heuristics with those provided by CPLEX. Additionally, in Table 6.7 we report, for each single instance, the heuristic that found the best-known (or optimal) solution.
Specifically, in Table 6.5 we report, for each instance, the computational time required by CPLEX to find the optimal solution. We recall that CPLEX has not been able to prove the optimality of the best solution found for the four largest size instances, i.e. instances indtrack5-8. Thus, in those cases we report the threshold on the computational time that we have set. In Tables 6.5-6.7, for each instance we report in column Gap % the difference in percentage between the best solution found by the heuristic and the corresponding best solution found by CPLEX, i.e. Gap= $\frac{w^*_{heur}-w^*_{CPLEX}}{w^*_{CPLEX}}$, where w^*_{heur} is the objective function value of the best solution found by the heuristic whereas w^*_{CPLEX} is the corresponding value for the best solution found by CPLEX. The computational time required to execute the algorithm is also provided.

	CPLEX	Fixed-Bucket(0, 0, 1)		Fixed-Bucket(C, 2, 3)		I-Fixed-Bucket(C, 2, 3)	
Instances	CPU (sec.)	Gap %	CPU (sec.)	Gap %	CPU (sec.)	Gap %	CPU (sec.)
indtrack1	1.547	0.00%	1.219	0.00%	2.344	0.00%	2.531
indtrack2	151.750	0.00%	24.359	0.00%	71.078	0.00%	71.875
indtrack3	301.375	4.50%	36.250	0.00%	239.643	0.00%	256.823
indtrack4	244.922	5.08%	106.796	0.00%	229.391	0.00%	243.984
indtrack5	3600.000	-0.87%	3601.156	0.70%	3600.313	0.70%	1802.921
indtrack6	3600.000	15.35%	3601.453	2.92%	3601.734	2.34%	3600.125
indtrack7	3600.000	35.54%	3602.745	36.45%	3600.588	22.45%	3600.573
indtrack8	3600.000	26.48%	3602.297	26.48%	3600.312	15.80%	3600.266

Table 6.5: The tested heuristics: Evaluating the new features.

As we expected, limiting the analysis to the only securities selected in the linear programming relaxation seems to be a sensible choice solely on the small size instances. In fact, for the first four instances the heuristic Fixed-Bucket(0, 0, 1) finds either the optimal solution (see instances indtrack1 and indtrack2), or a sub-optimal solution whose error is indeed relatively small (see instances indtrack3 and indtrack4). Additionally, the heuristic found the best solution considerably faster than CPLEX. On the contrary, the error reported solving the last three instances is very big. This can be explained by the fact that many securities are selected in the solution of the linear programming relaxation when solving the smallest instances. As a consequence, for each of these instances the heuristic Fixed-Bucket(0, 0, 1) solves (quickly) only one sub-problem that considers

	Fixed-Bucket($\left\lceil \frac{\lvert N\rvert - C}{12} \right\rceil$,12, 2)		I-Fixed-Bucket($\left\lceil \frac{\lvert N\rvert - C}{10} \right\rceil$, 10, 3)	
Instances	Gap %	CPU (sec.)	Gap %	CPU (sec.)
indtrack5	-7.23%	1666.469	-0.42%	2243.125
indtrack6	10.67%	1739.297	4.09%	3600.047
indtrack7	-22.85%	1803.047	-21.08%	3600.090
indtrack8	-11.88%	3600.182	-8.98%	3600.094

Table 6.6: The best average (Fixed-Bucket($\left\lceil \frac{\lvert N\rvert - C}{12} \right\rceil$,12, 2)) and the best worst error (I-Fixed-Bucket($\left\lceil \frac{\lvert N\rvert - C}{10} \right\rceil$, 10, 3)) heuristics.

Instances	Heuristics	Gap %	CPU (sec.)
indtrack1	Fixed-Bucket(0, 0, 1)	0.00%	1.219
indtrack2	Fixed-Bucket(0, 0, 1)	0.00%	24.359
indtrack3	Fixed-Bucket(C, 1, 2)	0.00%	225.773
indtrack4	Fixed-Bucket(C, 2, 3)	0.00%	229.391
indtrack5	I-Fixed-Bucket($\left\lceil \frac{\lvert N\rvert - C}{12} \right\rceil$,12, 3)	-7.23%	1112.203
indtrack6	Fixed-Bucket($\left\lceil \frac{\lvert N\rvert - C}{7} \right\rceil$, 7, 4)	-2.16%	3600.485
indtrack7	I-Fixed-Bucket($\left\lceil \frac{\lvert N\rvert - C}{12} \right\rceil$,12, 1)	-25.84%	1804.109
indtrack8	I-Fixed-Bucket($\left\lceil \frac{\lvert N\rvert - C}{5} \right\rceil$,5, 1)	-17.39%	3600.859

Table 6.7: The best heuristic for each single instance.

most of the much promising securities, i.e. the solution of the linear programming relaxation provides very useful information about the securities included in the optimal solution. However, as the number of securities composing the index increases, those of them selected in the optimal solution of the linear programming relaxation turn out to be only a few with respect to the universe of securities available. Thus, in these cases, heuristic Fixed-Bucket(0, 0, 1) does not consider many securities, and generating buckets becomes a valuable strategy to improve the results.

In fact, the computational results provided by the former heuristic are improved by heuristic Fixed-Bucket(C, 2, 3). As a matter of fact, heuristic Fixed-Bucket(C, 2, 3) found the optimal solution for each of the four smallest instances. The corresponding computational times are longer than those for Fixed-Bucket(0, 0, 1) because the former heuristic solves two sub-problems more than the latter. On the other side, for the remaining instances heuristic Fixed-Bucket(C, 2, 3) has improved significantly the solutions found in one instance (see instance indtrack6), whereas the error is very similar to that reported by heuristic Fixed-Bucket(0, 0, 1) on the other instances.

We reported also the results obtained testing heuristic I-Fixed-Bucket(C, 2, 3) in order to show the effectiveness of the improved version. Indeed, the improved version dominates the basic version solving all the tested instances. The improvement is quite large for the two largest size instances (see instances indtrack7 and indtrack8). As a matter of fact, in these cases the improved version improves the solution found by the basic implementation by more than 10%.

The former results show how, on the one hand, considering buckets can improve the solutions found considering only the securities selected in the linear programming relaxation. On the other hand, they show how the improved version of a heuristic can improve the solutions found by the basic version.

In Table 6.6 we show how, choosing a good heuristic, it is possible improving the solutions found by CPLEX. Specifically, in this table we report the results of two heuristics for the four largest

size instances. Both heuristics improve CPLEX in three out of four instances. Specifically, Fixed-Bucket($\left\lceil \frac{|N|-C}{12} \right\rceil$,12, 2) is the heuristic with the best average Gap value computed out of the four instances, whereas I-Fixed-Bucket($\left\lceil \frac{|N|-C}{10} \right\rceil$, 10, 3) is the heuristic that, among those that improve CPLEX three times out of four, reported the smallest worst error. The good results reported by heuristic Fixed-Bucket($\left\lceil \frac{|N|-C}{12} \right\rceil$,12, 2) are related to an effective choice of the number of buckets created. In fact, heuristic Fixed-Bucket($\left\lceil \frac{|N|-C}{12} \right\rceil$,12, 2) solves 13 sub-problems altogether, i.e. it separates all the securities not included in the initial kernel in 12 buckets. This permits, along with allowing the removal from the kernel of the securities, to maintain a small size of the sub-problems, especially when solving the largest size instances. On the one side, a smaller number of buckets would imply generating larger size sub-problems that would be hardly tackled by the solver within the computational time allocated. On the other side, a larger number of buckets seems to prevent evaluating relations among securities, thus providing worse results.
Analogous reasons motivate the effectiveness of heuristic I-Fixed-Bucket($\left\lceil \frac{|N|-C}{10} \right\rceil$, 10, 3).
To the sake of completeness, in Table 6.7 we report the best heuristic we found for each of the eight instances. The best heuristic has been computed as the one that found the solution with the smallest objective function value. If more than one heuristic found the same solution, the quickest heuristic is selected as the best one. It is worth pointing out that for each instance we found at least one heuristic that performs better than CPLEX, either in terms of computational time (for the small size instances) or in terms of quality of the solution (for the large size instances). Additionally, we highlight that on the smallest size instances the best heuristics are those that solve few sub-problems. As a matter of fact, creating few large buckets allow to consider profitable relations among securities within the bucket. The fact that all the former heuristics are basic variants can be explained considering that any improved variant solves at least one sub-problem more than the corresponding basic version. On the contrary, the improved versions provide better results solving the four largest size instances. For these instances the best heuristics are those that create several buckets. Most likely, this is due to the threshold set on the computational time. In fact, the bigger the buckets, the more difficult for CPLEX becomes the solution of the sub-problems, i.e. CPLEX often did not find the optimal or even a good feasible solution within the given time limit. Therefore, in all these cases it can be effective creating several buckets. On the other side, generating too many buckets could prevent considering relationships among securities belonging to different buckets.

