University of Bergen

In partial fulfillment of MSc. Informatics (Optimization)

The index tracking problem with a limit on portfolio size

Student:
Purity Mutunge

Supervisor:
Prof. Dag Haugland

November 19, 2014

Acknowledgements

I am most thankful to God for the opportunity to study at the University of Bergen and for the strength this far. Sincere gratitude to Prof. Dag Haugland for his guidance, commitment and patience; and for the informatics department – the administration's support, many thanks. Special thanks to S&P Dow Jones Indices for sending a constituent with weights for S&P 400. Thanks to Joanna for the inspirations, the course lecturers, course–mates and all who used Room 4152B, thank you. To HIS, LAGET, BKS and to all my friends, Oyugis, Kakunzas, Else, Judith, Sandves and the list goes on – you made this journey special! Last but not least, deep gratitude to my wider and immediate family; the five of you, Wills for all the support and sacrifices! To you Kate–Alyssa, I am lost for words, thanks for your cooperation; it may be difficult but that does not make it impossible! Thank you. The journey is to begin, there is no end to learning.

Abstract

For a passive fund manager tracking a benchmark, it is not uncommon to select some, and not all the assets in the index to his portfolio. In this thesis, we consider the problem of minimizing the tracking error under the mean–variance formulation which gives us a quadratic objective function. Our model includes a cardinality constraint, that puts a limit on the portfolio size. Our problem is a mixed integer nonlinear problem with a convex, quadratic objective function. For this NP–Hard problem, we apply continuous as well as Lagrangian relaxations. We illustrate a subgradient algorithm, modified to our problem. We also present two construction and three improvement heuristics to this problem. Our approaches are compared to the results of an exact and an interrupted solver and computational time is of interest. Our data sets range from 50–400 (500), with real constituent weights from S&P Dow Jones Indices for the largest set of index.

Contents

List of Tables

List of Figures

List of Algorithms

1 Introduction

In fund management, where wealth from numerous investors is managed, different strategies are employed. The goal is to invest this wealth in order to create more wealth. Making an investment is in itself a risky affair as one is not guaranteed of a gain on wealth. However, managers have the task of making decisions on how to achieve this goal while avoiding losses.

Generally, most companies prefer to raise their capital through equity financing which involves selling shares (stocks). A share is a small part of ownership of a company. This way, they are able to raise capital while avoiding paying interests which would have been the case if they opted for debt financing (borrowing loans). When a company opens up its ownership by issuing of stocks and then registering itself in a stock market, it opens up more opportunities for exchange of ownership in day–to–day trading. This distribution of ownership means that profits and losses are also shared among the owners of the shares (stockholders). One way for fund managers to increase this wealth is by owning shares of companies – investing in stocks.

Stock markets exist to facilitate the exchange of shares from one owner to another. A stock market index is a group of stocks used to measure the value of a section of the stock market, a tool used by investors to describe the market [7]. These indices are comprised of groups of various companies; companies that have met predefined criteria in order to qualify. Examples of stock markets indices are S&P (Standard and Poor's) Dow Jones Indices, the New York Stock Exchange (NYSE), NASDAQ, FTSE, among others. In fund management therefore, the stock market is a big area of interest.

There exist different strategies that fund managers have, depending on their attitude toward risk. Those managers who are known to be risk averse are the passive fund managers while those that are risk takers are called active fund managers. Active fund managers aim at outdoing the stock market and thus conduct much research in order to decide where to invest their funds while avoiding potential losses. They try to pick attractive stocks and also watch out for when to move in or out of the market sectors and may often develop complex trading systems. On the other hand, passive fund managers make little or no use of the information that active managers look out for but instead make decisions based upon long–term historical data [9]. Generally, they prefer tracking a benchmark, usually a stock market index.

This tracking may involve selecting all the stocks in a benchmark index, in the same proportions as the index. Such an approach, that is, full replication, is generally not practical in real life. There is the option to track a benchmark without selecting all the stocks, a practice that passive fund managers use. It is not surprising that active managers also limit the number of stocks in their portfolios [10]. A portfolio is in this case any combination of stocks that form a subset of a given index. The task is therefore, that of choosing the best portfolio.

The measure of difference between a portfolio and the benchmark which it mimics is referred to as a tracking error. The best portfolio is thus a portfolio that produces the lowest tracking error.

Different formulations of the index tracking error are proposed, one of which is the mean absolute deviation which is linear in nature. Another formulation makes use of standard deviations while another, known as the mean–variance formulation, makes use of a covariance matrix of the stock returns; further described in Section 2.1.

The problem we attempt to solve is that of minimizing the tracking error with a limit on the portfolio size. In using the mean–variance formulation, as formulated by Jansen and Dijk [6], we are presented with a quadratic objective function, which is what we adopt for this thesis.

The constraints to the problem include: a constraint that ensures that all the assets selected to the portfolio total up to 1, as is the case for the benchmark index. A cardinality constraint, that is, one that puts a limit on the number of assets allowed in a given portfolio, is an aspect that introduces integrality to the problem. This means that an asset is either chosen or not chosen, since it is not possible to pick assets in fractions. In addition, we have a constraint that ensures harmony in the allocation of weights such that only the selected assets are considered in the apportioning of weights to total up to 1. With these constraints, we have a mixed integer problem since some variables are continuous and others are integers. Our problem is therefore formulated as a mixed integer nonlinear problem with a convex, quadratic objective function.

The cardinality constraint to the index tracking problem makes this problem difficult, that is, non–deterministic polynomial time hard (NP-Hard) [1]. An NP–Hard problem is one that is very unlikely to be solvable in polynomial time. This NP–Hardness implies that solving instances of benchmarks with many assets is computationally infeasible and thus inexact approaches are proposed.

Finding exact optimal solutions is possible for small–sized benchmarks and portfolios comprising of few assets. Such instances may be solved by exact solvers, for instance, CPLEX, in general. However, when time is of interest and the benchmark as well as the portfolio sizes increase, we note that it is a challenge for the exact solver, in this case, CPLEX, to solve instances of realistic sizes to optimality in reasonable time. It is possible to interrupt the exact solver, henceforth referred to as the interrupted solver, by putting a time limit for each instance it is employed.

Continuous and Lagrangian relaxations are applied to the problem with a goal of obtaining lower bounds. Getting good lower bounds may however not be an easy task. A subgradient algorithm is also considered in order to get the best lower bound, if possible. We study the strength of the continuous relaxation as well as the theory of Lagrangian relaxations on mixed integer problems with quadratic objective functions.

On the other hand, heuristics may aid in obtaining feasible solutions which

are upper bounds to the problem. Improvement heuristics are proposed to improve the solutions obtained by construction heuristics which construct feasible solutions as their names imply.

These solutions obtained by the relaxations and the heuristics could also supply valuable information to a passive fund manager. Further, a comparison of the lower bound to the upper bound of a given instance, gives valuable information about where an optimal solution to the original problem lies.

With time being of interest, we evaluate the performance of the exact solver and accord it a fair amount of time to solve the given instances. We then compare its results with those of the heuristics. We also compare some of the solutions from the subgradient algorithm to those from the heuristics. A mathematical programming language is used to code while MATLAB is used for data generation for most of the instances.

This thesis is organized as follows: We present a review of literature related to the index tracking problem with a limit on portfolio size in Section 2. Section 3 is a mathematical presentation of the index tracking problem. In Section 4, we present relaxations to the problem, continuous and Lagrangian, Section 5 presents construction and improvement heuristic approaches. Finally, Sections 6 and 7 present the computational results and the conclusions, respectively.

2 Literature Review

2.1 Tracking Error Definitions

The tracking error (TE) can be defined as the measure of the difference between the returns of the tracking portfolio and the returns of the index. Roßbach and Karlow [8] formulate the tracking error in time period t as:

$$TE_{GEN,t} = R_t - v_t \tag{1}$$

where

R_t= return of the tracking portfolio in period t,

v_t= return of the index in period t,

and R_t is given as the sum of the total weights of the assets in the tracking portfolio multiplied by their respective returns in period t.

There are several approaches used in measuring the tracking error. One of these is the mean absolute deviation which measures the mean of the absolute differences between the benchmark and the portfolio returns. Over a set of time periods $t = 1, \ldots, T$, this can be formulated as [8]:

$$TE_{MAD} = \frac{1}{T}\sum_{t=1}^{T}|R_t - v_t|. \tag{2}$$

Rudolf et al. [2] present alternative definitions of the tracking error based on the absolute deviations between the benchmark and the portfolio returns. These models are compared to the tracking error model using the mean square approach which is defined as the square root of the sum of the squared deviations between the portfolio and the benchmark and can be formulated as [3]:

$$TE_{SD} = \frac{1}{T}\left(\sum_{t=1}^{T}|R_t - v_t|^{\alpha}\right)^{\frac{1}{\alpha}} \tag{3}$$

where $\alpha = 2$.

It is observable that for TE_{MAD}, $\alpha = 1$, which makes TE_{MAD} a special case of TE_{SD}. Using the α–norm of the difference vector $R - v$, it is possible to have variants of this formulation with $\alpha \geq 1$. There may be advantages to varying the values of α as Beasley et al. [3] remark. The higher the values of α, the greater the dominance of the large terms of the vector $R - v$. Taking this to the extreme, i.e. letting $\alpha \to \infty$, TE_{SD} approaches $\max_{t=1,\ldots,T}|R_t - v_t|$, also referred to as the infinity norm of $R - v$.

Another formulation is the tracking error variance, which Jansen and Dijk [6] formulate as:

$$TE_{COV} = (x - w)^T Q(x - w) \tag{4}$$

where

x_i is the percentage weight of asset i in the vector x of the portfolio weights, $i \in N$, where $N = \{1, \ldots, n\}$,

w_i is the percentage weight of asset i in the vector w of the benchmark weights,

Q is the covariance matrix of the stock returns.

TE_{MAD} and TE_{SD} are both averaged over a set of T time periods. On the other hand, TE_{COV} uses a covariance matrix which is estimated over a period of time from historical data. This matrix gives the correlation between all possible pairs of assets. The latter function, as Coleman et al. [4] remark, is mathematically more appealing as it is convex and can be used in financial interpretations on the assumption that the covariance matrix is accurate for future returns.

It is worth noting that as observed by Rudolf et al. [2] and as noted by Konno and Yamazaki [5], there exist similarities in performance when using the different approaches. In 2011, Roßbach and Karlow [8], presented a comparative study of these different approaches.

In this thesis, we adapt definition (4), giving us a convex quadratic objective function to be minimized subject to linear constraints. One of the constraints is a cardinality constraint which gives a bound M, to the number of assets in the portfolio,

$$\sum_{i=1}^{n} y_i \leq M,$$

where $y_i = 0$ when $x_i = 0$ and $y_i = 1$, when $x_i > 0$.

