An Iterated Greedy Heuristic for the 1/N Portfolio Tracking Problem

Oliver Strub and Norbert Trautmann
Department of Business Administration, University of Bern, Schützenmattstrasse 14, Bern, Switzerland

Keywords: 1/*N* Portfolio, Index Tracking, Portfolio Optimization, Iterated Greedy Heuristic.

Abstract: The 1/*N* portfolio represents a simple strategy to invest money in the stock market. Investors who follow this strategy invest an equal proportion of their investment budget in each stock from a given investment universe. Empirical results indicate that this strategy leads to competitive results in terms of risk and return compared to more sophisticated strategies. However, in practice, investing in all *N* stocks from a given investment universe can cause substantial transaction costs if *N* is large or if the market is illiquid. The optimization problem considered in this paper consists of optimally replicating the returns of the 1/*N* portfolio by selecting a small subset of the *N* stocks, and determining the respective weight for each selected stock. For the first time, we apply the concept of iterated greedy heuristics to this novel portfolio-optimization problem. For analyzing the performance of our heuristic approach, we also formulate the problem as a mixed-integer quadratic program (MIQP). Our computational results indicate that, within a limited CPU time, our heuristic approach outperforms the MIQP, in particular when the number of stocks *N* grows large.

1 INTRODUCTION

Stock-investment strategies aim at constructing portfolios of stocks that maximize the expected return for a given level of risk. Besides good risk-return characteristics, a desirable property of an investment strategy is low expenses for, e.g., transaction costs or investment research.

The 1/*N* portfolio represents an investment strategy that delivers good risk-return characteristics at low investment-research costs. An investor who follows the 1/*N* strategy invests an equal proportion of the available budget in each of the *N* stocks from a given investment universe. In an empirical analysis, (DeMiguel et al., 2009) showed that this simple strategy performed competitive in terms of risk and return compared to other more sophisticated investment strategies like the mean-variance approach (Markowitz, 1952) and extensions thereof. However, applying the 1/*N* strategy can lead to substantial transaction costs if the investment universe consists of a large number of stocks or if the market is illiquid.

The planning problem considered in this paper consists of selecting a small subset of the *N* stocks and determining each selected stock's portfolio weight such that the resulting tracking portfolio minimizes the variance of the expected return differences (cf. (Roll, 1992)) between the tracking portfolio and the 1/*N* portfolio at low transaction costs. The following constraints must be fulfilled by the tracking portfolio: The number of different stocks to be included in the tracking portfolio is limited, the weight of each selected stock must be within a specific range, the whole budget must be invested, and short selling is not allowed. We do not explicitly consider transaction costs, but implicitly limit the transaction costs by limiting the number of stocks to be included in the portfolio.

To the best of our knowledge, this planning problem has not been discussed in the literature. However, the 1/*N* portfolio can be interpreted as a stock-market index with equal weights for each stock, and therefore, so-called index-tracking methods known from the literature (cf., e.g., (Beasley et al., 2003; Canakgoz and Beasley, 2008; Guastaroba and Speranza, 2012)) can be applied. However, in general, a financial index consists of stocks with different weights, and general index-tracking methods do not take these weights into account.

In this paper, we present a heuristic approach to the planning problem stated above. To obtain a pure combinatorial optimization problem, we only consider tracking portfolios where each stock has the same weight. We propose an iterated greedy heuristic (cf., e.g., (Ruiz and Stützle, 2007)) that runs as follows. First, a feasible portfolio is constructed by applying a novel greedy insertion heuristic. Then, an improvement phase runs until some termination cri-