6.5 Conclusions

In this chapter we have, firstly, studied the problems of replicating and out-performing the performances of a market index. Secondly, we have introduced a heuristic framework to solve the index tracking problem that can be easily generalized to solve any MILP model that include binary variables. The first goal was to find a meaningful linear formulation for both problems of index tracking and enhanced indexation. To this aim, we have proposed a MILP formulation for each problem. Furthermore, we have tested the index tracking formulation solving benchmark instances. Then, we have provided evidences of its effectiveness comparing the out-of-sample behaviors of the optimized portfolios with those of the corresponding market indices. As the com-

putational burden needed solving to optimality the index tracking model becomes excessive when benchmark indices composed of many securities are considered, the second goal became to design an effective heuristic to solve the problem. To this aim, we have introduced the Enhanced Kernel Search framework. Then, we have tested several implementations of the proposed heuristic framework. We have confirmed its effectiveness showing that it is possible finding heuristics that perform better than CPLEX in all the tested instances.

7
Conclusions

In order for single-period portfolio optimization to have a strong impact in theory and practice of portfolio selection, we feel that the following issues are important

a) studying the influence of techniques that generate the input data feeding the optimization model;

b) examining the impact of techniques that address the problem of uncertainty in the input parameters;

c) investigating the effectiveness of investment strategies different from those classically assumed;

d) designing heuristic solution methods when the mathematical formulation is hard to solve with standard methods.

In the development of this thesis we tried to study each of the former issues. Specifically,

1) we have studied the problem of portfolio optimization when scenarios are generated by means of several techniques. The reference model include transaction costs and assume the Conditional Value-at-Risk as performance measure. It has been tested on real-life data from the London Stock Exchange. Under these assumptions we found that one technique outperforms the others under different market conditions;

2) we have investigated the impact of robust optimization techniques on a portfolio selection problem. We modified the Conditional Value-at-Risk optimization model according to two well-known robust optimization approaches. We then compared the optimal robust portfolios with those selected by the nominal model. As a result, we found that even if robust optimization techniques are, without any doubt, theoretically valuable, identifying their merits and weaknesses when applied to real problems still needs further investigation;

3) we have considered several investment strategies in addition to the commonly assumed buy-and-hold one. We have modified the Conditional Value-at-Risk model with transaction costs in order to take into consideration re-optimizing a current portfolio composition. We have compared the different investment strategies carrying out computational experiments on real-life data from the German Stock Exchange Market. The results show that rebalancing conveniently the portfolio composition can improve the ex-post performances of the optimal portfolios;

4) we have studied the problems of index tracking and enhanced indexation. We have proposed a MILP formulation for each problem. We have validated the index tracking problem solving benchmark instances. Then, we have introduced a heuristic framework to solve the problem. We have shown the effectiveness of the proposed framework implementing several heuristics and comparing their performances with CPLEX.

It is worth noting that most of the mathematical models we have introduced consider real-life features, such as, among others, transaction costs and limits on the weights of single assets in portfolio. Furthermore, we have carried out most of the computational experiments aiming at showing the effectiveness of the mathematical models under different market conditions. Finally, the computational results we have reported provide strong evidences of the effectiveness of the proposed models as supporting tools for financial decision-making.

8

Appendix: Introduction to Linear Programming

In this chapter we provide a brief introduction to *linear programming* problem, i.e. the problem of minimizing (or maximizing) a linear cost function subject to linear equality and inequality constraints. The purpose of this chapter is mainly introductory. We refer to [18] for any further detail.
An *unconstrained optimization* problem can be cast in the following form

$$\min_{\mathbf{x}\in\mathcal{R}^n} f(\mathbf{x}) \quad (\text{or } \max_{\mathbf{x}\in\mathcal{R}^n} f(\mathbf{x})).$$

Generally speaking, the idea of unconstrained optimization is to find the point where the objective function $f(\mathbf{x})$ takes the lowest (highest) value, without any constraint on the values the *decision variables*, i.e. vector $\mathbf{x}$, can take.
On the other hand, a *constrained optimization* problem can be cast in the following form

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \quad (\text{or } \max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})) \tag{8.1}$$

where $\mathcal{X} \subset \mathcal{R}^n$ is a set containing all and only the possible values the decision variables can take. $\mathcal{X}$ is called the *feasible set* (or feasible region) of the optimization problem. Set $\mathcal{X}$ is defined by a set of equalities and/or inequalities referred to as *constraints*. If the objective function $f : \mathcal{X} \subset \mathcal{R}^n \to \mathcal{R}$ and all the constraints are linear functions with respect to variables $\mathbf{x}$, problem (8.1) is said to be *linear*.
Several management practical problems can be formulated as linear programming problems. As a matter of fact, linear programming is the most commonly applied form for constrained optimization problems. Most extensively it is used in the domains of business and economic, but is often utilized for engineering problems as well. Some industrial sectors that use linear programming models include transportation, energy, telecommunications, finance engineering, and manufacturing. Finally, it has proved to be useful in modeling several types of problems in planning, routing, scheduling, assignment, and design.
Structure of the Appendix. In Section 8.1 we introduce the basic concepts of linear programming along with a short overview of the main solution methods. In Section 8.2 we introduce the key concepts of computational complexity. Finally, in Section 8.3 we extend the analysis to integer linear programming problems, and mention the foremost algorithms solving this class of problems.

8.1 Linear Programming Problems

In a *general* Linear Programming (LP) problem, we are given a cost vector $\mathbf{c} \in \mathcal{R}^n$ and we aim at minimizing a linear cost function $\mathbf{c}'\mathbf{x}$ over all the n-dimensional vectors $\mathbf{x} \in \mathcal{R}^n$, subject to a set of linear equality and/or inequality constraints. Specifically, let $i = 1, \ldots, m$ be the index corresponding to a specific constraint. For every constraint i we are given an n-dimensional vector $\mathbf{a}_i$ and a scalar b_i, that will be used to form the i-th constraint. Additionally, the variables are often constrained to take non-negative values. Therefore, an LP problem can be formulated as follows

$$\begin{aligned} &\min \quad \mathbf{c}'\mathbf{x} \\ &\text{subject to } \mathbf{a}_i'\mathbf{x} \geq b_i \quad i = 1, \ldots, m \\ &x_j \geq 0 \quad j = 1, \ldots, n. \end{aligned} \tag{8.2}$$