2.2 Solution Methods

When a cardinality constraint restricting the number of assets in a portfolio is introduced, the problem of optimizing the selection of assets in a portfolio becomes NP–Hard [4], [10]. This means that exact solutions to instances of realistic sizes are computationally infeasible and thus inexact solutions are the practical ones.

The tracking error function, as defined in (4), is a convex function whereas the cardinality constraint is expressed in terms of a discontinuous counting function, $|\{i \in N : x_i > 0\}| \leq M$. Jansen and Dijk [6] and Coleman et al. [4] present two different approaches to dealing with the discontinuity introduced by the cardinality constraint.

Jansen and Dijk [6], focus on minimizing the tracking error with a relatively small number of stocks by approximating the discontinuous counting function

by a continuous function,

$$\sum_{i=1}^{n} x_i^p$$

for some parameter p. They then compare their suggested diversity method with a sequential quadratic optimization method [6].

Coleman et al. [4] propose a graduated non-convexity (GNC) process for minimizing the tracking error while restricting the number of assets, which builds on the approach suggested by [6]. The discontinuous function is approximated by a continuously differentiable non-convex function,

$$\sum_{i=1}^{n} g_j(x_i; p)$$

with a parameter $j > 0$, which is a large constant set to 10^8. With $p > 0$, $r = \sqrt{\frac{2}{p} + \frac{1}{j}}$ and $q = \frac{1}{jr}$, they define:

$$g_j(x;p) = \begin{cases} jx^2 & \text{if } |x| \leq q \\ 1 - \frac{p}{2}(|x| - r)^2 & \text{if } q \leq |x| < r \\ 1 & \text{otherwise} \end{cases}$$

which is symmetric with respect to x. It is possible to see that g_j is continuously differentiable at $|x| = r$, as well as at $|x| = q$. They solve a sequence of problems with continuously differentiable approximated objective functions. They compare the GNC method with the method proposed by [6] and another heuristic approach that uses (3) as the tracking error definition.

Shaw et al. [10] propose a Lagrangian procedure for portfolio optimization with a cardinality constraint. Their procedure takes advantage of the covariance matrix Q, which they decompose into two parts. This leads to the formulation of two subproblems upon employing Lagrangian relaxation and obtaining a lower bound on the problem. According to [11], when applying Lagrangian relaxation to linear problems, there is a trade–off between the ease of the problem and the strength of the bound. The proposed Lagrangian–relaxed problem in [10] can be solved in polynomial time. By this choice, it is apparent that they prefer faster computations to stronger bounds. They then present a branch and bound procedure, with variable fixing, that attempts to solve the cardinality–constrained problem to optimality and make comparisons of their findings with CPLEX.

Chang et al. [12] present three heuristic algorithms based upon genetic algorithms (GA), tabu search (TS) and simulated annealing (SA) for finding the cardinality constrained efficient frontier. Since different asset combinations are considered, the cardinality constraint introduces discontinuity to the efficient frontier, which when unconstrained is a smooth non–decreasing curve that gives the best possible trade–off of risk and return. This also makes some parts of the

curve become invisible. In addition to the cardinality constraint, they limit the minimum and the maximum proportion that can be held by any asset, if any is held. The performance of the proposed heuristic algorithms is compared to the unconstrained efficient frontier in order to measure closeness to optimality. They present pooled results, that is, a combination of the results of GA, TS and SA applied on set G, and conclude that combination of the heuristics to be a sensible approach to the problem.

Li et al. [13] propose an exact solution algorithm in obtaining an optimal lot solution to cardinality constrained mean–variance formulation for portfolio selection under concave transaction costs. They make use of special features in the mean–variance formulation, which involve the covariance matrix Q, used in (4). Computational results with and without the cardinality constraint are presented. With their approach, they observe that the search for a better feasible point is more difficult when a cardinality constraint is included.

3 Problem Formulation

Out of the n available investment assets, an investor endeavours to select a portfolio of size not greater than M, that minimizes the tracking error as defined in (4), Section 2, that is,

$$TE_{COV} = (x - w)^T Q(x - w).$$

As explained earlier in Section 2.1, w is an n–dimensional column vector of given benchmark weights, x is an n–dimensional column vector of portfolio weights and Q is an nxn positive semi–definite covariance matrix of the stock returns.

For the cardinality constraint, introduced in Section 2, 0–1 decision variables are introduced:

$$y_i = \begin{cases} 1, & \text{if asset } i \ (i \in N) \text{ is included in the portfolio} \\ 0, & \text{otherwise,} \end{cases}$$

where, M with $M \leq n$, is a given positive integer denoting the maximum number of assets allowed in a portfolio, and the constraint is

$$\sum_{i=1}^{n} y_i \leq M.$$

A constraint to ensure that the proportions included in the portfolio sum up to one is given as

$$\sum_{i=1}^{n} x_i = 1.$$

We further define a constraint that links y_i and x_i. Since variable y_i is binary, if asset i is included in the portfolio, then $y_i = 1$ and $x_i \leq 1$ and therefore $y_i - x_i$ cannot be less than 0. If asset i is not included in the portfolio, $y_i = x_i = 0$ and thus we have the constraint,

$$y_i - x_i \geq 0.$$

A valid formulation of the tracking error problem with a limit on the portfolio size is thus:

(P)

$$\min_{x} \quad TE_{COV} = (x-w)^T Q(x-w) \tag{5}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_i = 1, \tag{6}$$

$$\sum_{i=1}^{n} y_i \leq M, \tag{7}$$

$$y_i - x_i \geq 0 \qquad i \in N, \tag{8}$$

$$y \in \{0,1\}^n, \tag{9}$$

$$x \in \mathbb{R}^n_+, \tag{10}$$

which is a mixed integer nonlinear problem with a convex, quadratic objective function.

4 Relaxations

In order to define a relaxation, we consider a general optimization problem,

$$z = \min_{x \in X} \{f(x) : g_i(x) \geq 0, i = 1, \ldots, m\},$$

which can be formulated as

$$z = \min_{x} \{f(x) : x \in S\},$$

with $S = \{x \in X : g_i(x) \geq 0, i = 1, \ldots, m\}$, which is the feasible region.

A relaxation for this problem, can in general be defined as,

$$z^R = \min_{x} \{c(x) : x \in F\},$$

where $F \supseteq S$ and $c(x) \leq f(x)\ \forall x \in S$.

The idea behind relaxation as a way of dealing with optimization problems, is to obtain lower bounds on the optimal objective function value(s) of the original problem. It is an important characteristic that a relaxation problem is easier to solve than the original one. A relaxation can be constructed, for instance, by either removing or replacing the constraints such that the feasible region is extended.

4.1 Lower Bounds

For a minimization problem, as introduced above, the optimal value from such a relaxation provides a lower bound on the optimal objective function value of the original problem (P), such that,

$$z^R \leq z.$$

Lower bounds provide useful information for narrowing down a search in tree search algorithms, for instance, in branch and bound for discrete optimization problems.

While satisfying the optimality conditions, it is possible that the optimal value of the relaxation is equal to the optimal value of the original problem such that,

$$z^R = z,$$

which may be used to terminate a solution process in an exhaustive search.

How good a bound is may however, be determined by how good the constructed relaxation is. The smaller the gap $z - z^R$, the better, or the stronger the bound. The quality of the bound and the ease of the relaxed problem are, for certain problems, inversely related in that the easier it is to solve the relaxed problem, the weaker the bound is likely to be, and the converse applies [11], [14]. This therefore means that there exists a trade–off between the ease of the relaxed problem and the quality of the bound.

4.2 Examples of Relaxations

In this section, we discuss two frequently used relaxations, namely continuous relaxation and Lagrangian relaxation.

4.2.1 Continuous Relaxation

For an arbitrary problem where some of the variables are constrained to take only integer values, allowing such variables to take values that are non-integral leads to a continuous relaxation.

To explain this further, we consider constraint (9) for the problem (P) in Section 3, where y_i is a 0–1 variable, which means that variable y_i is constrained to take discrete values, that is, $y_i \in \{0, 1\}$.

A continuous relaxation would involve replacing such a constraint with a constraint that allows variable y_i to take any value between 0 and 1, that is, $y_i \in [0, 1]$. The feasible region is thus enlarged while leaving the objective function unchanged. The relaxed problem is thus formulated as:

$$\begin{aligned}
\min_{x} \quad & TE_{COV} \\
\text{subject to} \quad & \sum_{i=1}^{n} x_i = 1, && (11)\\
& \sum_{i=1}^{n} y_i \leq M, && (12)\\
& y_i - x_i \geq 0, \quad i \in N, && (13)\\
& y \in [0, 1]^n && (14)\\
& x \in \mathbb{R}^n_+. && (15)
\end{aligned}$$

This continuously relaxed problem, which is solvable in polynomial time [15], [16], replaces the mixed integer nonlinear problem (P), and its solution gives a lower bound on the optimal objective function value of (P).

4.2.2 Lagrangian Relaxation

Lagrangian relaxation involves removing one, some, or all the constraints. Each constraint removed is multiplied with a penalty variable, that is, each constraint is dualized, and then added to the objective function. This penalty variable is referred to as the dual variable or the Lagrangian multiplier. Since one, some, or all constraints may be dualized, it is possible to generate as many as $2^k - 1$ subproblems, where k is the number of constraints. These subproblems are unique as they differ from each other depending on the constraints dualized.

To apply the relaxation, we consider problem (P) in Section 3, where we may choose to remove, for instance, the cardinality constraint (7). We then introduce

a nonpositive Lagrangian multiplier v, then multiply it with the constraint to get,

$$v\left(M-\sum_{i=1}^{n}y_i\right),$$

which when added to the the objective function TE_{COV}, gives us subproblem $\mathcal{L}(\cdot,v,\cdot)$. An analogous notation will be applied for all the possible subproblems. The optimal objective function value, $L(\cdot,v,\cdot)$, to this subproblem, is obtained by solving,

$$\begin{aligned} L(\cdot,v,\cdot)=\min_{x,y}\quad & TE_{COV}+v\left(M-\sum_{i=1}^{n}y_i\right) \\ \text{subject to}\quad & \sum_{i=1}^{n}x_i=1, \\ & y_i-x_i\geq 0, \qquad i\in N, \\ & y\in\{0,1\}^n, \\ & x\in\mathbb{R}^n_+. \end{aligned} \tag{16}$$

It is easy to show that for all feasible solutions, the objective function of the subproblem $\mathcal{L}(\cdot,v,\cdot)$, takes a value less than or equal to the value of objective function of the problem (P). The feasible region in this relaxation is also enlarged and as introduced in Section 4, $\mathcal{L}(\cdot,v,\cdot)$ qualifies as a relaxation, giving a lower bound on the optimal value of problem (P).