Strub, O. and Trautmann, N.
An Iterated Greedy Heuristic for the 1/N Portfolio Tracking Problem.
DOI: 10.5220/0005827704240431
In *Proceedings of 5th the International Conference on Operations Research and Enterprise Systems (ICORES 2016)*, pages 424-431
ISBN: 978-989-758-171-7

terion is met; this improvement phase consists of a deconstruction and a construction sub-phase. During the deconstruction sub-phase, some randomly selected stocks are deleted from the current solution. In the construction sub-phase, stocks are added to the portfolio by applying the greedy insertion heuristic that has also been used to construct the initial solution. New solutions are accepted or discarded based on an acceptance criterion that can also be satisfied for worse solutions with a small probability. Because our paper is the first that considers the problem of tracking the $1/N$ portfolio, we also formulate the problem as a mixed-integer quadratic program (MIQP) to analyze the performance of our heuristic approach. We used a standard commercial solver for the solution of the MIQP. For a set of 23 test instances obtained from real-world stock-market data, it turned out that, in particular for the larger instances, our iterated greedy heuristic (IGH) devises better solutions within less CPU time.

The paper is organized as follows. In Section 2, we state the decision problem. In Section 3, we provide a short overview on the literature on index tracking and on iterated greedy heuristics. In Section 4, we present our iterated greedy heuristic approach. In Section 5, we report on our computational results. In Section 6, we give some concluding remarks and an outlook on future research.

2 PLANNING PROBLEM

The planning problem considered in this paper consists of constructing a portfolio composed of a small subset of the N stocks from a given investment universe to reproduce the returns of the $1/N$ portfolio. More specifically, we want to construct a portfolio that has the lowest possible variance of the relative returns (VRR), where the relative returns correspond to the differences between the returns of the tracking and the $1/N$ portfolio. A lower VRR means more stable relative returns over time and therefore a smaller risk of having large return differences in single periods. (Roll, 1992) formulates the problem of minimizing the VRR between a tracking portfolio and some market index using the following quadratic objective function of the decision variables x_i representing the portfolio weights of each stock i in the tracking portfolio, the weights w_i of each stock i in the index, and the covariances σ_{ij} between the returns of stock i and j (see Table 1 for the nomenclature):

$$\text{Min.} \sum_{i=1}^{N} \sum_{j=1}^{N} \sigma_{ij}(x_i - w_i)(x_j - w_j) \qquad (1)$$

Table 1: Nomenclature for the MIQP.

Parameters	
N	Number of available stocks
k	Maximum cardinality of the portfolio
$\delta_i > 0$	Maximum weight of stock i if included in the tracking portfolio
$\varepsilon_i > 0$	Minimum weight of stock i if included in the tracking portfolio
σ_{ij}	Covariance between returns of stock i and j
w_i	Weight of stock i in the index ($= \frac{1}{N}$)
Decision variables	
x_i	Weight of stock i in the portfolio
z_i	$\begin{cases} = 1, & \text{if } x_i > 0 \\ = 0, & \text{otherwise} \end{cases}$

To construct a portfolio that tracks the $1/N$ portfolio, we replace the index weights w_i by $\frac{1}{N}$. The following constraints are considered: The maximum number of stocks to include in the tracking portfolio must not exceed a prescribed value of k, and each stock included in the portfolio is required to have a weight within the predefined range $[\varepsilon_i, \delta_i]$, $i = 1, \ldots, N$. This range should include $\frac{1}{k}$ such that we are able to construct a portfolio of k stocks, each with a weight of $\frac{1}{k}$. Assigning a weight of $\frac{1}{k}$ to each included stock leads to good tracking portfolios according to (Fiterman and Timkovsky, 2001), who argue that a stock included in the portfolio should have a relative weight that is proportional to its relative weight in the original index, i.e., to $\frac{1}{N}$. Finally, we must invest the whole budget, and short selling is not possible. We obtain the following mixed-integer quadratic program:

$$\text{MIQP} \begin{cases} \text{Min.} \sum_{i=1}^{N} \sum_{j=1}^{N} \sigma_{ij}(x_i - \frac{1}{N})(x_j - \frac{1}{N}) & (2) \\ \text{s.t.} \sum_{i=1}^{N} x_i = 1 & (3) \\ \sum_{i=1}^{N} z_i \leq k & (4) \\ \varepsilon_i z_i \leq x_i \leq \delta_i z_i \quad (i = 1, \ldots, N) & (5) \\ x_i \geq 0, z_i \in \{0,1\} \; (i = 1, \ldots, N) & (6) \end{cases}$$