The variables $x_1, \ldots, x_n$ are called *decision variables*. A vector $\mathbf{x} \in \mathcal{R}^n$ satisfying all the constraints is said to be a *feasible solution* for problem (8.2). The function $\mathbf{c}'\mathbf{x}$ is called the *objective function* or cost function. A feasible solution $\mathbf{x}^*$ that minimizes the objective function, i.e. $\mathbf{c}'\mathbf{x}^* \leq \mathbf{c}'\mathbf{x}$ for all feasible solutions $\mathbf{x}$, is said to be an optimal feasible solution for problem (8.2), or shortly an *optimal solution*. The value $\mathbf{c}'\mathbf{x}^*$ is then called the *optimal value*. On the other hand, if for any real number K we can find a feasible solution $\mathbf{x}$ whose value is less than K, we say that the cost is unbounded below, or concisely that the problem is *unbounded*.
Finally, we point out that there is no need to study maximization problems separately, since maximizing a cost function $\mathbf{c}'\mathbf{x}$ is equivalent to minimizing function $(-\mathbf{c})'\mathbf{x}$. Furthermore, solving a problem with an equality constraint $\mathbf{a}_i'\mathbf{x} = b_i$ is equivalent to solving a problem with the two inequality constraints $\mathbf{a}_i'\mathbf{x} \geq b_i$ and $\mathbf{a}_i'\mathbf{x} \leq b_i$, certainly doubling the number of constraints. In addition, any constraint of the form $\mathbf{a}_i'\mathbf{x} \leq b_i$ can be rewritten as $(-\mathbf{a}_i)'\mathbf{x} \geq -b_i$. Finally, the non-negativity constraints $x_j \geq 0$, $j = 1, \ldots, n$, can be interpreted as special cases of constraint of the form $\mathbf{a}_i'\mathbf{x} \geq b_i$, where $\mathbf{a}_i$ is equal to the unit vector and $b_i = 0$. As a consequence, we conclude that the feasible set in a general linear programming problem can be expressed always in terms of inequality constraints of the form $\mathbf{a}_i'\mathbf{x} \geq b_i$.
Let $\mathbf{b}' = [b_1, \ldots, b_m]$, and let $\mathbf{A} \in \mathcal{M}(m \times n)$, where $\mathcal{M}(m \times n)$ denotes the set of all m-by-n matrices, be the matrix whose rows are the row vectors $\mathbf{a}_1', \ldots, \mathbf{a}_m'$. Therefore, the constraints $\mathbf{a}_i'\mathbf{x} \geq b_i$, $i = 1, \ldots, m$, can be expressed compactly in the form $\mathbf{A}\mathbf{x} \geq \mathbf{b}$, and the linear programming problem (8.2) can be written as

$$\begin{aligned} &\min \quad \mathbf{c}'\mathbf{x} \\ &\text{subject to } \mathbf{A}\mathbf{x} \geq \mathbf{b}. \end{aligned}$$

A set $\mathcal{H}$ that can be described in the form $\{\mathbf{x} \in \mathcal{R}^n | \mathbf{A}\mathbf{x} \geq \mathbf{b}\}$ is called a *polyhedron*. In particular, a set of the form $\{\mathbf{x} \in \mathcal{R}^n | \mathbf{A}\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ is also a polyhedron and is usually referred to as a polyhedron in standard form.
Inequality constraints can be transformed in equivalent equality constraints by means of the introduction of dummy variables. For instance, given an inequality constraint of the form $\mathbf{a}_i'\mathbf{x} \leq b_i$, we introduce a new variable s_i and transform the constraint as follows

$$\mathbf{a}_i'\mathbf{x} + s_i = b_i$$

$$s_i \geq 0.$$

Variable s_i is called a *slack variable*. Conversely, an inequality constraint of the form $\mathbf{a}_i'\mathbf{x} \geq b_i$ can be transformed in an equality constraint by the introduction of a *surplus* variable s_i. Then, the constraint can be cast as $\mathbf{a}_i'\mathbf{x} - s_i = b_i$, $s_i \geq 0$.
A linear programming problem of the form

$$\begin{aligned} \min \quad & \mathbf{c}'\mathbf{x} \\ \text{subject to } & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{8.3}$$

is said to be expressed in *standard form*. Because of the former reasons, every linear programming problem can be brought into standard form. Specifically, when describing and using algorithms to solve LP problems, the reference form is the standard formulation (8.3), which is computationally more convenient.
We now introduce the important concept of *extreme point* of a polyhedron. Let $\mathcal{H}$ be a polyhedron. A vector $\mathbf{x} \in \mathcal{H}$ is an extreme point (or *corner*) of $\mathcal{H}$ if we cannot find two vectors $\mathbf{y}, \mathbf{z} \in \mathcal{H}$, such that $\mathbf{y} \neq \mathbf{x}$ and $\mathbf{z} \neq \mathbf{x}$, and a scalar $\lambda \in [0, 1]$, such that $\mathbf{x} = \lambda\mathbf{y} + (1 - \lambda)\mathbf{z}$. Moreover, let us assume that $\mathbf{y}$ and $\mathbf{z}$ are corners of polyhedron $\mathcal{H}$. Let $L = \{\lambda\mathbf{y} + (1 - \lambda)\mathbf{z} | 0 \leq \lambda \leq 1\}$ be the segment that joins points $\mathbf{y}$ and $\mathbf{z}$. Then, the line segment L is said to be an *edge* of polyhedron $\mathcal{H}$.
So far, we have assumed that the feasible set is bounded (i.e. it does not extent to infinity), and that the problem has an unique optimal solution. Actually, as we mentioned already, this is not always the case. As a matter of fact, we can have one of the following possibilities

1. there exists an unique optimal solution;
2. the feasible set is empty, i.e. there is not any feasible solution;
3. there exist multiple optimal solutions;
4. the optimal cost is $-\infty$, i.e. no feasible solution is optimal: This happens when, for any feasible solution, we can always find another feasible solution with lower value.

Solution Methods

Fortunately, a general feature of an LP problem is that if the problem has at least one optimal solution, then an optimal solution can be found among the corners of the feasible set. This fact has been the trigger that led Dantzig (e.g., see [45]) to the development of the simplex method.
The simplex algorithm solves LP problems by constructing a feasible initial solution at a corner of the feasible region and then walking along edges of the polyhedron to vertices with successively lower values of the objective function until the optimum is reached. Although this algorithm is quite efficient in practice and can be guaranteed to find the global optimum if certain precautions against cycling are taken, it has poor worst-case behavior: It is possible to construct a linear programming problem for which the simplex method takes a number of steps exponential in the problem size. In fact, for a long time it was not known whether the linear programming problem was solvable in polynomial time.

This long standing issue was resolved by Khachiyan in [92] with the introduction of the ellipsoid method, the first worst-case polynomial time algorithm for linear programming. The ellipsoid method has been of seminal importance for establishing the polynomial time solvability of linear programs. However, see [18], the ellipsoid method does not lead to a practical algorithm for solving linear programming problems. Rather, it demonstrates that linear programming is efficiently solvable from a theoretical point of view. Most importantly, the ellipsoid method inspired new lines of research in linear programming, and in particular the development of a new class of algorithms, known as interior point methods, that are both practically and theoretically efficient.
The field of interior point methods has its origins in the work of Karmarkar [89], who introduces the first interior point algorithm with polynomial time complexity. Karmarkar algorithm not only improves on Khachiyan theoretical worst-case polynomial bound, but also promises dramatic practical performance improvements over the simplex method. In few words, and in contrast to the simplex algorithm which finds the optimal solution by progressing along extreme points of the polyhedron, interior point methods move through the interior of the feasible region.
Though the interior point algorithm has been proven to converge in a number of steps that is polynomial in the problem size, whereas the simplex method has been shown to have a poor worst-case behavior, the simplex algorithm can be quite efficient on average even with respect to an interior point algorithm.