4.3 Improving the bound

For the subproblem $\mathcal{L}(\cdot,v,\cdot)$ in Section 4.2.2, each nonpositive value of v gives a lower bound on the optimal objective function value of problem (P). Thus we may have as many lower bounds as there are different nonpositive values of v. The best bound is the largest possible, the one closest to the optimal value of (P), that would give the smallest gap as mentioned in Section 4.1. In order to find it, we may solve the Lagrangian dual, $\mathcal{L}^v$. For all the possible Lagrangian dual problems, an analogous notation will be applied.

This Lagrangian dual $\mathcal{L}^v$, is an optimization problem defined as,

$$\mathcal{L}^v=\max_{v\leq 0}L(\cdot,v,\cdot),$$

and solving it amounts to computing the value of v, maximizing the optimal objective function value of the Lagrangian relaxation $\mathcal{L}(\cdot,v,\cdot)$. A possible solution method for solving $\mathcal{L}^v$ would for instance, be the subgradient algorithm [11], [14], as explained in Section 4.7.

4.4 Strength of the Continuous Relaxation

If we could track the benchmark portfolio perfectly, then we would have $x_i=w_i$ for $i\in N$, and thus the optimal objective function value would be $TE_{COV}=0$,

since the covariance matrix Q is positive semi-definite. With a positive definite Q matrix, which is likely to be the case in practice, this would be possible only if we are allowed to pick all the assets, that is, $M = n$.

Proposition 4.1. *The optimal objective function value of the continuous relaxation of the tracking error problem as defined in Section 4.2.1, is 0.*

Proof. We prove that $x_i = w_i = y_i$ for $i \in N$, is feasible in the continuous relaxation. Constraints (11) - (15) are satisfied as follows: Since $\sum_{i=1}^{n} w_i = 1$, it follows that $\sum_{i=1}^{n} x_i = 1$, thus constraint (11) is satisfied. The cardinality constraint (12), is satisfied with $\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} w_i = 1 \leq M$. Constraint (13) is satisfied by an equality since $x_i = y_i$ for $i \in N$ and thus $y_i - x_i = 0$ for $i \in N$. Also, for $i \in N$, since $y_i = w_i$ with $0 \leq w_i \leq 1$, and $x_i = w_i \geq 0$, constraints (14) and (15) are satisfied respectively. With the above values of x and y, the objective function value is 0. As Q is positive semi-definite, it follows that (x, y) is also optimal in the continuous relaxation, and thus has an optimal objective function value 0. □

It follows from Proposition 4.1 that the continuous relaxation produces a trivial bound. Further, it follows that any relaxation whose bound is no better than that of the continuous relaxation, gives an objective function value of 0. Examples of such relaxations are given in Section 4.6.1.

A possible branch-and-bound tree, with the continuous relaxation at the root of the tree, would involve fixing some variables $y_i = 0$ or $y_i = 1$, for $i \in N$ in the branching process. With this in mind, we define the following problem that leads to the proposition below.

Let $N^0 \subseteq N$ and $N^1 \subseteq N$, where $|N^1| < M$, and consider the problem where the constraints $y_i = 0$ for all $i \in N^0$ and $y_i = 1$ for all $i \in N^1$, are added to the continuous relaxation given in Section 4.2.1.

Proposition 4.2. *If $N^0 = \emptyset$, this problem has an optimal objective function value 0. If $N^0 \neq \emptyset$, and Q is positive definite, then the problem has a positive optimal objective function value.*

Proof. When $N^0 = \emptyset$, $x_i = w_i \leq y_i$ for $i \in N$, is feasible for this problem. Constraints (11) and (15) are satisfied as in Proposition 4.1. Since $|N^1| < M$ and $y_i = 1$ for $i \in N^1$, then $\sum_{i \in N^1} y_i \leq M-1$ and by letting $y_i = w_i$ for $i \notin N^1$, then $\sum_{i \in N \setminus N^1} y_i \leq 1$ as $1 = \sum_{i \in N} w_i \geq \sum_{i \in N \setminus N^1} w_i$; and thus Constraint (12) is satisfied with $\sum_{i=1}^{n} y_i \leq M$. It also follows that constraint (13) is satisfied with an equality as in Proposition 4.1 for $i \notin N^1$, and with an inequality $y_i - x_i > 0$ for $i \in N^1$. Constraint (14) is satisfied with $0 \leq y_i = w_i \leq 1$ for $i \notin N^1$ and $y_i = 1$ for $i \in N^1$. These values of x and y give a similar result as in Proposition 4.1. However, when $N^0 \neq \emptyset$, $y_i = 0$ for $i \in N^0$. Constraint (13) is to be satisfied with $y_i - x_i \geq 0$ for $i \in N$, this means that for the corresponding $i \in N^0$, $x_i = 0$ and therefore, $x_i \neq w_i$ for $i \in N^0$. Since $x - w \neq 0$, it follows that $TE_{COV} = (x - w)^T Q (x - w) > 0$, if Q is positive definite. □

Therefore, with reference to Section 4.1, the continuous relaxation of the tracking error problem does not provide useful bound information. However, Proposition 4.2 shows that in given circumstances, if it is decided beforehand that some assets in the benchmark cannot be part of the portfolio of M assets, then, the continuous relaxation can give a positive bound.

Thus, while processing the nodes in a branch and bound tree, except at the root node and the nodes where only $y_i = 1$, then it is possible to obtain nontrivial bounds.

4.5 Theory of Lagrangian Relaxation of Mixed Integer Problems with Quadratic Objective Functions

In mixed integer linear problems, Lagrangian relaxation may in some instances, be used as an alternative in order to obtain better bounds than those that could be obtained by solving the original problem while ignoring or dropping the integrality constraints. However, when Lagrangian relaxation is applied to mixed integer linear problems, a choice is to be made between the ease of the relaxed problem and the strength of the bound to be obtained.

When the Lagrangian subproblem for a mixed integer linear problem is solvable in polynomial time, that is, when the subproblem is easily solvable, the resulting bound is fairly weak. A fairly weak bound means a bound that is no better than the bound obtained by solving the linear relaxation of the original problem [11]. It is worth observing that this trade–off applies also to *nonlinear* integer problems.

Li and Sun [14], prove that the Lagrangian bound for an integer problem with a convex objective function is at least as good as the bound obtained by the continuous relaxation. They further explain that when some of the functions for the problem are nonseparable, the Lagrangian subproblem is not easier to solve than the original one.

Li and Sun [14] show that when the Lagrangian subproblem for an integer nonlinear problem is solvable in polynomial time, the resulting bound is one that is no better than the bound obtained by the continuous relaxation. When the Lagrangian subproblem is difficult, it may give a better bound on the optimal objective function value of the nonlinear integer problem.

This theory will be taken into consideration when choosing the relaxations for this thesis. The discussions in Section 4.6 for the problem (P) will be built upon this theory.

4.6 Lagrangian Relaxation for the Index Tracking Problem

We now consider the possible subproblems when Lagrangian relaxation is applied to problem (P) as formulated in Section 3. For the relaxation of constraints

(8), each of the n constraints shall not be relaxed separately, we dualize either all or none of them. We thus consider the 7 Lagrangian subproblems arising from the different combinations of constraints (6), (7) and (8).

The subproblems may give varying lower bounds to the optimal value of problem (P) as they have varying levels of difficulty, as explained in Section 4.5. We categorize the subproblems according to their ease of solution (strength of bound) and present the two categories in Sections 4.6.1 and 4.6.2.

4.6.1 Easy subproblems, fairly weak bounds

Separability is one of the properties that often lead to easy problems [11]. Whenever constraints (8) are dualized, the subproblem decouples further into two subproblems, with respect to variables x and y. This decoupling results to a linear subproblem with regard to variable y and a quadratic problem with regard to variable x.

We consider subproblem $\mathcal{L}(\cdot, \cdot, z)$, one of the 7 combinations where only constraints (8) are dualized, each with a nonnegative parameter z_i. This subproblem decouples to (17) - (19) and (20) - (22), that is,

$$\min_x \quad TE_{COV} + \sum_{i=1}^{n} z_i x_i \tag{17}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_i = 1, \tag{18}$$

$$x \in \mathbb{R}^n_+, \tag{19}$$

and

$$\min_y \quad -\sum_{i=1}^{n} z_i y_i \tag{20}$$

$$\text{subject to} \quad \sum_{i=1}^{n} y_i \leq M. \tag{21}$$

$$y \in \{0,1\}^n. \tag{22}$$

Subproblem (17) - (19) is a quadratic problem that is known to be solvable in polynomial time [15]. Similarly, subproblem (20) – (22) is easily solvable in that it involves sorting the assets in a nonincreasing manner with respect to the values $z_1, \ldots, z_n$, and picking those with the largest values to a total of M. From Section 4.5, it thus follows that the optimal values of these subproblems give fairly weak bounds to the optimal value of problem (P); that is, a bound that is equal to that of a continuous relaxation, as in Section 4.2.1.

A similar observation is made in three other subproblems, that is when one or both of constraints (6) and (7) are relaxed together with (8). These four subproblems thus form the category of easy subproblems. From this category,

for implementation purposes, we choose the subproblem $\mathcal{L}(u, \cdot, z)$, which relaxes constraints (6) and (8).

4.6.2 Hard subproblems, stronger bounds

For the other 3 combinations, where constraints (8) are not relaxed, there exists some level of difficulty. The subproblems are nonseparable and thus may not be easier to solve than problem (P) [14]. We believe that the subproblems in this group are NP-Hard. A possible reduction could be done from the unconstrained quadratic 0–1 problem which is known to be NP–hard, we are however not aware of any proof for this.

One of the subproblems in this category, which we choose for implementation, is subproblem $\mathcal{L}(\cdot, v, \cdot)$ as given in Section 4.2.2. The other two are formulated when only constraint (6) is relaxed, and when both constraints (6) and (7) are relaxed, respectively.

With reference to the trade–off as explained in Section 4.5, the optimal values of these subproblems have the potential to give better, or stronger bounds to the optimal value of problem (P).

4.7 Subgradient Algorithm

As introduced in Section 4.3, a possible way to get the best lower bound from a Lagrangian relaxation involves solving the Lagrangian dual using a subgradient algorithm.

In our case, in order to get the best bound for the subproblems $\mathcal{L}(u, \cdot, z)$ and $\mathcal{L}(\cdot, v, \cdot)$, chosen in Sections 4.6.1 and 4.6.2, respectively, we employ the subgradient algorithm with some modifications and solve the corresponding Lagrangian duals $\mathcal{L}^{uz}$ and $\mathcal{L}^{v}$. However, for the easy subproblem as discussed in Section 4.6.1, we observe from Proposition 4.1, that such a relaxation gives a bound that is no better than that of the continuous relaxation, that is an objective function value of 0.

For the hard subproblem, as discussed in Section 4.6.2, each nonpositive value of v gives a lower bound. The idea behind the algorithm is to aid in obtaining the best bound possible by iteratively altering the value of the multiplier v for the given subproblem, in this case subproblem $\mathcal{L}(\cdot, v, \cdot)$.