The objective function (2) corresponds to the VRR between tracking and $1/N$ portfolio. Constraint (3) ensures that the whole budget is invested, and constraint (4) limits the total number of stocks to include in the tracking portfolio to k. The binary decision variables z_i are used to formulate the cardinality constraint (4). Constraints (5) both guarantee that the binary variables z_i are equal to one if and only if the portfolio weight x_i is greater than zero, and that the

portfolio weights x_i are within the range $[\varepsilon_i, \delta_i]$ if and only if $z_i = 1$. Finally, constraints (6) specify the domains of the decision variables z_i and x_i.

3 RELATED LITERATURE

To the best of our knowledge, there is no specific literature on the planning problem described in Section 2. However, since we can interpret the $1/N$ portfolio as a special index with equal weights for each stock, the returns of the $1/N$ portfolio could be replicated by applying index-tracking methods. Index-tracking methods are applied to replicate the returns of large market indices with a small set of stocks to save transaction costs. In Subsection 3.1, we give an overview on the literature on index tracking. In Subsection 3.2, we summarize the literature on iterated greedy heuristics.

3.1 Index Tracking

Index-tracking methods are applied to solve the index-tracking problem, which consists of constructing a portfolio that best-possibly replicates the index returns by investing in a small subset of some set of stocks. With the exception of (Roll, 1992), most approaches to the index-tracking problem assume that the true index weights are unknown. For example, (Beasley et al., 2003) develop an evolutionary heuristic to the index-tracking problem. They minimize some function of the differences between the portfolio and index returns and consider various practical portfolio constraints such as a maximum number of stocks to include in the portfolio, minimum and maximum weights for each included stock, and a budget for transaction costs. (Canakgoz and Beasley, 2008) develop a regression-based approach for the index-tracking problem. Their objective is to construct a portfolio that has an intercept of zero and a slope of one when regressing the portfolio returns on the index returns. They consider similar practical portfolio constraints as (Beasley et al., 2003). (Guastaroba and Speranza, 2012) present a mixed-integer linear program as well as a MIP-based heuristic called Kernel Search for the index-tracking problem. They consider similar practical portfolio constraints as (Canakgoz and Beasley, 2008), but additionally consider fixed transaction costs. By assuming the true index weights to be unknown, these approaches have the advantage that they can be applied to track any index by selecting a subset of stocks from any set of stocks, even if the index is not composed of this set of stocks.

3.2 Iterated Greedy Heuristics

Iterated greedy heuristics start by constructing an initial solution. This solution is then improved during an improvement phase that consists of the two sub-phases deconstruction and construction (Ruiz and Stützle, 2007). During the deconstruction sub-phase, some elements of the current candidate solution are removed. In the construction sub-phase, a new candidate solution is constructed by adding elements in a greedy way back to the deconstructed solution. The new candidate solution is then either accepted or discarded based on a specific acceptance criterion. To escape local minima, the acceptance criterion is designed such that also worse solutions are accepted with some probability. By repeating the deconstruction and construction sub-phases, iterated greedy heuristics overcome the drawbacks of a simple greedy heuristic such as those mentioned in (Gutin et al., 2002).

Iterated greedy heuristics are simple to understand and easy to implement in practice, yet very effective in providing high quality solutions for various problems. For example, (Jacobs and Brusco, 1995) apply IGH to the set covering problem. (Ruiz and Stützle, 2007) and (Ruiz and Stützle, 2008) demonstrate the effectiveness of iterated greedy heuristics for variants of the flowshop problem. IGH is also applied to the unrelated parallel machine scheduling problem (Fanjul-Peyro and Ruiz, 2010). However, to the best of our knowledge, iterated greedy heuristics have not been applied to portfolio-optimization problems.