8.2 Introduction to Computational Complexity

Before introducing the concept of computational complexity, it is useful to draw a distinction between a *problem* and an *instance* of the problem. For instance, linear programming is a problem, whereas

$$\min 2x_1 + 3x_2$$

$$\text{subject to } x_1 + x_2 \leq 1$$

$$x_1, x_2 \geq 0$$

is an instance of the linear programming problem. More generally (we refer to [18] for the formers and all the following definitions and concepts), an instance of an optimization problem consists of a feasible set $\mathcal{X}$ and an objective function $f : \mathcal{X} \rightarrow \mathcal{R}$. On the other hand, an optimization problem can be defined as a collection of instances. Instances of a problem need to be described according to a common format. For example, instances of the linear programming problem expressed in standard form can be described by listing the entries of matrix $\mathbf{A}$, and vectors $\mathbf{c}$ and $\mathbf{b}$.
Technically, the *size* of an instance is the number of bits required to encode it. However, it is usually measured in terms of the inherent dimensions of the instance (such as the number of nodes and edges in a graph), plus the number of bits required to encode the numerical information in the instance (such as the edge costs) in order to take into consideration that, for example, adding or multiplying large integers or high-precision floating point numbers is more demanding than adding or multiplying single-digit integers. Since numerical data are encoded in binary, an integer C requires about $log_2|C|$ bits to encode and so contributes logarithmically to the size of the instance.
An *algorithm* for a problem is a finite set of instructions of the type used in common programming

language (arithmetic operations, conditional statements,...) that solve any instance of that optimization problem. In the simplest of circumstances, one can assume that each instruction (including arithmetic operations) takes unit time: The former is usually referred to as the arithmetic model. Assuming this model to count time, a *step* consists of one of the following operations: addition, subtraction, multiplication, finite-precision division, and comparison of two numbers. Thus if an algorithm requires 100 additions and 220 comparisons for some instance, we say that the algorithm requires 320 steps on that instance. In order to make this number meaningful, one would like to express it as a function of the size of the corresponding instance, but determining the exact function would be impractical. Instead, since one is mainly concerned with how long the algorithm takes asymptotically as the size of an instance becomes large, one can formulate a simple function of the input size that is a reasonably tight upper bound on the actual number of steps. Such a function is called the *complexity* or *running time* of the algorithm.

For the reasons we explained before, the running time of the algorithm is usually expressed as a function of the input data, rather than the precise input size. For instance, considering the TSP, an algorithm running time might be expressed as a function of the number of nodes, the number of edges, and the maximum number of bits required to encode any edge cost.

In general, and particularly in analyzing the inherent tractability of a problem, we are interested in an asymptotic analysis. In other words, we are interested in how the running time grows as the size of the instance becomes very large. For these reasons, it is useful to introduce the following concept. Let f and g be two functions that map positive numbers to positive numbers. We write $f(n) = O(g(n))$ if there exist positive numbers n_0 and c such that $f(n) \leq cg(n)$ for all $n \geq n_0$. The function $cg(n)$ is thus an asymptotic upper bound of function $f(n)$. Additionally, we write $f(n) = \Omega(g(n))$ if there exist positive numbers n_0 and c such that $f(n) \geq cg(n)$ for all $n \geq n_0$.

Usually instead of trying to estimate the running time for each possible choice of the input, it is customary to estimate the running time for the *worst possible input data* of a given size. For example, given an algorithm for linear programming, we might be interested in estimating its worst-case running time over all problems with a given number of variables and constraints. This emphasis on the worst-case is somewhat conservative and, in practice, the average running time of an algorithm might be more relevant. However, the average running time is much more difficult to estimate, or even to define, and for this reason, the worst-case approach is widely used.

Let $T(n)$ be the worst-case running time of some algorithm over all instances of size n. An algorithm runs in *polynomial time* if there exists an integer k such that $T(n) = O(n^k)$.

On the other hand, algorithms whose running time is $\Omega(2^{cn})$, where n is a parameter representing the problem size and c is a constant, are said to take at least *exponential time*. For such algorithms and if $c = 1$, each time that computer hardware becomes faster by a factor of 2, we can increase the value of n that can be handle only by 1. It is then reasonable to expect that no matter how much technology improves, problems with truly large values of n will always be difficult to handle by means of those algorithms.

Therefore, an algorithm is considered efficient if its running time grows polynomially with the size of the input; otherwise, it is usually considered inefficient. Clearly, this view of algorithmic complexity has its problems. Since in the counting of the running time one has to consider the worst-case running time, it only takes one instance to pronounce an algorithm inefficient, while the algorithm might be very fast in the vast majority of instances. Thus, this view of worst-case efficiency, although insightful, might not always reflect reality. For instance, the simplex method solves routinely large-scale problems, even though it takes exponential time in the worst-case.

We mentioned in the previous section that polynomial time algorithms are available for linear programming problems. To the contrary, no polynomial time algorithm for integer programming problems has been discovered, despite many efforts over the course of several decades have been made. The same is true for many other discrete optimization problems, the traveling salesman problem being a prominent example. As more and more problems of this type emerged, it was realized that many such seemingly intractable problems were closely related. The root cause for their intractability was then sought in complexity theory. To a great extent, complexity theory focuses on the relation between different problems and looks at classes of problems that have comparable computational requirements (referred to as *computational classes*) in terms or resources, e.g. time and memory, needed to solve the problems. The prime example of an important complexity class is the class $\mathcal{P}$, defined as the set of all problems that can be solved by a polynomial time algorithm.

To introduce the computational class $\mathcal{NP}$ we use the definition proposed by Bertsimas and Tsitsiklis [18]. The starting point is the fact that ZOIP (see the following section for a definition) seems to be a difficult problem, and the existence of a polynomial time algorithm that can solve it is considered unlikely. Hence any problem that is "at least as hard" as ZOIP is also unlikely to be polynomial time solvable. Then, a problem is said to be $\mathcal{NP}$*-hard* if ZOIP can be transformed to it in polynomial time. A polynomial time algorithm for an $\mathcal{NP}$-hard problem would lead to a polynomial time algorithm for ZOIP, which is considered unlikely. For this reason, $\mathcal{NP}$-hardness is viewed as strong evidence that a problem is not polynomially solvable.

The following definition refers to problems that are "no harder" than ZOIP. A problem is said to belong to $\mathcal{NP}$ if it can be transformed to ZOIP in polynomial time. It can be shown that the class $\mathcal{P}$ of polynomially solvable problems is contained in $\mathcal{NP}$. Despite that it is not known whether the inclusion is proper, and this is probably the most important open problem in the theory of computation. If it turns out that $\mathcal{P} = \mathcal{NP}$, then ZOIP and all other problems in $\mathcal{NP}$ can be solved in polynomial time. Conversely, if ZOIP belongs to $\mathcal{P}$, then every problem in $\mathcal{NP}$ also belongs to $\mathcal{P}$, since it can be transformed to ZOIP. We therefore have $\mathcal{P} = \mathcal{NP}$ if and only if ZOIP can be solved in polynomial time.

The last definition refers to problems that are "exactly as hard" as ZOIP. A problem Π is said to be $\mathcal{NP}$*-complete* if it belongs to $\mathcal{NP}$ and is also $\mathcal{NP}$-hard; that is, if Π can be transformed to ZOIP and ZOIP can be transformed to Π in polynomial time. In some sense, $\mathcal{NP}$-complete problems can be viewed as "the hardest problems in $\mathcal{NP}$".

Concepts related to $\mathcal{P}$ and $\mathcal{NP}$ have been discussed by Cobham [36] and Edmonds [55]. In particular, Edmonds made the conjecture that the traveling salesman problem does not belong to $\mathcal{P}$. Cook [42] and Levin [102] first established that certain problems are $\mathcal{NP}$-complete. Karp [90] showed that several important discrete optimization problems are $\mathcal{NP}$-complete. For a comprehensive reference on computational complexity, see the books by Garey and Johnson [68] and by Papadimitriou [129].