Consider the Lagrangian dual $\mathcal{L}^{v}$ as presented in Section 4.3:

$$\mathcal{L}^{v} = \max_{v \leq 0} L(\cdot, v, \cdot), \tag{23}$$

We present our modified subgradient algorithm for $\mathcal{L}^{v}$, applied to our problem as follows:

Algorithm 1 Modified Subgradient Algorithm

Set $v \leftarrow v^0 = 0$; $\mu_1, \mu_2 \leftarrow$ suitable value(s) $k \leftarrow 0$
repeat
  Solve $\mathcal{L}(\cdot, v, \cdot)$ and obtain lower bound z^R
  if $\sum_{i=1}^{n} y_i < M$ **then**
    $v^{k+1} \leftarrow \min\{v^k + \mu_{1k}(M - \sum_{i=1}^{n} y_i), 0\}$
  else
    $v^{k+1} \leftarrow \min\{v^k + \mu_{2k}(M - \sum_{i=1}^{n} y_i), 0\}$
  Update μ_{1k}, μ_{2k}
until convergence or $k = k^*$
return z^R

In solving $\mathcal{L}^v$, we apply two modifications to the subgradient algorithm. One of the aspects of the modification is the introduction of a time limit. In essence, the process of solving the subproblem should not be interrupted, we however do this because after numerous experiments, we observe that to solve subsequent iterations is time consuming.

When we interrupt a branch and bound solver, in our case, CPLEX, the output comprises of the upper and lower bounds that the solver has found in a branch and bound tree at that time of interruption. We therefore are not guaranteed of the best lower bound possible. In a branch and bound tree, a node may have an upper bound which might as well be the optimal solution, but for this solution to be proved to be optimal, further branching is necessary until when the upper and the lower bounds are equal.

We apply two different stopping criteria. One stopping criterion is after k^* iterations where k^* is predetermined by considering a time limit. It is also possible that convergence is achieved, which is the second stopping criterion, that is, when we allow improvements to be made until no further move can be made. The challenge is to choose which criterion to apply to which problem, and as we observe from Section 6.5, we may not base it on the size of the benchmark neither on the portfolio size.

For an initial multiplier v, to all the instances solved, we use an initial value of $v = 0$ and allow this value to vary as the iterations increase. With such an initial value of $v = 0$, solving for $k = 1$, implies obtaining a trivial bound as expected since all assets would be picked into the portfolio.

The value of the multiplier is adjusted in each iteration based on whether the relaxed cardinality constraint (7), is satisfied or not. Whenever satisfied, v is rewarded, that is the absolute value is decreased, whereas whenever the constraint is not satisfied, v is penalized, that is the absolute value is increased.

The value of v is adjusted using the step lengths μ_{1k} and μ_{2k}, which are varied. This is the other aspect of our modification. After much trial and error, we observe that tuning a single step length to be used in updating the variable

(multiplier) for each iteration requires much time (to converge), we therefore choose two different step lengths. It may be possible to predetermine the value of the initial step length, perhaps by some simulations, but these computations are beyond the scope of this thesis.

The different step length sizes are assigned such that one is for penalizing and another for rewarding. It is also worth noting that different ways of adjusting the step lengths are considered during the experiments with the aim of obtaining faster convergence. After trial and error on small instances, we employ similar initial step lengths to all the varying sizes of benchmarks with varying portfolio sizes.

Experiments, as in Section 6.5, show that reasonable bounds can be obtained with lots of computations implying more computational time. In general however, a single iteration of the subproblem may be solved relatively faster than solving its respective original problem.

5 Heuristic Approaches

Relaxations, on minimization problems provide lower bounds on the optimal objective function value of the original problems as discussed in Section 4.1. Any feasible solution to a minimization problem, on the other hand, gives an upper bound on the optimal value of the objective function of the original problem. In this section we present the various ways we have used in the search for upper bounds for the index tracking problem with a limit on portfolio size.

5.1 Upper Bounds

Although every feasible solution gives an upper bound to a minimization problem, the main goal is to find a good upper bound. Here, by a good upper bound we mean a bound that is close to the optimal solution, such that the gap $z^U - z$, is minimal, where z^U and z are the upper bound and the optimal solution to the original problem respectively.

As discussed earlier, in Section 2.2, the index tracking problem with a limit on portfolio size known to be NP–Hard which implies that exact solutions to instances of realistic sizes are computationally infeasible. Heuristic approaches can therefore be employed with the goal of obtaining good upper bounds relatively faster.

The aim of a construction method is to find a feasible solution for a given problem by constructing one from an empty set, whereas improvement methods aim at improving an already given feasible solution, if possible, by searching a neighborhood of solutions. Forward and backward greedy algorithms are examples of construction methods while a local search falls in the category of improvement methods.

For the index tracking problem, we employ forward greedy algorithms, giving upper bounds on the original problem (P). For the improvement methods, we import a solution from one of the greedy algorithms as the initial solution and perform local search heuristics to improve the solutions, if possible. It is worth noting that for the local searches, other feasible solutions may also be used as initial solutions.

5.2 Construction Methods

A greedy approach may be applied to the index tracking problem in a myriad of ways. We define two approaches where we start with an empty set of assets, then 'greedily' pick out M assets to satisfy the cardinality constraint (8).

For the first variant of the greedy approach, in each iteration, we pick the best asset that can be included in our portfolio. This best asset is the asset that gives us the lowest tracking error if included in the portfolio. Once an asset is picked, it is factored in the choice of the subsequent assets, as the number of assets allowed in a portfolio increase by one in each iteration. Having fixed the

M discrete variables, our problem becomes convex and solvable in polynomial time.

We present this variant of a forward greedy heuristic algorithm for the index tracking problem with a limit on the portfolio size as follows:

Algorithm 2 Solve–Forward Greedy Heuristic

Set $B \leftarrow \emptyset$
for $j = 1, \ldots, M$ **do**
  for $l \in N \setminus B$ **do**
    $$z_l = \min \left\{ TE_{COV} : \sum_{i=1}^{n} x_i = 1, x_i = 0 \ \forall \ i \in N \setminus B \setminus \{l\}, x \in \mathbb{R}^n_+ \right\}$$
  Find some $l^* \in \arg\min\{z_l : l \in N \setminus B\}$
  $B \leftarrow B \cup \{l^*\}$
Solve (P) and obtain upper bound z^U
return z^U

The second variant of the greedy approach involves picking the largest M assets in the benchmark. With this approach, our problem involves sorting the benchmark weights in descending order, fixing M assets, then solving a convex problem. For each instance therefore, only one quadratic problem is solved. This makes the second variant of the greedy approach relatively faster than the first. The algorithm to this approach is as follows:

Algorithm 3 Sort–Forward Greedy Heuristic

Set $B \leftarrow \emptyset$
for $j = 1, \ldots, M$ **do**
  Find some $l^* \in \arg\max\{w_l : l \in N \setminus B\}$
  $B \leftarrow B \cup \{l^*\}$
Solve (P) and obtain upper bound z^U
return z^U

These construction methods do not necessarily guarantee any closeness to optimality. This means that though they yield feasible solutions, we may not guarantee how minimal the gap $z^U - z$, is, in general. However, it is possible that in some instances, $z^U = z$.

5.3 Improvement Methods

One of the ways in which we may improve an upper bound is by performing a local search. For these local searches, as mentioned earlier, in Section 5.1, the initial feasible solution, that is, the incumbent, could be a solution obtained by either of the forward greedy approaches presented in Section 5.2. There exist

different ways of defining a neighborhood and the procedures for a search. We here present three variants.

All these heuristics involve swapping the assets in the incumbent with those that were not included in the portfolio with the aim of obtaining an improved solution, if any exists.

For variant A, we search for improving swaps by swapping all the assets in the portfolio with those not included in the portfolio and picking the best improvement, if any exists. The asset giving this better solution than the initial is included in the portfolio chosen such that it replaces the asset with which it had been swapped. This is repeated until no further improvement can be achieved.

For the variant B however, instead of looking for the best improvement from among the $M \cdot (n - M)$ possibilities as in variant A, we improve the incumbent as follows: All the assets in the incumbent are enqueued. One asset is dequeued and swapped with all the assets that were not in the portfolio and a best improvement, if any exists, is chosen from the $n - M$ possibilities. The asset that yields the best improvement, replaces the asset with which it was swapped with, and is enqueued. This replacement is done before proceeding to perform swaps for the next asset in the queue. The search for an improvement through swapping continues until the queue is empty. Thus, all the assets in the initial solution and those that provided any improvement are all factored in the search for further improvements, if possible, before the algorithm stops.

For variant C, we perform quick small improvements in that whenever an improvement is found during the swapping process, a replacement is done. As opposed to variants A and B which choose the best improvement out of some possibilities whenever such exist, variant C performs a replacement at the encounter of an improvement, if any exists, and proceeds to the next asset in the queue. Just as with variant B, any asset that gives an improvement is enqueued. Thus, before the algorithm stops, all the assets in the queue are swapped with the assets not included in the portfolio in the search for an improvement, if any exists.

Our local search heuristics for the index tracking problem with a limit on portfolio size expressed in algorithmic terms are as follows:

Algorithm 4 Local Search Heuristic variant A

Require: Initial B, where $|B| = M$, with an upper bound z^U
  repeat
    done $\leftarrow$ true
    for $r \in B$ **do**
      for $l \in N \setminus B$ **do**

$$z_{r,l} = \min \left\{ TE_{COV} : \sum_{i=1}^{n} x_i = 1, x_i = 0 \ \forall i \in \{r\} \cup (N \setminus B \setminus \{l\}), x \in \mathbb{R}^n_+ \right\}$$

    Find some $(\sigma, \beta) \in \arg\min\{z_{r,l} : r \in B, l \in N \setminus B\}$
    if $z_{\sigma,\beta} < z^U$ **then**
      $B \leftarrow B \cup \{\beta\} \setminus \{\sigma\}$
      $z^U \leftarrow z_{\sigma,\beta}$
      done $\leftarrow$ false
  until done
  return z^U

Algorithm 5 Local Search Heuristic variant B

Require: Initial B, where $|B| = M$, with an upper bound z^U
  Put all assets in B into a queue q
  while $q \neq \emptyset$ **do**
    Remove asset r from q
    for $l \in N \setminus B$ **do**

$$z_{r,l} = \min \left\{ TE_{COV} : \sum_{i=1}^{n} x_i = 1, x_i = 0 \ \forall i \in \{r\} \cup (N \setminus B \setminus \{l\}), x \in \mathbb{R}^n_+ \right\}$$

    Find $\beta \in \arg\min\{z_{r,l} : l \in N \setminus B\}$
    if $z_{r,\beta} < z^U$ **then**
      $B \leftarrow B \cup \{\beta\} \setminus \{r\}$
      $z^U \leftarrow z_{r,\beta}$
      Add β to q
  return z^U

Algorithm 6 Local Search Heuristic variant C

Require: Initial B, where $|B| = M$, with an upper bound z^U
  Put all assets in B into a queue q
  while $q \neq \emptyset$ **do**
    Remove asset r from q
    for $l \in N \setminus B$ **do**

$$z_{r,l} = \min \left\{ TE_{COV} : \sum_{i=1}^{n} x_i = 1, x_i = 0 \ \forall i \in \{r\} \cup (N \setminus B \setminus \{l\}), x \in \mathbb{R}^n_+ \right\}$$

      if $z_{r,l} < z^U$ **then**
        $B \leftarrow B \cup \{l\} \setminus \{r\}$
        $z^U \leftarrow z_{r,l}$
        Add l to q
  return z^U

Just as the construction methods cannot guarantee the nearness of the upper bound obtained to the optimal solution for the original problem, so is it with the improvement methods. However, if any improvement is possible, then our solution is guaranteed to be better than it was before and thus closer to the optimal. Moreover, it is also possible that some improvements may yield solutions such that $z^U = z$.