4 ITERATED GREEDY HEURISTIC

In this section, we present our iterated greedy heuristic, which is based on an IGH developed for the permutation flowshop scheduling problem (Ruiz and Stützle, 2007). In Subsection 4.1, we give an overview on the design of our IGH. In Subsection 4.2, we present the greedy insertion heuristic that is used as a subroutine in our IGH.

4.1 Overview

For our iterated greedy heuristic to track the $1/N$ portfolio, we made two simplifications: (1) We always include as many stocks as possible ($= k$) in the tracking portfolios, and (2) each stock in the portfolio has the same weight ($= \frac{1}{k}$). These simplifications allow us to save the CPU time required for determining the portfolio weights of each stock, e.g., by solving a

quadratic program. We assume that for all instances $\varepsilon_i \leq \frac{1}{k} \leq \delta_i$, $i = 1, \ldots, N$, i.e., a portfolio weight of $\frac{1}{k}$ for every stock is feasible (cf. Section 2).

Our iterated greedy heuristic proceeds as follows (cf. Algorithm 1): We start with an empty portfolio π, where all stocks have a portfolio weight of zero. We then greedily add k stocks with a weight of $\frac{1}{k}$ to π by applying the greedy insertion heuristic discussed in Subsection 4.2 such that the objective function value by adding one new stock declines the most. The best known solution is stored in π_b and will be outputted at the end of Algorithm 1.

Algorithm 1: Iterated greedy heuristic (IGH).

```
π := ∅;
for i := 1 to k do
    Add best stock to portfolio π with weight 1/k by applying Algorithm 2
π_b := π;
while termination criterion not satisfied do {Improvement phase}
    π' := π
    p := ⌊random * d⌋ + 1
    for i := 1 to p do {Deconstruction sub-phase}
        remove a randomly selected stock from portfolio π'
    for i := 1 to p do {Construction sub-phase}
        Add best stock to portfolio π' with weight 1/k by applying Algorithm 2
    if VRR(π') < VRR(π) then
        π := π';
        if VRR(π) < VRR(π_b) then
            π_b := π;
    else
        if random ≤ e^(-(VRR(π')-VRR(π))/temp) then
            π := π';
return π_b;
```

After having found an initial solution, Algorithm 1 enters the improvement phase and tries to improve π_b until some termination criterion is met, e.g., until the limit on total CPU time is reached. From the current solution, p random stocks are removed during the deconstruction sub-phase, where p is calculated as $\lfloor random * d \rfloor + 1$. The parameter d denotes the maximum number of stocks to remove. Because *random* is a uniformly distributed random number in the interval $[0,1)$, p is a uniform random integer from the set $\{1, \ldots, d\}$. During the construction sub-phase, p stocks are added back to the portfolio with a weight of $\frac{1}{k}$. These p stocks are selected greedily by applying our greedy insertion heuristic (cf. Subsection 4.2). The new portfolio π' is then compared to the old portfolio π before removing and adding back p stocks. If the solution π' is better than π, the new portfolio is immediately accepted as new solution. Algorithm 1 then also checks if the new portfolio improves the best known solution π_b. If this is the case, π_b is updated. With some probability, the solution π' is accepted as new solution even if it is worse than π. Due to the greedy nature of our heuristic, Algorithm 1 could be trapped in a local minimum if only better solutions were accepted (Johnson et al., 1989). Therefore, worse solutions are accepted with a small probability to enable Algorithm 1 to escape such local minima. The probability to accept worse solutions depends on the parameter *temp*, which can be interpreted as the temperature used in simulated annealing heuristics (Johnson et al., 1989; Johnson et al., 1991). To compare the objective function values $(VRR(\pi), VRR(\pi'), VRR(\pi_b))$ of the solutions, Algorithm 1 does not actually evaluate the objective function, but uses the information about the possible improvement in the objective function value for adding a given stock. This information is generated by Algorithm 2, which is presented in the following subsection.