8.3 Integer Linear Programming Problems

In discrete optimization problems we aim at finding a solution $\mathbf{x}^*$ in a discrete set $\mathcal{D}$ that optimizes an objective function $f(\mathbf{x})$ defined for all $\mathbf{x} \in \mathcal{D}$. Discrete optimization problems arise in a great

variety of contexts in science, business, economy and engineering. A natural and systematic way to study the class of discrete optimization problems is to express them as integer programming problems. We refer to [18] for the following definitions and any further detail not covered in the present section.
Generally speaking, a mixed integer linear programming problem is similar to a linear programming problem except the further requirement that some of the variables must take integer values. In what follows, given matrices $\mathbf{A}$, $\mathbf{B}$, and vectors $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$, we will refer to the following formulation

$$\begin{aligned} \min \quad & \mathbf{c}'\mathbf{x} + \mathbf{d}'\mathbf{y} \\ \text{subject to } & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b} \\ & \mathbf{x}, \mathbf{y} \geq \mathbf{0} \\ & \mathbf{y} \quad \text{integer.} \end{aligned} \tag{8.4}$$

Problem (8.4) is said to be a *mixed integer linear programming* (MILP) problem. If there are no continuous variables $\mathbf{x}$, the problem is said to be a *pure integer linear programming* (ILP) problem. Furthermore, if there are no continuous variables and the components of vector $\mathbf{y}$ are restricted to be either 0 or 1, the problem is called *zero-one (or binary) integer linear programming* (ZOILP or BILP) problem.
We highlight that even if there are inequality constraints, we can still express the problem in the reference form (8.4) by adding slack or surplus variables (in the same way we described for linear programming).
Integer linear programming is a rather powerful modeling framework that provides great flexibility for expressing discrete optimization problems. On the other hand, the price for this flexibility is that integer linear programming "seems" to be a much more difficult problem than linear programming.
Moreover, comparing the mathematical formulations, in linear programming a good formulation is one that has a small number n of variables and m of constraints because the computational complexity of the problem grows polynomially in n and m (see [18], page 461). Additionally, given the availability of several efficient algorithms for linear programming, the choice of a formulation does not critically affect the ability to solve the problem. Conversely, the situation in integer linear programming is drastically different. Extensive computational experience suggests that the choice of a formulation is crucial.
Let us introduce the key concept of linear programming *relaxation*.

Definition 1 *Given the mixed integer linear programming problem (8.4), the following problem where the requirement that* $\mathbf{y}$ *is a vector of integers is dropped (***relaxed***)*

$$\begin{aligned} \textit{min} \quad & \mathbf{c}'\mathbf{x} + \mathbf{d}'\mathbf{y} \\ \textit{subject to } & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b} \\ & \mathbf{x}, \mathbf{y} \geq \mathbf{0} \end{aligned} \tag{8.5}$$

is said to be its **linear programming relaxation**. *In case the requirement in model (8.4) is that the decision variables* y_j, $j = 1, \ldots, n$, *take either value 0 or value 1, i.e.* $y_j \in \{0, 1\}$, $j = 1, \ldots, n$,

then, in the linear programming relaxation, y_j takes values between 0 and 1, i.e. $y_j \in [0,1]$, $j = 1, \ldots, n$.

We point out that if an optimal solution of the linear programming relaxation is feasible for the original MILP problem, it is also an optimal solution for the latter.
The concept of linear programming relaxation has revealed fundamental in the development of several solution algorithms.

Solution Methods

Unlike linear programming problems, mixed integer linear programming problems are usually very difficult to solve. Many researchers proposed solution methods to solve an ILP problem, and many of them proposed problem-specific algorithms. The methods known in the literature can be classified into three main classes (see [18])

1. **exact algorithms** that are guaranteed to find an optimal solution, but may take an exponential number of iterations;

2. **approximation algorithms** that provide in polynomial time a sub-optimal solution together with a bound on the degree of sub-optimality;

3. **heuristic algorithms** that provide a sub-optimal solution, but without a guarantee on its quality. Although the running time is not guaranteed to be polynomial, empirical evidence suggests that some of these algorithms find a good solution quickly.

It is worth mentioning, among the exact algorithms, the methods belonging to the class of *cutting planes*. The basic idea in cutting plane methods is to solve the mixed integer programming problem (8.4) by solving a sequence of linear programming problems. Firstly, the linear relaxation (8.5) is solved to optimality. If its solution $\mathbf{x}^*_{LP}$ is integer, then it is an optimal solution for the original problem. On the contrary, if $\mathbf{x}^*_{LP}$ is not integer, a linear inequality constraint is added to problem (8.5) such that all integer solutions of (8.4) satisfy, but $\mathbf{x}^*_{LP}$ does not. Then, the previous two steps are iterated until an integer solution is found. A classical manner of building the linear inequality constraint has been suggested by Gomory in [75]. A difficulty with general purpose cutting plane algorithms is that the added inequalities cut only a very small portion of the feasible set of the linear programming relaxation. As a result, the practical performance of such algorithms has not been impressive. For this reason, cutting plane algorithms with deeper cuts utilizing the particular structure of the problem at hand have been designed.
Another category of exact algorithms are the *branch-and-bound* methods. Instead of exploring the entire feasible set, they use bounds on the optimal value to avoid exploring certain portions of the set of feasible integer solutions. The branch-and-bound methods are based on the solution of a sequence of linear integer programming sub-problems. The basic idea is to compute a lower bound to the optimal value of a sub-problem. If, for instance, the optimal cost of a sub-problem is difficult to compute exactly, a lower bound might be a lot easier to obtain. A popular choice is to use as a bound the optimal cost of the linear programming relaxation. Moreover, in the course of the algorithm, we will occasionally solve certain sub-problems to optimality. This allows us to

maintain an upper bound U on the optimal cost, which could be the cost of the best solution encountered so far. The essence of the method is that if the lower bound of a particular sub-problem is greater than the upper bound U, then this sub-problem does not need to be considered further, since the optimal solution of the sub-problem is not better than the best feasible integer solution encountered so far. For a comprehensive description of the branch-and-bound method, see, among all, [124].

A variant of the method, called *branch-and-cut*, represents a combination of cutting plane with branch-and-bound methods. It utilizes cuts when solving the sub-problems. Specifically, the formulation of the sub-problems is augmented with additional cuts, in order to improve the bounds obtained from the linear relaxations. A general branch-and-cut algorithm for mixed ZOILP is proposed in Balas, Ceria and Cornuéjols [6].

Approximation algorithms provide, in polynomial time, a feasible, but not guaranteed optimal, solution together with a provable guarantee regarding its degree of sub-optimality. These algorithms are usually very specific for the problem analyzed.

Local search is a general approach to design heuristics for an optimization problem of the form (8.4). The basic concept is to start from an initial feasible solution $\mathbf{x}'$ of problem (8.4), set that solution as the incumbent and evaluate the objective function in that point. Subsequently, the algorithm evaluates the objective function for some other feasible points $\mathbf{w}$ that belong to a "neighborhood" of the incumbent solution. If the cost of a neighbor solution $\mathbf{w}'$ is better than the cost of the incumbent solution $\mathbf{x}'$, solution $\mathbf{w}'$ becomes the new incumbent and the algorithm is iterated. If no such neighbor solution is found, the algorithm stops: A *local optimum* solution has been found, i.e. a feasible solution which is at least as good as all the solutions belonging to its neighborhood.

The specifics of a local search algorithm are strictly related to what it means for two feasible solutions to be neighbors, and this represents an arbitrary choice made during the design of the algorithm. For example, in the domain of linear programming problems, we can say that two vertices of the feasible set are neighbors if they are connected by an edge. Under this definition of neighborhood, the simplex method can be interpreted as a local search method. However, while the simplex algorithm finds the globally optimal solution, local search methods only guarantee a local optimal solution, in general.

A generic trade-off that arises in local search methods is that when larger neighborhoods are considered, there are fewer local minima and a better solution is likely to be obtained when the algorithm terminates. On the other hand, the larger the neighborhood, the more feasible solutions need to be examined at each iteration and this makes the algorithm slower.