6 Computational Results

In this section, we present the computational results from employing the heuristic approaches as well as the results obtained when the original problem is solved by the exact solver. All the algorithms in this thesis are programmed in AMPL and run on a computer with an Intel Core i5–2400 processor, speed of 3.10GHz, and 3.8GB Memory.

6.1 Data Sets

Just as there are varying numbers of constituents for numerous indices in the real world, we use varied sets of assets (indices) for our implementations. Our index sizes include: 50, 100, 200 and 400 assets. In addition, we also use the following index sizes for Section 6.2: 70, 150, 250, 300 and 500, to obtain more detailed results.

For the benchmark weights, which are deterministic for the purpose of comparison, we generate random sets that total to 1. However, for the 400 assets, we obtained real–world constituents and weights for benchmark index from S&P 400 as of 31 December 2013. The covariance matrices are also randomly generated for each of the sets. These data are generated using MATLAB, and these sets of data are also used for the observations and findings for the relaxations introduced in Section 4.

The bound M, on the portfolio size, is not fixed for any specific sets of assets, rather, we vary its value systematically, and compare the results when the heuristics are employed and when they are not.

Thus, as presented in Section 3, for the problem formulation, for each instance, the given data includes: an n–array of benchmark weights totalling 1, an nxn matrix, Q, and an integer M, where n is the given number of assets in the benchmark.

6.2 Exact Solver Results

We shall in this section present the results obtained when the exact solver is put to task. We observe that the exact solver requires considerably much time to prove that most of its solutions are optimal even for instances where $n = 50$.

Figure 1 summarizes the computational time that the solver uses to solve the given instances, for selected values of $M < n/2$. For $M \geq n/2$, the computational time is expected to decrease as $M \to n$ as we observe in Section 6.3, Table 1, for instance. Thus, when the portfolio is to include almost all the assets, the computational time is almost negligible.

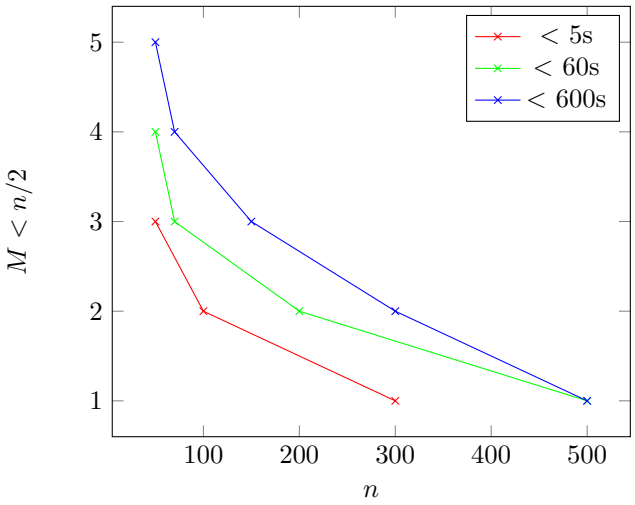

Figure 1: Exact Solver Performance

6.2.1 Limit 600s

Given a time limit of 10 minutes, the exact solver finds a feasible solution but cannot prove it to be optimal while solving an instance where $n = 500$ and $M > 1$. However, within the same amount of time, the solver finds a feasible solution and proves it to be optimal for the following combinations: $n = 50$ with $M = 5$, $n = 70$ with $M = 4$, $n = 150$ with $M = 3$ and $n = 300$ with $M = 2$.

Given that it can solve $n = 300, M = 2$, then it is as observable, possible for the solver to prove optimality for instances $n < 300$ with $M \leq 2$ within the given time limit.

6.2.2 Limit $\leq$ 60s

For $n = 50$ and $M = 4$, $n = 70$ and $M = 3$, $n = 200$ and $M = 2$ and $n = 500$ and $M = 1$, the solver can within 1 minute only solve these instances and prove the solutions to be optimal.

Further, we note that if the time was constrained to 5 seconds, the largest instance the exact solver can solve and yield optimal results for, is an instance where $n = 300$, with $M = 1$. With this time limit and for $M = 2$, we can have a maximum set of $n = 100$ and for $n = 50$, our portfolio cannot have more than three assets.

Figure 1 gives an overview of the capability of the solver to solve instances and prove these solutions to be optimal for the given sets of assets we have worked with. We may not, however, in this thesis, tell exactly how big an instance the solver can handle within given time limits. With a limit of 5 seconds, for instance, and given $M = 2$, for example, we observe that the

maximum that the solver can handle is some $n \in [100, 150)$. We could get the exact set of assets using binary search but this would be time consuming. We therefore give an overview of the results from the categories of the sets of assets we have worked with, as introduced in Section 6.1.

With these observations, we note that it is a challenge for the exact solver to solve instances of realistic sizes to optimality in reasonable time. When we interrupt the solver as we do by putting time constraints, we can, in general, not even be guaranteed to get feasible solutions. However, in our experiments, the solver was able to generate feasible solutions and in the subsequent sections, we analyze these results further as well as those of the heuristics.

6.3 Construction Heuristics Results

We present the findings of the two construction methods we use. We here also observe a trade–off between the time used to solve some of the problem instances and the strength of the solutions obtained. This is as was the case in Section 4.1.

The sort–forward greedy method, that is, the second variant introduced in Section 5.2, in general gives solutions that are relatively weaker compared to its counterpart, the solve–forward greedy algorithm. However, it is notable that in certain instances, these algorithms yield identical results. In addition, there exist cases where $z^U = z$, as we observe in Tables 1 and 5 among others that are presented in this section.

After running the two heuristics, we approximate favourable time limits and use them to interrupt the solver. As mentioned earlier and as we have observed in Section 6.2, since the solver cannot always prove that its solutions are optimal within the given time limits, we may not tell exactly how far our heuristics solutions are from the optimal solutions. For the instances when the solver yields optimal solutions, however, we can tell how far the solutions produced by the heuristics are from their respective optimal solutions. We therefore compare the results obtained from the interrupted solver to those of the heuristics in the subsequent sections. We further highlight some differences between the two heuristics in relation to the solutions yielded.

We use tables to present our findings and in all the tables presented in Section 6.3.1 and 6.3.2: The first column represents the bound M, the second and third columns represent the time used, in seconds, and the tracking error of the interrupted solver respectively; whereas the fourth and the fifth columns represent the time used, in seconds, and the tracking error of the respective heuristic, respectively. The last column informs of the number of assets that make the portfolios chosen by the interrupted solver and the given heuristic non–identical.

For the choices of M, we start with 2 assets in a portfolio and systematically increase the portfolio size by a constant number; this constant number varies as the size of the set of assets, n, varies. We also have an upper limit to the

largest portfolio size and we assume that no investor is to pick all the assets in the benchmark.

For the results reported, where the time required to an instance is less than or equal to 1000 seconds, we rerun these instances 3 times and report the least amount of time required. For the instances requiring more than 1000 seconds, we perform only one run. The limit of time put on the solver, is set to be slightly larger than the highest amount of time taken by the specific heuristics to solve an instance for a given index.

6.3.1 Solve–Forward Greedy Results

As the value of M increases, the amount of time that this algorithm requires increases progressively. As we make comparisons between the results of this heuristic and those of the interrupted solver, we note that in some instances, the solutions are identical and for some of these, we observe that they are optimal solutions. For other instances however, there exists a difference.

50 Assets

The time limit on the solver is 15 seconds, we make the observations as shown in Table 1. We observe that out of the 16 instances solved, the difference in the portfolios chosen by the interrupted solver and the heuristic is at most 1 per portfolio; this is as shown in column 6 of the table.

For 10 of the instances solved, we observe that the interrupted solver and the heuristic yield identical results. Of these, it is worth noting that out of the 8 instances when the solver gives optimal solutions, that is for the instances solved when $M = 2$ and $M \geq 29$, this heuristic gives 6 solutions which are optimal.

Table 1: Results of the interrupted solver and the solve–forward greedy heuristics when $n = 50$ and time limit= 15 seconds

M	Interrupted Solver		Solve–Forward Greedy		# Diff. assets
	Time(s)	TE	Time(s)	TE	
2	0.20	3.307E-1	0.39	3.427E-1	1
5	Limit	1.072E-1	1.23	1.099E-1	1
8	Limit	5.406E-2	2.08	5.487E-2	1
11	Limit	3.062E-2	2.70	3.133E-2	1
14	Limit	1.870E-2	3.66	1.870E-2	0
17	Limit	1.191E-2	4.25	1.191E-2	0
20	Limit	7.639E-3	4.96	7.658E-3	1
23	Limit	4.725E-3	5.75	4.725E-3	0
26	Limit	2.824E-3	6.24	2.824E-3	0
29	4.97	1.762E-3	6.68	1.762E-3	0
32	1.20	1.039E-3	7.31	1.044E-3	1
35	0.38	5.746E-4	7.86	5.746E-4	0
38	0.29	3.413E-4	8.18	3.413E-4	0
41	0.13	1.728E-4	8.40	1.728E-4	0
44	0.10	5.946E-5	8.82	5.946E-5	0
47	0.11	5.013E-5	8.94	5.013E-5	0

100 Assets

For an index of 100 assets and with a time limit of 200 seconds on the solver, we obtain the results as presented in Table 2.

In only 3 out of all the instances solved did the interrupted solver and the solve–forward greedy algorithm produce different results, that is, when $M = 20$, $M = 38$ and $M = 62$. For all the instances with proven optimal solutions, we observe that the results of the heuristic are also optimal solutions.