4.2 Greedy Insertion Heuristic

In this subsection, we explain the greedy insertion heuristic that is used both to add back stocks to the portfolio during the construction sub-phase and to construct an initial solution at the beginning of the IGH. A naive approach to find the best stock to add to a current portfolio would proceed as follows: For each stock that has not been selected yet, a portfolio is constructed by adding the stock to the current portfolio with a weight of $\frac{1}{k}$. Then, the objective function value is computed by using (2) for each of these portfolios. The stock that leads to the portfolio with the lowest objective function value is then selected. To calculate the objective function value for a given portfolio, N^2 times two subtractions and three multiplications need to be performed, i.e., in total $5N^2$ floating point operations (flops). If the current portfolio includes m stocks, this approach needs $5N^2(N-m) = 5N^3 - 5N^2m$ flops to select the best stock.

However, by applying Algorithm 2, the best stock can be found in a more efficient way, i.e., in $2N^2 - 2Nm$ flops. To see this, assume that we want to calculate the possible improvement in the objective function value if we increase the weight of some stock j' that has not been selected yet from 0 to $\frac{1}{k}$. The only summands in the objective function affected by this

change are:

$$\begin{aligned} &\sigma_{j'j'}(x_{j'}-\frac{1}{N})^2+ \\ &\sum_{i\in\{1,\ldots,N\}\setminus j'}[\sigma_{ij'}(x_i-\frac{1}{N})(x_{j'}-\frac{1}{N})]+ \\ &\sum_{i\in\{1,\ldots,N\}\setminus j'}[\sigma_{j'i}(x_{j'}-\frac{1}{N})(x_i-\frac{1}{N})] \qquad (7) \\ &\equiv \sigma_{j'j'}(x_{j'}-\frac{1}{N})^2+ \\ &2\sum_{i\in\{1,\ldots,N\}\setminus j'}[\sigma_{ij'}(x_i-\frac{1}{N})(x_{j'}-\frac{1}{N})] \end{aligned}$$

Because the covariance matrix is symmetric, i.e., $\sigma_{ij}=\sigma_{ji}$, $i,j\in\{1,\ldots,N\}$, the two sums in (7) are equal. If we add stock j' to the portfolio, there are three different cases how the summands in (7) are affected.

1. Consider some stock $i\neq j'$ that has not been selected yet. The contribution to (7) of the two stocks i and j' before adding stock j' to the portfolio is $2\sigma_{ij'}(0-\frac{1}{N})(0-\frac{1}{N})$. After changing the weight of stock j' from 0 to $\frac{1}{k}$, this contribution changes to $2\sigma_{ij'}(0-\frac{1}{N})(\frac{1}{k}-\frac{1}{N})$. So, the improvement can be calculated as the difference between these two contributions and corresponds to $\frac{2\sigma_{ij'}}{Nk}$. In Algorithm 2, we use $w_1=\frac{2}{Nk}$ as factor to weight the covariances that belong to this first case to compute the possible improvement for stock j'.
2. Consider some other stock $i\neq j'$ that has already been selected and thus, has a weight of $\frac{1}{k}$ in the portfolio. Therefore, the contribution of the stocks i and j' to the objective function value before changing the weight of stock j' is $2\sigma_{ij'}(\frac{1}{k}-\frac{1}{N})(0-\frac{1}{N})$, and becomes $2\sigma_{ij'}(\frac{1}{k}-\frac{1}{N})^2$ after changing the weight of stock j'. So, the difference is $2\sigma_{ij'}(\frac{1}{kN}-\frac{1}{k^2})$. In Algorithm 2, we use $w_2=w_1-\frac{2}{k^2}$ to weight the covariances for this second case.
3. The variance of the returns of stock j' contributes as follows to the objective function value before changing its weight: $\sigma_{j'j'}(0-\frac{1}{N})^2$. After adding stock j' to the portfolio, the contribution transforms to $\sigma_{j'j'}(\frac{1}{k}-\frac{1}{N})^2$. So, the difference is $\sigma_{j'j'}(\frac{2}{kN}-\frac{1}{k^2})$. For this third case, we use the weight $w_3=w_1-\frac{1}{k^2}$ in Algorithm 2.