We mentioned already that the main drawback of local search algorithms is that they only guarantee to find a local optimum. In recent years, several researchers proposed metaheuristics that improve upon local search methods by allowing occasional moves to feasible solutions with higher (worse) costs. To this end, it is worth mentioning the classes of Genetic Algorithms (e.g., see [73]), Simulated Annealing (e.g., used in discrete optimization problems by Kirkpatrick, Gelatt, and Vecchi [93]), Tabu Search (a description of the method can be found in Glover and Laguna [72]), and Ant Colony Algorithms (see [53] for an excellent description of the algorithm).

References

[1] C. Acerbi and D. Tasche. On the coherence of expected shortfall. *Journal of Banking & Finance*, 26:1491–1507, 2002.

[2] F. Andersson, D. Rosen, and S. Uryasev. Credit risk optimization with Conditional Value-at-Risk criterion. *Mathematical Programming*, 89:273–291, 2001.

[3] E. Angelelli, R. Mansini, and M.G. Speranza. Kernel search: A heuristic framework for MILP problems with binary variables. Technical Report 2007-04-56, Department of Electronics for Automation, University of Brescia, Italy, 2007.

[4] C. Archetti, M. Savelsberg, and M.G. Speranza. An optimization-based heuristic for the split delivery vehicle routing problem. *Transportation Science*, 42:22–31, 2008.

[5] P. Artzner, F. Delbaen, J.M. Eber, and D. Heath. Coherent measures of risk. *Mathematical Finance*, 9:203–228, 1999.

[6] E. Balas, S. Ceria, and G. Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42:1229–1246, 1996.

[7] E. Balas and E. Zemel. An algorithm for large zero-one knapsack problems. *Operations Research*, 28:1130–1154, 1980.

[8] J.E. Beasley, N. Meade, and T.-H. Chang. An evolutionary heuristic for the index tracking problem. *European Journal of Operational Research*, 148:621–643, 2003.

[9] A. Ben-Tal, T. Margalit, and A. Nemirovski. *Robust modeling of multi-stage portfolio problems*, chapter in High Performance Optimization, pages 303–328. H. Frenk, K. Roos, T. Terlsky and S. Zhang eds., Kluwer Academic Publisher, 2000.

[10] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23:769–805, 1998.

[11] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25:1–13, 1999.

[12] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88:411–424, 2000.

[13] D.P. Bertsekas. *Dynamic Programming and Optimal Control (Vol. I and II, Two-Volume Set)*. Athena Scientific, 3rd Edition, 2007.

[14] D. Bertsimas, D. Pachamanova, and M. Sim. Robust linear optimization under general norms. *Operations Research Letters*, 32:510–516, 2004.

[15] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52:35–53, 2004.

[16] D. Bertsimas and A. Thiele. A data-driven approach to inventory theory. *Working Paper*, 2004.

[17] D. Bertsimas and A. Thiele. *Robust and data-driven optimization: Modern decision-making under uncertainty*, chapter 4 in Tutorials on Operations Research. INFORMS, 2006.

[18] D. Bertsimas and J.N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[19] D. Bienstock. Histogram models for robust portfolio optimization. *Journal of Computational Finance*, 11(1), 2007.

[20] F. Black and R. Litterman. Global portfolio optimization. *Financial Analysts Journal*, 48:28–43, 1992.

[21] M.E. Blume, J. Crockett, and I. Friend. Stock ownership in the United States: Characteristics and trends. *Survey of Current Business*, 54:16–40, 1974.

[22] T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Economics*, 31:307–327, 1986.

[23] T. Bollerslev. Modelling the coherence in short-run nominal exchange rates: A multivariate generalized ARCH model. *Review of Economics and Statistics*, 72:498–505, 1990.

[24] T. Bollerslev, R.Y. Chou, and K.F. Kroner. ARCH modelling in finance: A review of the theory and empirical evidence. *Journal of Econometrics*, 52:5–59, 1992.

[25] T. Bollerslev, R. Engle, and J.M. Wooldridge. A capital asset pricing model with time varying covariances. *Journal of Political Economy*, 96:116–131, 1988.

[26] S.P. Bradley and D.B. Crane. A dynamic model for bond portfolio management. *Management Science*, 19:139–151, 1972.

[27] P. Bühlmann. Bootstrap for time series. *Statistical Science*, 17:52–72, 2002.

[28] G.C. Calafiore. Ambiguous risk measures and optimal robust portfolios. *SIAM Journal on Optimization*, 18:853–877, 2007.

[29] N.A. Canakgoz and J.E. Beasley. Mixed-integer programming approaches for index tracking and enhanced indexation. *European Journal of Operational Research*, 196:384–399, 2009.

[30] D.R. Carino, T. Kent, D.H. Myers, C. Stacy, M. Sylvanus, A.L. Turner, K. Watanabe, and W.T. Ziemba. The Russel-Yasuda Kasai model: An asset/liability model for a japanese insurance company using multistage stochastic programming. *Interfaces*, 24:29–49, 1994.

[31] D.R. Carino, D.H. Myers, and W.T. Ziemba. Concepts, technical issues and uses of the Russel-Yasuda Kasai financial planning model. *Operations Research*, 46:450–463, 1998.

[32] S. Ceria and R.A. Stubbs. Incorporating estimation errors into portfolio selection: Robust portfolio construction. *Journal of Asset Management*, 7:109–127, 2006.

[33] T.-J. Chang, N. Meade, J.E. Beasley, and Y.M. Sharaiha. Heuristic for cardinality constrained portfolio optimisation. *Computers & Operations Research*, 27:1271–1302, 2000.

[34] L. Chiodi, R. Mansini, and M.G. Speranza. Semi-absolute deviation rule for mutual funds portfolio selection. *Annals of Operations Research*, 124:245–265, 2003.

[35] Y. Chong Ho. Resampling methods: Concepts, applications, and justification. *Practical Assessment, Research & Evaluation*, 8, 2003.

[36] A. Cobham. *Logic, Methodology and Philosophy of Science*, chapter in The intrinsic computational difficulty of functions, pages 24–30. Y. Bar-Hillel eds., Elsevier/North-Holland, 1965.

[37] G. Connor and H. Leland. Cash management for index tracking. *Financial Analysts Journal*, 51:75–80, 1995.

[38] A. Consiglio, F. Cocco, and S.A. Zenios. The value of integrative risk management for insurance products with guarantees. *Journal of Risk Finance*, 2:6–11, 2001.

[39] A. Consiglio, F. Cocco, and S.A. Zenios. Asset and liability modelling for participating policies with guarantees. *European Journal of Operational Research*, 186:380–404, 2008.

[40] A. Consiglio and S.A. Zenios. Designing portfolios of financial products via integrated simulation and optimization models. *Operations Research*, 47:195–208, 1999.

[41] A. Consiglio and S.A. Zenios. Integrated simulation and optimization models for tracking international fixed income indices. *Mathematical Programming*, 89:311–339, 2001.

[42] S.A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd ACM Symposium on the Theory of Computing*, pages 151–158, 1971.

[43] G. Cornuejols and R. Tütüncü. *Optimization Methods in Finance*. Cambridge University Press, 2007.

[44] O.L.V. Costa and A.C. Paiva. Robust portfolio selection using linear-matrix inequalities. *Journal of Economic Dynamics & Control*, 26:889–909, 2002.

[45] G.B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[46] R. De Franceschi, M. Fischetti, and Toth P. A new ILP-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, 105:471–499, 2006.

[47] R. Dembo, A. Aziz, D. Rosen, and M. Zerbs. *Mark To Future: A Framework for Measuring Risk and Reward*. Algorithmics Publications, 2000.

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

[49] U. Derigs and N.-H. Nickel. On a local-search heuristic for a class of tracking error minimization problems in portfolio management. *Annals of Operations Research*, 131:45–77, 2004.