Table 2: Results of the interrupted solver and the solve–forward greedy heuristics when $n = 100$ and time limit=200 seconds

M	Interrupted Solver		Solve–Forward Greedy		# Diff. assets
	Time(s)	TE	Time(s)	TE	
2	1.16	3.175E-1	3.18	3.175E-1	0
8	Limit	6.357E-2	19.74	6.357E-2	0
14	Limit	2.736E-2	35.69	2.736E-2	0
20	Limit	1.462E-2	50.13	1.467E-2	1
26	Limit	8.454E-3	63.32	8.454E-3	0
32	Limit	5.270E-3	76.00	5.270E-3	0
38	Limit	3.308E-3	87.74	3.333E-3	2
44	Limit	2.046E-3	98.61	2.046E-3	0
50	Limit	1.322E-3	108.65	1.322E-3	0
56	Limit	8.176E-4	118.58	8.176E-4	0
62	Limit	4.797E-4	126.00	4.803E-4	1
68	Limit	2.525E-4	133.08	2.525E-4	0
74	116.85	1.248E-4	138.71	1.248E-4	0
80	13.10	5.971E-5	143.50	5.971E-5	0
86	3.09	2.717E-5	147.37	2.717E-5	0
92	0.83	9.642E-6	151.68	9.642E-6	0

200Assets

For such a large index, we limit our portfolio sizes to not more than 110. We observe from Table 3, that the portfolios differ by 2 assets except for three instances; when $M = 2$, where both the solver and the heuristic yield an optimal solution, when $M = 26$ where the portfolios differ by 3 assets and when $M = 58$, where the portfolios chosen differ by 5 assets.

We note that it is only for one instance among those solved that both the interrupted solver and the heuristic yield a solution that is optimal. For the rest of the instances, the interrupted solver yields feasible solutions but is not able to prove them to be optimal even with a time limit of 40 minutes for each instance.

Table 3: Results of the interrupted solver and the solve–forward greedy heuristics when $n = 200$ and time limit= 2400 seconds

M	Interrupted Solver		Solve–Forward Greedy		# Diff. assets
	Time(s)	TE	Time(s)	TE	
2	17.44	3.326E-1	27.97	3.326E-1	0
10	Limit	5.283E-2	241.33	5.450E-2	2
18	Limit	2.318E-2	450.36	2.385E-2	2
26	Limit	1.254E-2	652.39	1.325E-2	3
34	Limit	7.620E-3	845.61	7.964E-3	2
42	Limit	4.972E-3	1033.49	5.180E-3	2
50	Limit	3.376E-3	1210.68	3.507E-3	2
58	Limit	2.387E-3	1385.48	2.470E-3	5
66	Limit	1.701E-3	1548.59	1.750E-3	2
74	Limit	1.227E-3	1715.08	1.264E-3	2
82	Limit	8.899E-4	1855.79	9.157E-4	2
90	Limit	6.487E-4	1998.67	6.668E-4	2
98	Limit	4.732E-4	2137.82	4.809E-4	2
106	Limit	3.459E-4	2254.86	3.494E-4	2

400Assets

As we observe in Table 4, the solve–forward greedy algorithm requires more than 6 hours to solve the instance with the largest portfolio size among the instances we solve. Therefore, the time set to interrupt the solver is 6 hours and 15 minutes. We observe that this is still not enough time for the solver to prove that most of its feasible solutions are optimal.

For the smallest portfolio size, when $M = 2$, the solver requires almost twice the amount of time required by the heuristic to give a feasible result, which it proves to be optimal. For the rest of the instances solved, the interrupted solver does give relatively better solutions than those of the heuristic, and the difference in the solutions is generally small.

Table 4: Results of the interrupted solver and the solve–forward greedy heuristics when $n = 400$ and time limit= 22500 seconds

M	Interrupted Solver		Solve–Forward Greedy		# Diff. assets
	Time(s)	TE	Time(s)	TE	
2	429.32	3.950E-1	224.82	3.954E-1	1
15	Limit	4.806E-2	3273.15	4.863E-2	5
28	Limit	2.331E-2	6255.00	2.371E-2	11
41	Limit	1.445E-2	9153.59	1.476E-2	16
54	Limit	9.843E-3	11959.00	1.010E-2	15
67	Limit	7.167E-3	14660.50	7.319E-3	10
80	Limit	5.467E-3	17318.10	5.536E-3	12
93	Limit	4.284E-3	19895.70	4.334E-3	12
106	Limit	3.404E-3	22433.10	3.447E-3	9

We observe that in general, this heuristic is time consuming, even though for some instances it is promising. The interrupted solver is capable of obtaining equally good or even better solutions for most of the instances given. As it demands more time in the way it picks assets into the portfolio, it is not surprising that some of its solutions are stronger than those of its counterpart presented in Section 6.3.2.

6.3.2 Sort–Forward Greedy Results

This second variant of the greedy heuristics requires very little time to give a feasible solution regardless of the value of M, the bound on the portfolio size. For most instances, the solutions are weaker than those obtained by the solve–forward greedy approach. The results may be useful as initial solutions for improvement heuristics introduced in Section 5.3.

There are some instances where this sort–forward greedy algorithm yields identical to or even better results than the solve–forward greedy algorithm. In such instances, using this approach, we get good feasible solutions in a few seconds. Even though we cannot determine beforehand the specific instances where we get good solutions, this approach is useful as the portfolio sizes increases with increase in the number of assets in the benchmark indices, and as well as when time is of interest.

Here, we put a time limit varying between 1 and 5 seconds on the solver, and compare these results with those of the sort–forward heuristic, which in general requires a few milliseconds to solve an instance.

50 Assets

We observe, in Table 5, that our sort–forward heuristic yields 5 optimal results out of the 8 known optimal solutions as observed in Table 1, where we

had observed that the solve–forward greedy heuristic yielded 6 optimal solutions. We note that for such instances, the latter heuristic requires on average more than 100 times the amount of time the former needs to obtain these identical results. We observe that when $M = 5$, the sort–forward greedy heuristic gives a better solution as compared to the solve–forward greedy heuristic.

Table 5: Results of the interrupted solver and the sort–forward greedy heuristics when $n = 50$ and time limit= 1 second

M	Interrupted Solver		Sort–Forward Greedy		# Diff. assets
	Time(s)	TE	Time(s)	TE	
2	0.20	3.307E-1	0.03	3.553E-1	1
5	Limit	1.072E-1	0.02	1.072E-1	0
8	Limit	5.406E-2	0.04	5.444E-2	1
11	Limit	3.062E-2	0.03	3.062E-2	0
14	Limit	1.870E-2	0.05	1.873E-2	1
17	Limit	1.191E-2	0.02	1.214E-2	1
20	Limit	7.639E-3	0.03	7.670E-3	1
23	Limit	4.725E-3	0.05	4.725E-3	0
26	Limit	2.824E-3	0.02	2.824E-3	0
29	Limit	1.762E-3	0.02	1.762E-3	0
32	Limit	1.039E-3	0.05	1.044E-3	1
35	0.38	5.746E-4	0.02	5.746E-4	0
38	0.29	3.413E-4	0.04	3.508E-4	1
41	0.13	1.728E-4	0.03	1.728E-4	0
44	0.10	5.946E-5	0.03	5.946E-5	0
47	0.11	5.013E-6	0.03	5.013E-6	0

100 Assets

We observe, in Table 6, that the sort–forward heuristic yields weaker solutions as compared to the solve–forward greedy heuristic as in Table 2. We observe that when the solver is interrupted at 1 second, when $M = 20$, the result obtained is not similar to the one in Table 2. For this specific instance, the sort–forward greedy heuristic yields a better solution faster even though we know from Table 2, that the interrupted solver would yield a better solution given more time.

However, it is worth noting that this heuristic gives solutions very fast, each instance is solved in milliseconds. When $M = 74$, the two heuristics and the interrupted solver yield an identical result. For this specific instance, while the solve–forward greedy requires more than 2 minutes and the solver requires almost two minutes, the sort–forward greedy heuristic requires a few milliseconds to choose an identical portfolio, which we know to be the optimal solution as we observe in Table 2.

Moreover, for the solution when $M = 38$, we observe that the result from the

sort–forward greedy algorithm is better than that of the solve–forward greedy algorithm.

Table 6: Results of the interrupted solver and the sort–forward greedy heuristics when $n = 100$ and time limit= 1 second

M	Interrupted Solver		Sort–Forward Greedy		# Diff. assets
	Time(s)	TE	Time(s)	TE	
2	1.16	3.175E-1	0.06	3.839E-1	2
8	Limit	6.357E-2	0.05	6.514E-2	1
14	Limit	2.736E-2	0.05	2.767E-2	2
20	Limit	1.470E-2	0.05	1.469E-2	2
26	Limit	8.454E-3	0.06	8.832E-3	2
32	Limit	5.270E-3	0.05	5.444E-3	2
38	Limit	3.308E-3	0.07	3.329E-3	1
44	Limit	2.046E-3	0.07	2.108E-3	1
50	Limit	1.322E-3	0.07	1.332E-3	1
56	Limit	8.176E-4	0.07	8.252E-4	1
62	Limit	4.797E-4	0.07	4.852E-4	2
68	Limit	2.525E-4	0.08	2.555E-4	1
74	Limit	1.248E-4	0.08	1.248E-4	0
80	Limit	5.971E-5	0.09	6.019E-5	1
86	Limit	2.717E-5	0.08	2.746E-5	1
92	0.83	9.642E-6	0.09	9.757E-6	1

200 Assets

We observe here that this heuristic yields better solutions than the solutions of the solve–forward greedy heuristic, that is, we get 11 better results out of the 14 instances solved.

Table 7 gives these results and the comparisons are made against Table 3 presented earlier in Section 6.3.1. Further, we observe that when the solver is interrupted after 2 seconds, it yields solutions that are not as good as when it is given more time as we see in Table 3.

The solutions yielded by the heuristic when $M \geq 26$ are in this case better than those given by the interrupted solver except when $M = 42, M = 50$ and $M = 58$. When $M = 18$, the interrupted solver and the sort–forward greedy heuristic yielded identical results.

Table 7: Results of the interrupted solver and the sort–forward greedy heuristics when $n = 200$ and time limit= 2 seconds

M	Interrupted Solver		Sort–Forward Greedy		# Diff. assets
	Time(s)	TE	Time(s)	TE	
2	Limit	3.375E-1	0.09	3.807E-1	2
10	Limit	5.352E-2	0.10	5.406E-2	1
18	Limit	2.362E-2	0.12	2.362E-2	0
26	Limit	1.363E-2	0.11	1.309E-2	1
34	Limit	8.267E-3	0.12	7.700E-3	1
42	Limit	5.030E-3	0.13	5.091E-3	1
50	Limit	3.395E-3	0.12	3.439E-3	2
58	Limit	2.428E-3	0.14	2.455E-3	1
66	Limit	1.865E-3	0.13	1.733E-3	3
74	Limit	1.574E-3	0.14	1.246E-3	2
82	Limit	1.204E-3	0.14	9.011E-4	3
90	Limit	9.473E-4	0.19	6.542E-4	3
98	Limit	7.593E-4	0.19	4.815E-4	4
106	Limit	6.719E-4	0.17	3.612E-4	6

400 Assets

Even though the results of the sort–forward greedy heuristic are not better than those of its counterpart as we observed in the comparison for the 200 index, it is worth noting that in a few milliseconds, this heuristic gives solutions that are close to those of the solve–forward greedy approach.