In total, to compute the possible improvement for increasing the weight of stock j' from 0 to $\frac{1}{k}$, we can use Algorithm 2 that multiplies the covariances between the returns of stock j' and stocks $i\in\{1,\ldots,N\}$ with the respective weights w_1, w_2, or w_3. The weighted covariances are then added up to calculate the total improvement. So, in total, this takes N multiplications and N additions (ignoring the flops to calculate the weights w_1,w_2,w_3). Therefore, to select the best stock from the $N-m$ stocks that have not been selected yet, i.e., the stock that leads to the largest decline ω^* in the objective function value, Algorithm 2 takes $2N^2-2Nm$ flops.

Algorithm 2: Greedy insertion heuristic.

```
π' := current portfolio (input);
w1 := 2/(Nk);
w2 := w1 − 2/k^2;
w3 := w1 − 1/k^2;
ω* := −∞
for j := 1 to N do
    if stock j has a weight of 0 in π' then
        ω := 0
        for i := 1 to N do
            if i = j then
                ω := ω + w3 σij;
            else
                if stock i has a weight of 0 in π'
                then
                    ω := ω + w1 σij;
                else
                    ω := ω + w2 σij;
        if ω > ω* then
            ω* := ω
            j* := j
return j* and ω*;
```

5 COMPUTATIONAL EXPERIMENT

We tested our iterated greedy heuristic on a set of test instances generated from real-world stock-market data, and compared the best results obtained by the heuristic to those obtained by the MIQP. In Subsection 5.1, we present the test instances. In Subsection 5.2, we explain the test setting. In Subsection 5.3, we report on the computational results.

5.1 Test Instances

In total, we used 23 test instances for our computational experiment (cf. Table 2). We generated the first eight of these instances from index-tracking benchmark instances that can be downloaded from the OR-Library (Beasley, 1990). We constructed 15 further

Table 2: Test instances.

Nr.	Index	Number of stocks N	k
1	Hang Seng	31	10
2	DAX100	85	10
3	FTSE100	89	10
4	S&P100	98	10
5	Nikkei225	225	10
6	S&P500	457	40
7	Russell2000	1,319	70
8	Russell3000	2,152	90
9	SMI	20	10
10	Hang Seng	49	10
11	EUROSTOXX50	50	10
12	FTSE100	96	10
13	S&P100	99	10
14	NASDAQ100	101	10
15	DAX100	102	10
16	SPI	198	10
17	Nikkei225	220	10
18	S&PASX300	254	10
19	S&P500	489	40
20	FTSE All Share	567	40
21	STOXXEURO600	575	40
22	S&P1200	1,179	70
23	Nasdaq Composite	2,140	90

test instances from real-world stock-market indices in the same way as (Beasley, 1990). Each of the 23 instances represents a specific investment universe composed of N stocks, and consists of the weekly prices of all the stocks that constitute the given market index and do not have any missing prices. The $1/N$ portfolio is then composed of each of these N stocks with a weight of $\frac{1}{N}$ for each stock, and can be seen as an artificial index that is equally weighted and rebalanced every week. The objective is to replicate the returns of this artificial index.