[50] L. Di Gaspero, G. di Tollo, A. Roli, and A. Schaerf. Hybrid local search for constrained financial portfolio selection problems. In *Proceedings of the CPAIOR*, pages 44–58, 2007.

[51] G. di Tollo and A. Roli. Metaheuristics for the portfolio selection problem. *International Journal of Operations Research*, 5:13–35, 2008.

[52] Z. Ding and R. Engle. Large scale conditional covariance matrix modeling, estimation and testing. *Academia Economic Papers*, 29:157–184, 2001.

[53] M. Dorigo and T. Stützle. *Ant Colony Optimization*. The MIT Press, Massachusetts Institute of Technology, 2004.

[54] C. Dose and S. Cincotti. Clustering of financial time series with application to index and enhanced index tracking portfolio. *Physica A: Statistical Mechanics and its Applications*, 355:145–151, 2005.

[55] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[56] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1993.

[57] L. El-Ghaoui and H. Lebret. Robust solutions to least-square problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18:1035–1064, 1997.

[58] L. El-Ghaoui, H. Lebret, and F. Oustry. Robust solutions to uncertain semidefinte programs. *SIAM Journal on Optimization*, 9:33–52, 1998.

[59] L. El-Ghaoui, M. Oks, and F. Oustry. Worst-case Value-at-Risk and robust portfolio optimization: A conic programming approach. *Operations Research*, 51:543–556, 2003.

[60] D. Ellsberg. Risk, ambiguity, and the savage axioms. *The Quarterly Journal of Economics*, 75:643–669, 1961.

[61] E.J. Elton and M.J. Gruber. On the optimality of some multiperiod portfolio selection criteria. *Journal of Business*, 47:231–243, 1974.

[62] E.J. Elton, M.J. Gruber, S.J. Brown, and W.N. Goetzmann. *Modern Portfolio Theory and Investment Analysis - Sixth Edition*. John Wiley & Sons, 2002.

[63] E. Erdŏgan and G. Iyengar. Ambiguous chance constrained problems and robust optimization. *Mathematical Programming, Series B*, 107:37–61, 2006.

[64] F.J. Fabozzi, P.N. Kolm, D.A. Pachamanova, and S. Focardi. *Robust Portfolio Optimization and Management*. John Wiley and Sons, 2007.

[65] F.J. Fabozzi, P.N. Kolm, D.A. Pachamanova, and S. Focardi. Robust portfolio optimization: Recent trends and future directions. *Journal of Portfolio Management*, 33:40–48, 2007.

[66] J.E. Falk. Exact solutions to inexact linear programs. *Operations Research*, 24:783–787, 1976.

[67] Y. Fang and S.-Y. Wang. A fuzzy index tracking portfolio selection model. In *Computational Science ICCS 2005*, volume 3516 of *Lecture Notes in Computer Science*, pages 554–561. Springer Berlin / Heidelberg, 2005.

[68] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to The Theory of NP-Completeness*. Freeman, 1995.

[69] G. Gennotte and A. Jung. Investment strategies under transaction costs: The finite horizon case. *Management Science*, 40:385–404, 1994.

[70] M. Gilli and E. Këllezi. *The threshold accepting heuristic for index tracking*, volume Financial Engineering, E-commerce and Supply Chain, Applied Optimization Vol. 70, pages 1–18. P. Pardalos and V.K. Tsitsiringos eds., Kluwer Academic Publishers, 2002.

[71] P. Glasserman. *Montecarlo Methods in Financial Engineering*. Springer-Verlag, 2003.

[72] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[73] D.E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, 2002.

[74] D. Goldfarb and G. Iyengar. Robust portfolio selection problems. *Mathematics of Operations Research*, 28:1–38, 2003.

[75] R.E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958.

[76] R.C. Grinold and R.N. Kahn. *Active Portfolio Management, 2nd Edition*. McGraw-Hill Professional, 1999.

[77] G. Guastaroba, Mitra G., and M.G. Speranza. Investigating the effectiveness of robust portfolio optimization techniques. Technical Report 333, Department of Quantitative Methods, University of Brescia, Italy, 2009.

[78] G. Guastaroba, R. Mansini, and M.G. Speranza. On the use of CVaR model in a rebalancing portfolio strategy. Technical Report 249, Department of Quantitative Methods, University of Brescia, Italy, 2005.

[79] G. Guastaroba, R. Mansini, and M.G. Speranza. On the effectiveness of scenario generation techniques in single-period portfolio optimization. Technical Report 268, Department of Quantitative Methods, University of Brescia, Italy, 2006.

[80] G. Guastaroba, R. Mansini, and M.G. Speranza. Models and simulations for portfolio rebalancing. *Computational Economics*, 33:237–262, 2009.

[81] G. Guastaroba, R. Mansini, and M.G. Speranza. On the effectiveness of scenario generation techniques in single-period portfolio optimization. *European Journal of Operational Research*, 192:500–511, 2009.

[82] N. Gülpinar and B. Rustem. Worst-case optimal robust decisions for multi-period mean-variance portfolio optimization. *European Journal of Operational Research*, 183:981–1000, 2007.

[83] N. Gülpinar, B. Rustem, and R. Settergren. Simulation and optimization approaches to scenario tree generation. *Journal of Economic Dynamics & Control*, 28:1291–1315, 2004.

[84] P. Hall, J.L. Horowitz, and B. Jing. On blocking rules for the bootstrap with dependent data. *Biometrika*, 82:561–574, 1995.

[85] K. Høyland and S.W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47:295–307, 2001.

[86] N.J. Jobst, M.H. Horniman, C. Lucas, and G. Mitra. Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints. *Quantitative Finance*, 1:1–13, 2001.

[87] P.H. Jorion. *Value at Risk: The New Benchmark for Managing Financial Risk, 3rd Edition*. McGraw-Hill, 2006.

[88] L.-Y. Ju, X.-D. Ji, and S.-Y. Wang. Stochastic programming models in financial optimization: A survey. *Advanced Modeling and Optimization*, 5:1–26, 2003.

[89] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[90] R.M. Karp. *Complexity of Computer and Computations*, chapter in Reducibility among combinatorial problems, pages 85–103. R.E. Miller and J.W. Thacher eds., Plenum Press, 1972.

[91] H. Kellerer, R. Mansini, and M.G. Speranza. Selecting portfolios with fixed costs and minimum transaction lots. *Annals of Operations Research*, 99:287–304, 2000.

[92] L.G. Khachiyan. A polynomial algorithm in linear programming. *Akademiia Nauk SSSR, Doklady*, 224:1093–1096, 1979. (English Translation: Soviet Mathematics Doklady, 20, 191-194, 1979.).

[93] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[94] H. Konno and T. Hatagi. Index-plus-alpha tracking under concave transaction cost. *Journal of Industrial and Management Optimization*, 1:87–98, 2005.

[95] H. Konno and A. Wijayanayake. Minimal cost index tracking under nonlinear transaction costs and minimal transaction unit constraints. *International Journal of Theoretical and Applied Finance*, 4:939–958, 2001.

[96] H. Konno and A. Wijayanayake. Portfolio optimization problem under concave transaction costs and minimal transaction unit constraints. *Mathematical Programming*, 89:233–250, 2001.

[97] H. Konno and H. Yamazaki. Mean-absolute deviation portfolio optimization model and its application to Tokyo stock market. *Management Science*, 37:519–531, 1991.

[98] R. Kouwenberg. Scenario generation and stochastic programming models for asset liability management. *European Journal of Operational Research*, 134:279–292, 2001.

[99] R. Kouwenberg and S.A. Zenios. *Stochastic Programming Models for Asset Liability Management*, chapter 6 in Handbook of Asset and Liability Management: Vol.1, Theory and Methodology, Handbooks in Finance, pages 253–304. S.A. Zenios and W.T. Ziemba eds., Elsevier, 2006.

[100] W.H. Kruskal and J.M. Tanur. *International Encyclopedia of Statistics*. Old Tappan, 1978.