We note that as the size M of the portfolio increases, that is from when $M = 41$, the results given by the interrupted solver are not as good as those yielded by the heuristic. This is as expected due to the strict time limit. The number of the assets differing between the portfolios increases progressively with increase in M until $M = 80$.

We also observe that when the portfolio comprises of 93 and 106 assets, the interrupted solver yields identical solutions. For these instances, the interrupted solver managed to pick only 88 assets, due to the time limitation. Picking fewer assets than expected implies a larger tracking error; this accounts for the large difference in the assets differing from the portfolios chosen by the sort–forward heuristic and those the interrupted solver picks.

Table 8: Results of the interrupted solver and the sort–forward greedy heuristics when $n = 400$ and time limit= 5 seconds

M	Interrupted Solver		Sort–Forward Greedy		# Diff. assets
	Time(s)	TE	Time(s)	TE	
2	Limit	4.069E-1	0.33	4.205E-1	1
15	Limit	5.051E-2	0.34	5.134E-2	2
28	Limit	2.443E-2	0.34	2.492E-2	6
41	Limit	1.584E-2	0.35	1.501E-2	15
54	Limit	1.162E-2	0.35	1.028E-2	27
67	Limit	9.569E-3	0.45	7.481E-3	42
80	Limit	7.851E-3	0.47	5.705E-3	50
93	Limit	7.238E-3	0.48	4.481E-3	60
106	Limit	7.238E-3	0.48	3.579E-3	70

The sort–forward greedy heuristic is a more favourable heuristic compared to the solve–forward greedy heuristic when time is of concern, with some of its solutions being optimal. In general, the number of assets differing from the results of the interrupted solver, is not large with reference to the portfolio sizes. These results also form good initial solutions for possible improvements, if any exists, as presented in Section 6.4.

6.4 Improvement Heuristics Results

We shall here present the results of the heuristic introduced in Section 5.3. Since we are in search of a swap that yields a better solution, that is, a smaller tracking error, the task requires more computations as the number of assets n and / or the bound M increases. Our solutions here build upon the results obtained from the construction heuristics as presented in Section 6.3.

We compare the improvements, if any exists, with the results of the interrupted solver when the time limit for interruption was larger, that is, the results used in comparison to the solve–forward greedy heuristic findings. These are the results of the interrupted solver as presented in Section 6.3.1, column 6.

Generally, variant B of the improvement heuristic is faster compared to variant A. Variant C is however fastest. As the size of the indices increase with increase in the portfolio sizes, the time taken to perform the swaps generally increases. Therefore, we choose to employ the faster variants, that is B and C, for the improvements with large indices and apply variant A for the improvements with small sized indices. Since the approaches for improvement are different, we cannot tell beforehand whether their results would be identical if they were applied to a given instance.

We also use tables to present our findings. The first column represents the bound M, while the second column represents the tracking error from the interrupted solver. For the third column, we present the initial solution from which

the improvement heuristic departs, which is the tracking error from the sort–forward greedy heuristic. The fourth column represents the total time used to obtain the initial solution and to perform the improvement, in seconds, whereas the fifth column represents the tracking error after employing the improvement heuristic. The sixth column represents the number of assets differing between the results of the interrupted solver and the specific improvement results. The seventh column represents the number of executed improvements.

50 Assets

We here present the results of the improvement when the improvement heuristic variant A is put into use. We observe that in a few seconds, we obtain improvements, if any exists, such that the tracking errors for all the instances solved are equal to those of the interrupted solver. Within the given time limit, the interrupted solver is not able to prove that its solutions are optimal for $M = 5$ until $M = 26$, as we observed in Table 1 in Section 6.3.1.

We note that using the results of the sort–forward greedy heuristic and improving them using variant A local search heuristic, our results are equal to those of the interrupted solver and the time used is less than the time limit on the solver. The largest number of executed improvements is 2 that is, when $M = 38$.

Table 9: Results of the interrupted solver when time limit = 15 seconds and the variant A improvement heuristic when $n = 50$

M	I.Solver	Initial Sol	Imp.Var A		# Diff.Ass	# Improve
	Results		Time(s)	TE		
2	3.307E-1	3.553E-1	1.26	3.307E-1	0	1
5	1.072E-1	1.072E-1	1.54	1.072E-1	0	0
8	5.406E-2	5.444E-2	4.51	5.406E-2	0	1
11	3.062E-2	3.062E-2	2.94	3.062E-2	0	0
14	1.870E-2	1.873E-2	7.04	1.870E-2	0	1
17	1.191E-2	1.214E-2	8.04	1.191E-2	0	1
20	7.639E-3	7.670E-3	8.89	7.639E-3	0	1
23	4.725E-3	4.725E-3	4.77	4.725E-3	0	0
26	2.824E-3	2.824E-3	4.98	2.824E-3	0	0
29	1.762E-3	1.762E-3	5.07	1.762E-3	0	0
32	1.039E-3	1.044E-3	9.89	1.039E-3	0	1
35	5.746E-4	5.746E-4	4.75	5.746E-4	0	0
38	3.413E-4	3.508E-4	12.78	3.413E-4	0	2
41	1.728E-4	1.728E-4	3.66	1.728E-4	0	0
44	5.946E-5	5.946E-5	2.76	5.946E-5	0	0
47	5.013E-6	5.013E-6	1.56	5.013E-6	0	0

100 Assets

We observe in Table 10, that all the improvement solutions are equal to the results of the interrupted solver and that all these solutions are obtained within five minutes.

We note from Table 2 that at least five solutions were optimal, that is, when $M = 2$ and when $M \geq 74$. Table 6 shows that the sort–forward greedy heuristic had results that were close to the optimal solutions. For all the solutions we know to be optimal, variant A improvement heuristic can obtain improvements that are the optimal solutions, in less than two minutes. For all the other results, only in two instances does the heuristic require more than 200 seconds to give the same solutions as the interrupted solver which has a time limit of 200 seconds.

Table 10: Results of the interrupted solver when time limit = 200 seconds and the variant A improvement heuristic when $n = 100$

M	I.Solver	Initial Sol	Imp Var A		# Diff.Ass	# Improve
	Results		Time(s)	TE		
2	3.175E-1	3.839E-1	16.56	3.175E-1	0	2
8	6.357E-2	6.514E-2	42.41	6.357E-2	0	1
14	2.736E-2	2.767E-2	106.17	2.736E-2	0	2
20	1.462E-2	1.469E-2	92.29	1.462E-2	0	1
26	8.454E-3	8.832E-3	159.86	8.454E-3	0	3
32	5.270E-3	5.444E-3	251.75	5.270E-3	0	3
38	3.308E-3	3.329E-3	139.33	3.308E-3	0	1
44	2.046E-3	2.108E-3	157.03	2.046E-3	0	1
50	1.322E-3	1.332E-3	155.02	1.322E-3	0	1
56	8.176E-4	8.252E-4	154.36	8.176E-4	0	1
62	4.797E-4	4.852E-4	297.61	4.797E-4	0	3
68	2.525E-4	2.555E-4	145.47	2.525E-4	0	1
74	1.248E-4	1.248E-4	67.03	1.248E-4	0	0
80	5.971E-5	6.019E-5	114.25	5.971E-5	0	1
86	2.717E-5	2.746E-5	87.99	2.717E-5	0	1
92	9.642E-6	9.757E-6	53.97	9.642E-6	0	1

200 Assets

As we observe in Table 11, we are able to make improvements that give us solutions that are equal to those of the interrupted solver except for three of the instances solved, that is, when $M = 58, 66$ and 74. We also observe that when using the variant B of the improvement heuristic, the total time needed increases with increase in M. This is contrary to what we observed

when employing variant A of the improvement as in Table 10, for instance.

Table 11: Results of the interrupted solver when time limit = 2400 seconds and the variant B improvement heuristic when $n = 200$

M	Int.Solver	Initial Sol	Improvement Var B		# Diff. assets	# Improve
	Results		Time(s)	TE		
2	3.326E-1	3.807E-1	46.14	3.326E-1	0	2
10	5.283E-2	5.406E-2	262.73	5.283E-2	0	3
18	2.318E-2	2.362E-2	459.71	2.318E-2	0	3
26	1.254E-2	1.309E-2	640.63	1.254E-2	0	6
34	7.620E-3	7.700E-3	804.57	7.620E-3	0	2
42	4.972E-3	5.091E-3	954.46	4.972E-3	0	4
50	3.376E-3	3.439E-3	1082.14	3.376E-3	0	3
58	2.387E-3	2.455E-3	1203.02	2.389E-3	1	9
66	1.701E-3	1.733E-3	1312.41	1.702E-3	2	6
74	1.227E-3	1.246E-3	1414.13	1.231E-3	3	7
82	8.899E-4	9.011E-4	1487.30	8.899E-4	0	3
90	6.487E-4	6.542E-4	1578.87	6.487E-4	0	4
98	4.732E-4	4.815E-4	1625.18	4.732E-4	0	3
106	3.459E-4	3.612E-4	1659.11	3.459E-4	0	10

400 Assets

We observe from Table 12, that only in one instance did the improvement lead to a solution that we know to be optimal from Table 4.

In all the other instances, the improvement solutions are close to the solutions of the interrupted solver. The highest relative gap is $2.672 \cdot 10^{-2}$ when $M = 28$. All the solutions of the improvement are obtained in less time than the limit allowed on the interrupted solver.

Table 12: Results of the interrupted solver when time limit = 22500 seconds and the variant B improvement heuristic when $n = 400$

M	Int.Solver Results	Initial Sol	Improvement Var B Time(s)	TE	# Diff.assets	# Improve
2	3.950E-1	4.205E-1	384.37	3.950E-1	0	2
15	4.806E-2	5.134E-2	3466.56	4.830E-2	7	9
28	2.331E-2	2.492E-2	6313.87	2.395E-2	13	16
41	1.445E-2	1.501E-2	8960.92	1.449E-2	9	16
54	9.843E-3	1.028E-2	11487.60	9.856E-3	4	22
67	7.167E-3	7.481E-3	13894.50	7.174E-3	3	22
80	5.467E-3	5.705E-3	16077.50	5.468E-3	6	26
93	4.284E-3	4.481E-3	18239.50	4.299E-3	8	29
106	3.404E-3	3.579E-3	20188.20	3.410E-3	3	33

Noting that the amount of time required by improvement heuristic variant B is above 10 minutes when $M > 15$, we present results of variant C of the improvement heuristic for $n = 400$ in Table 13.

We observe that even though the time used by variant C is not as much as that used by variant B, the solution is weaker. This is as expected from discussions in Section 4.1 and Section 6.3, where the strength of the solution and the time required to obtain the solution are inversely related.