5.2 Test Setting

For our computational experiments, we defined the specific values for the different parameters from Table 1 as follows. The number of stocks in the investment universe (N) depends on the specific test instance and is depicted in Table 2. This table also shows the maximum number of stocks to include in a tracking portfolio (k) for each instance. For the first eight instances, we used the same values for k as (Guastaroba and Speranza, 2012), whose objective was to track the original market index. For the remaining 15 instances, the corresponding k of the first eight instances with a similar size N was used. As lower and upper limits on the portfolio weights for each stock included in the tracking portfolio, we used a minimum weight of 0.5% ($\varepsilon_i = 0.005$, $i = 1,\ldots,N$) and a maximum weight of 20% ($\delta_i = 0.2$, $i = 1,\ldots,N$) for each instance. These values allowed us to construct tracking portfolios composed of k stocks with a weight of $\frac{1}{k}$ for all instances. The covariances between the returns of the stocks σ_{ij} $(i, j = 1,\ldots,N)$ were computed based on the historical stock returns over a time window of 104 weeks. Since the resulting MIQP is only convex if the matrix of covariances is positive semi-definite, we use the covariance estimator developed by (Ledoit and Wolf, 2004) that always yields a positive semi-definite covariance matrix (even if a test instance contains more stocks than weekly prices). The last parameter in Table 1 is the weight of each stock in the artificial index we want to replicate (w_i, $i = 1,\ldots,N$). Because our target index corresponds to the $1/N$ portfolio, we used $\frac{1}{N}$ for this parameter.

Furthermore, we had to define the parameters d and *temp* that are relevant for the iterated greedy heuristic only. For the maximum number of stocks d to remove and add during the deconstruction and the construction sub-phase, we tested the four values $2,3,4,5$ for all instances. For the largest instances with $N > 500$ and $k \geq 40$, we also tested the values $d = 10$ and $d = 20$. For the parameter *temp*, which defines the acceptance criterion for new solutions, we used a constant value of 0.5 for all instances. This means that we tested four and six different variants of the iterated greedy heuristic for the smaller and the larger test instances, respectively.

For the MIQP and the IGH, we limited the CPU time to 300 and 100 seconds per instance, respectively. For IGH, the 100 seconds CPU time limit were used as only termination criterion, which means that the deconstruction and construction sub-phases were executed repeatedly until this time limit was reached. We used the Gurobi solver 6.0 to solve the MIQP. The iterated greedy heuristic was implemented in C (compiler: gcc 4.9.3). All computations were performed on a standard PC with an Intel i7 CPU with 3.4 GHz and 4 GB RAM.

5.3 Results

Table 3 presents the computational results for the 23 test instances sorted by the number of stocks N in the instance. Columns three and four show the best objective function values and the best lower bounds on the objective function value for the MIQP (scaled by a factor of 10^6). Columns five to ten report the results obtained by the variants of our iterated greedy heuristic. The figures in these columns represent the relative

Table 3: Computational results.

Instance		MIQP		IGH with $temp = 0.5$					
N	Nr.	Best obj.	Best bound	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 10$	$d = 20$
20	9	9.84	9.84	25.4%	25.4%	25.4%	25.4%	–	–
31	1	32.92	32.92	18.3%	16.1%	16.1%	18.3%	–	–
49	10	19.45	12.13	17.0%	17.0%	17.0%	17.0%	–	–
50	11	18.06	7.86	4.1%	4.1%	4.1%	4.1%	–	–
85	2	42.04	3.85	−2.1%	−2.1%	−2.1%	−2.1%	–	–
89	3	47.57	3.39	1.1%	−1.0%	−1.0%	−1.0%	–	–
96	12	20.99	1.29	8.2%	5.5%	5.5%	5.5%	–	–
98	4	58.80	2.46	2.9%	0.5%	0.5%	0.5%	–	–
99	13	16.45	1.08	4.5%	4.5%	4.5%	4.5%	–	–
101	14	44.91	2.73	6.3%	−1.2%	−1.2%	−1.2%	–	–
102	15	37.39	1.96	3.0%	0.9%	0.9%	0.9%	–	–
198	16	147.85	0.84	0.0%	0.0%	0.0%	0.0%	–	–
220	17	31.08	0.06	2.3%	0.2%	0.2%	−0.3%	–	–
225	5	29.85	0.05	23.9%	20.6%	19.3%	20.0%	–	–
254	18	116.18	0.27	3.2%	−2.8%	−3.6%	−3.6%	–	–
457	6	14.74	0.03	−9.7%	−9.3%	−11.4%	−12.2%	–	–
489	19	4.98	0.01	−11.8%	−14.7%	−13.9%	−14.1%	–	–
567	20	10.94	0.01	−12.7%	−14.8%	−15.1%	−14.5%	−14.2%	−12.3%
575	21	8.19	0.01	−7.5%	−11.1%	−9.2%	−12.5%	−6.7%	−2.5%
1179	22	5.39	0.00	−30.5%	−32.2%	−31.7%	−31.2%	−29.7%	−26.9%
1319	7	37.75	0.04	−25.2%	−26.6%	−26.4%	−26.7%	−26.6%	−26.9%
2140	23	44.48	0.01	−22.3%	−22.3%	−22.3%	−22.4%	−22.3%	−22.3%
2152	8	103.64	0.01	−84.4%	−84.5%	−84.4%	−84.6%	−84.4%	−84.3%