[101] M.J. Lawrence, R.H. Edmunson, and M.J. O'Connor. The accuracy of combining judgemental and statistical forecasts. *Management Science*, 32:1521–1532, 1986.

[102] L.A. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973. English translation of original in Problemy Peredachi Informatsii (in Russian).

[103] G. Levy. *Computational Finance: Numerical Methods for Pricing Financial Instruments*. Elservier, 2004.

[104] D. Li and W.-L. Ng. Optimal dynamic portfolio selection: Multiperiod mean-variance formulation. *Mathematical Finance*, 10:387–406, 2000.

[105] Z.-F. Li, Z.-X. Li, S.-Y. Wang, and X.-T. Deng. Optimal portfolio selection of assets with transaction costs and no short sales. *International Journal of Systems Science*, 32:599–607, 2001.

[106] Z.-F. Li, S.-Y. Wang, and X.-T. Deng. A linear programming algorithm for optimal portfolio selection with transaction costs. *International Journal of Systems Science*, 31:107–117, 2000.

[107] M.S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

[108] D.G. Luenberger. *Investment science*. Oxford University Press, 1998.

[109] S. Makridakis and R. Winkler. Average of forecasts: Some empirical results. *Management Science*, 29:987–996, 1983.

[110] B. Mandelbrot. The variation of certain speculative prices. *The Journal of Business of the University of Chicago*, 26:394–419, 1993.

[111] R. Mansini, W. Ogryczak, and M.G. Speranza. LP solvable models for portfolio optimization: A classification and computational comparison. *IMA Journal of Management Mathematics*, 14:187–220, 2003.

[112] R. Mansini, W. Ogryczak, and M.G. Speranza. Conditional Value at Risk and related linear programming models for portfolio optimization. *Annals of Operations Research*, 152:227–256, 2005.

[113] R. Mansini and M.G. Speranza. Heuristic algorithms for the portfolio selection problem with minimum transaction lots. *European Journal of Operational Research*, 114:219–233, 1999.

[114] R. Mansini and M.G. Speranza. An exact approach for portfolio selection with transaction costs and rounds. *IIE Transactions*, 37:919–929, 2005.

[115] D. Maringer and H. Kellerer. Optimization of cardinality constrained portfolios with a hybrid local search algorithm. *OR Spectrum*, 25:481–495, 2003.

[116] D. Maringer and O. Oyewumi. Index tracking with constrained portfolios. *Intelligent Systems in Accounting, Finance and Management*, 15:57–71, 2007.

[117] H.M. Markowitz. Portfolio selection. *Journal of Finance*, 7:77–91, 1952.

[118] H.M. Markowitz. *Portfolio selection: Efficient diversification of investments*. John Wiley & Sons, 1959.

[119] W. Michalowski and W. Ogryczak. Extending the MAD portfolio optimization model to incorporate downside risk aversion. *Naval Research Logistics*, 48:185–200, 2001.

[120] R. Moral-Escudero, R. Ruiz-Torrubiano, and A. Suárez. Selection of optimal investment with cardinality constraints. In *Proceedings of the IEEE World Congress on Evolutionary Computation (CEC 2006)*, pages 2382–2388, 2006.

[121] J.M. Mulvey, R.J. Vanderbei, and S.A. Zenios. Robust optimization of large-scale systems. *Operations Research*, 43:264–281, 1995.

[122] J.M. Mulvey and H. Vladimirou. Stochastic network programming for financial planning problems. *Management Science*, 38:1643–1664, 1992.

[123] K. Natarajan, D. Pachamanova, and M. Sim. Incorporating asymmetric distributional information in robust Value-at-Risk optimization. *Management Science*, 54:573–585, 2008.

[124] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.

[125] W. Ogryczak. Multiple criteria linear programming model for portfolio selection. *Annals of Operations Research*, 97:143–162, 2000.

[126] W. Ogryczak and A. Ruszczyński. From stochastic dominance to mean-risk models: Semideviations as risk measures. *European Journal of Operational Research*, 116:33–50, 1999.

[127] W. Ogryczak and A. Ruszczyński. Dual stochastic dominance and related mean-risk models. *SIAM Journal on Optimization*, 13:60–78, 2002.

[128] K.J. Oh, T.Y. Kim, and S. Min. Using genetic algorithm to support portfolio optimization for index fund management. *Expert System with Applications*, 28:371–379, 2005.

[129] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[130] N.R. Patel and M.G. Subrahmanyam. A simple algorithm for optimal portfolio selection with fixed transaction costs. *Management Science*, 38:303–314, 1982.

[131] G. Pflug and D. Wozabal. Ambiguty in portfolio selection. *Quantitative Finance*, 7:435–442, 2007.

[132] G.C. Pflug. *Some remarks on the Value-at-Risk and the Conditional Value-at-Risk*, volume Probabilistic constrained optimization: Methodology and applications, pages 272–281. S. Uryasev eds., Kluwer Academic Publisher, 2000.

[133] D. Pisinger. Core problems in knapsack algorithms. *Operations Research*, 47:570–575, 1999.

[134] G.A. Pogue. An extension of the Markowitz portfolio selection model to include variable transaction costs, short sales, leverage policies and taxes. *Journal of Finance*, 25:1005–1028, 1970.

[135] R.T. Rockafellar and S. Uryasev. Optimization of Conditional Value-at-Risk. *Journal of Risk*, 2:21–41, 2000.

[136] M. Rothschild and J.E. Stiglitz. Increasing risk: I. A definition. *Journal of Economic Theory*, 2:225–243, 1970.

[137] B. Rustem and M. Howe. *Algorithms for Worst-Case Design and Applications to Risk Management*. Princeton University Press, 2002.

[138] A. Schaerf. Local search techniques for constrained portfolio selection problems. *Computational Economics*, 20:177–190, 2002.

[139] W.F. Sharpe. A linear programming approximation for the general portfolio analysis problem. *Journal of Financial and Quantitative Analysis*, 6:1263–1275, 1971.

[140] W.F. Sharpe. Mean-absolute deviation characteristic lines for securities and portfolios. *Management Science*, 18:B1–B13, 1971.

[141] C. Singh. Convex programming with set-inclusive constraints and its applications to generalized linear and fractional programming. *Journal of Optimization Theory and Applications*, 38:33–42, 1982.

[142] K.V. Smith. A transition model for portfolio revision. *Journal of Finance*, 22:425–439, 1967.

[143] A.L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21:1154–1157, 1973.

[144] M.G. Speranza. Linear programming models for portfolio optimization. *Finance*, 14:107–123, 1993.

[145] M.C. Steinbach. Markowitz revisited: Mean-variance models in financial portfolio analysis. *SIAM Review*, 43:31–85, 2001.

[146] R.H. Tütüncü and M. Koenig. Robust asset allocation. *Annals of Operations Research*, 132:157–187, 2004.

[147] K.J. Worzel, C. Vassiadou-Zeniou, and S.A. Zenios. Integrated simulation and optimization models for tracking indices of fixed-income securities. *Operations Research*, 42:223–233, 1994.

[148] L.C. Wu, S.C. Chou, C.C. Yang, and C.S. Ong. Enhanced index investing based on goal programming. *Journal of Portfolio Management*, 33:49–56, 2007.

[149] S. Yitzhaki. Stochastic dominance, mean variance, and Gini's mean difference. *American Economic Review*, 72:178185, 1982.

[150] A. Yoshimoto. The mean-variance approach to portfolio optimization subject to transaction costs. *Journal of the Operation Research Society of Japan*, 39:99–117, 1996.

[151] M.R. Young. A minimax portfolio selection rule with linear programming solution. *Management Science*, 44:673–683, 1998.

[152] S.S. Zhu and M. Fukushima. Worst-Case Conditional Value-at-Risk with application to robust portfolio management. *Operations Research*, 57:1155–1168, 2009.