Generally, the number of improvements done by variant C are more than those of variant B except for one instance when $M = 2$. For $M = 2$, we observe that variant B yields a better solution even though both heuristics had two improvements.

When comparing the number of assets differing from those picked by the interrupted solver, generally variant C results have a larger gap than the results of variant B except for two instances. These two instances are when $M = 15$ and $M = 28$. For the former, both variants have a difference of 7 assets but yield varying results; for the latter, variant C result give a smaller number of differing assets but a weaker solution compared to the solution from variant B.

Table 13: Results of the interrupted solver when time limit = 22500 seconds and the variant C improvement heuristic when $n = 400$

M	Int.Solver	Initial Sol	Improvement Var C		# Diff assets	# Imp. Iter
	Results		Time(s)	TE		
2	3.950E-1	4.205E-1	19.80	4.158E-1	2	2
15	4.806E-2	5.134E-2	1426.82	5.027E-2	7	12
28	2.331E-2	2.492E-2	2606.22	2.414E-2	10	22
41	1.445E-2	1.501E-2	6086.62	1.459E-2	12	24
54	9.843E-3	1.028E-2	8394.57	1.011E-2	12	25
67	7.167E-3	7.481E-3	10817.10	7.304E-3	10	26
80	5.467E-3	5.705E-3	12303.10	5.568E-3	10	31
93	4.284E-3	4.481E-3	13706.30	4.344E-3	12	39
106	3.404E-3	3.579E-3	15620.20	3.491E-3	10	36

6.5 Relaxation Results

As introduced in Section 4.7, the subgradient algorithm can aid in obtaining a better lower bound by solving the Lagrangian dual, $\mathcal{L}^v$, of the hard subproblem $\mathcal{L}(\cdot, v, \cdot)$, where the cardinality constraint (7), is relaxed.

We observe that many computations are required in order to achieve convergence, which would imply non–trivial solutions, even possibly an optimal solution where $z^R = z$. These computations also imply more time as well as good techniques of adjusting the step lengths so as to improve the solutions progressively. With our modified version of the subgradient algorithm, we are able to get non-trivial bounds for some of the instances but not all as shown in Table 6.5 below.

As mentioned in Section 4.7, each portfolio with varying number of assets may require different initial guesses of step lengths, we however, apply the same initial step length for the instances given.

For $M = 2$ instances with various combinations of n, the solutions could be found by enumerating $\frac{n(n-1)}{2}$ portfolios; we therefore choose, from the given indices, the least portfolio sizes and present our findings in Table 6.5. This time limit is set according to the time applied in the heuristics in Section 6.3.1.

Column one and two show the index size n, and the portfolio size M, respectively. Column three indicates the value of the dual variable when the algorithm terminates whereas column four gives the best lower bound at the time of interruption. We compare our data with the results of the improvement heuristics from Section 6.4 which are presented in the last column.

Table 14: Results of Lagrangian Relaxation – Hard Subproblem

n	M	v	Best LB	Imp Heur S
50	2	-0.124547	3.307E-1	3.307E-1
100	2	-0.107511	3.124E-1	3.175E-1
200	2	-0.110851	-4.991E-1	3.326E-1
400	2	-0.302415	2.532E-1	3.950E-1

For $n = 50$, with $M = 2$, we set a time limit of 15 seconds per iteration and run the algorithm to convergence which is achieved after 202 iterations. We observe that we obtain a positive lower bound, which is equal to the solution from the improvement heuristic; a case where $z^R = z^U$. It is worth noting that even though a time limit was set to 15 seconds, the initial and the latter iterations used significantly less time. On the other hand, for $n = 100$, with $M = 2$, we observe convergence after 14 iterations. Our time limit was 200 seconds per iteration.

For the other combinations, that is, when $n = 200$, with $M = 2$ and $n = 400$, with $M = 2$, we set k^* to 10. Since the time limit would be 2400 and 22500 seconds respectively, we reduce the total time allocated per iteration to 240 and 22250 seconds respectively.

For $n = 200$, we observe that the bound obtained is trivial. Perhaps, it would be possible to improve this bound given more computational and with a better tuning of the step lengths.

For $n = 400$, convergence happens after 4 iterations, with a total time used of 4507.08 seconds. We note that even though we observe convergence here, there exists a gap between the best lower bound achieved and the upper bound solution, which we know to be the optimal solution – the challenge of interrupting the solver.

The results in column four show how good the solutions from Section 6.3.1, presented in the last column, are. For $n = 50$ with $M = 2$, the lower bound from the relaxation is equal to the upper bound from the improvement heuristic.

The results show that it is possible for the subgradient algorithm to yield good lower bounds, even in this case where $z^R = z^U$ which implies an optimal solution as the results in Section 6.3.1 show. On the other hand, when time is of interest, we observe from the results of $n = 200$, that the result of the algorithm provides no valuable information.

These few observations may however not be conclusive for all instances and as discussed in Section 4.7, a lot is required to have a good initial guess as well as the step lengths used for the instances. A lot of computational time is also needed while using this algorithm.

6.6 In Summary

The exact (or the interrupted) solver is able to generate solutions that are feasible but to prove some of them to be optimal is time–consuming. In 10 minutes, and for the S&P 400 index, the exact solver can only prove an optimal solution for a portfolio comprising of 2 assets.

For the instances with 50 assets, we know of 50% of the solutions to be optimal. Our construction heuristics give 75% of the optimal solutions in a time interval of 0.03 seconds to 8.94 seconds, whereas the improvement methods give 100% of the optimal solutions in a range of 1.26 seconds to 12.78 seconds. In 4 out of the 8 instances with known optimal solutions, our sort–forward greedy heuristic gives these optimal solutions in less time compared to the exact solver.

For the instances with 100 assets, out of the 5 instances where the optimal solutions are known, the sort–forward heuristic generates one of the optimal solution in 8 milliseconds, whereas the solver uses almost 2 minutes to generate an identical solution. The solve–forward greedy heuristic is able to give all the other optimal solutions in a time range of 3.18 seconds to 151.68 seconds. The improvement heuristics require up to 114.25 seconds to give the optimal solutions. The solver uses a range of 0.83 seconds to 116.85 seconds to prove these feasible solutions to be optimal.

For the instances with 200 assets, the solve–forward greedy heuristic gives an optimal solution for the only instance where the optimal solution is known, in less than half a second just as the interrupted solver; the improvement gives a similar solution in 46.14 seconds. Our improvement heuristic variant B is able to generate similar solutions to those of the interrupted solver to all but three instances with a time range of 46.14 seconds to 1659.11 seconds. On the other hand, even though the interrupted solver generated better results when interrupted at 2400 seconds, it was not able to prove that these solutions are optimal for 13 out of the 14 instances solved.

For the instances with 400 assets, the improvement heuristic B is able to generate an optimal solution for the least held portfolio size in less time than the solver. For the rest of the solutions, the improvement heuristic variant B required less time than that used to interrupt the solver, even though its solutions were not as good as those of the interrupted solver. Improvement heuristic variant C does reduce the amount of time required and even though its solutions are relatively weaker. We observe that it is possible to better the results from the sort–forward heuristic in less time, if any improvement exists.

The subgradient algorithm could be helpful in obtaining good lower bounds given suitable dual variables as well as suitable initial and subsequent step lengths. We observe that even with our modified version, a lot of computational time is needed.

7 Conclusions

In this thesis, we have considered the problem of minimizing the index tracking error with a limit on portfolio size and have presented heuristic approaches and attempted relaxations on the problem. We have observed that a continuous relaxation on this problem gives an objective function value of 0, which is a trivial bound. However, we have proposed that a stronger bound using the continuous relaxation is achievable when beforehand, at least one asset is not to be part of the portfolio.

Lagrangian relaxation on hard subproblems may produce good solutions when a suitable dual variable is available albeit more computational time required when employing the subgradient algorithm. Solving this hard subproblem is difficult. Some convergence may be possible while using the subgradient algorithm but this cannot be guaranteed, an area for further research.

The different variants of heuristic approaches, provide valuable information to a fund manager and the general trade–off is between time and the strength of the solution. A hybrid is also possible. The fastest construction heuristic forms a good initial solution to the improvement heuristics in the interest of total computational time. The improvement heuristics do improve initial solutions, if any exist.

Our results indicate that when the benchmark index comprises of few assets, then our construction heuristics could give a fund manager a good choice of portfolio in good time. Improvement heuristics on large benchmark indices generally require a lot of computational time. In using these improvement heuristics, it is possible to constrain the time allowed, or limit the number of iterations on improvements, an area that could be further researched.

8 References

[1] M. Woodside–Oriakhi, C. Lucas, J. E. Beasley: Heuristic algorithms for the cardinality constrained efficient frontier, *European Journal of Operational Research.* 213, 538-550, 2011.

[2] M. Rudolf, H. J. Wolter, H. Zimmermann: A linear model for tracking error minimization, *Journal of Banking & Finance* 23(1), 85–103, 1999.

[3] J. E. Beasley, N. Meade, T. J. Chang: An evolutionary heuristic for the index tracking problem, *European Journal of Operational Research* 148(3), 621–643, 2003.

[4] F. Coleman, L. Yuying, H. Jay: Minimizing tracking error while restricting the number of assets, *Unpublished note* 2006.

[5] H. Konno, H. Yamazaki: Mean absolute deviation portfolio optimization model and its applications to tokyo stock–market *Management Science* 37(5), 519–531, 1991.

[6] R. Jansen, R. van Dijk: Optimal benchmark tracking with small portfolios, *Journal of Portfolio Management* 28(2), 33–39, 2002.

[7] G. Dave: What is a stock index?, `http://www.ehow.com`

[8] P. Roßbach, D. Karlow: The stability of traditional measures of index tracking quality, *Working paper series// Frankfurt School of Finance & Management* No 164, 2011.

[9] E. Steve : A Comparison of Active and Passive Investment Strategies, `http://www.evansonasset.com/`

[10] D. X. Shaw, S. Liu, L. Kopman: Lagrangian relaxation procedure for cardinality–constrained portfolio optimization, *Optimization methods & software* 23(3), 411–420, 2008.

[11] L. Wolsey: Integer Programming, *John Wiley & Sons, Inc.* 1998.

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

[13] D. Li, X. Sun, J. Wang: Optimal lot solution to cardinality–constrained mean-variance formulation for portfolio selection, *Mathematical Finance* 16(1), 83–101, 2006.

[14] D. Li, X. Sun: Nonlinear Integer Programming, *Springer Science + Business Media, LLC.* 2006.

[15] M. K. Kozlov, S. P. Tarasov, L. G. Hacijan: Polynomial solvability of convex quadratic programming, *Soviet Math. Dokl.* 20, 1108- 1111, 1979.

[16] J. Skorin–Kapov Quadratic Programming: Quantitative analysis and Polynomial Running Time Algorithms, *PhD Thesis, The University of British Columbia* 1987.