differences between the best objective function values obtained by the iterated greedy heuristics ($VRR(\pi_b)$ from Algorithm 1) and the MIQP (referred to as ofv MIQP) within the CPU time limits. These relative differences are computed as $\frac{VRR(\pi_b)-\text{ofv MIQP}}{\text{ofv MIQP}}$, and are negative if IGH obtained better (smaller objective function value) solutions than the MIQP within the given CPU times.

For the smaller instances, the Gurobi solver obtained better solutions than the iterated greedy heuristics and could also prove optimality of the solutions for the two smallest instances. However, with respect to the average over all instances, and in particular for the larger problem instances, the iterated greedy heuristic obtained portfolios with a considerably lower objective function value. Over all 23 instances, the variants of the iterated greedy heuristic obtained solutions whose objective function values were roughly 5% lower than those obtained by the MIQP. For the larger instances with $N > 500$ the average relative difference was approximately -35%. Furthermore, Table 3 also indicates that the greedy heuristics delivered good results independently of the choice for the parameter d.

6 CONCLUSIONS

In this paper, we considered the problem of replicating the $1/N$ portfolio by investing only in a subset of the N stocks from a given investment universe. To find solutions to this problem, we presented a mixed-integer quadratic program (MIQP) and developed an iterated greedy heuristic. In a computational experiment, it turned out that the iterated greedy heuristic was able to obtain better solutions for the larger instances in shorter CPU time.

In future research, we will evaluate our approach on a larger set of test instances, and compare the results of our approach with standard index-tracking approaches that ignore the actual index weights. Further, we will analyze the impact of the minimum and maximum weights of each stock included in the tracking portfolio as well as the maximum number of stocks to invest in. Also, a local search method should be developed that can be applied to improve the solutions in each iteration of the heuristic. Furthermore, our IGH approach could be extended by relaxing the two simplifications that exactly k stocks with an equal weight have to be included in a tracking portfolio; for this purpose, we should apply quadratic pro-

gramming to determine optimal portfolio weights for each included stock. According to the result of (Balas and Saltzman, 1991) that max-regret greedy heuristics outperform simple greedy heuristics, it could also be interesting to develop an iterated max-regret greedy heuristic for the problem considered in this paper.

REFERENCES

Balas, E. and Saltzman, M. J. (1991). An algorithm for the three-index assignment problem. *Operations Research*, 39:150–161.

Beasley, J. E. (1990). OR-Library: Distributing test problems by electronic mail. *The Journal of the Operational Research Society*, 41:1069–1072.

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

Canakgoz, N. A. and Beasley, J. E. (2008). Mixed-integer programming approaches for index tracking and enhanced indexation. *European Journal of Operational Research*, 196:384–399.

DeMiguel, V., Garlappi, L., and Uppal, R. (2009). Optimal versus naive diversification: How inefficient is the 1/N portfolio strategy? *The Review of Financial Studies*, 22:1915–1953.

Fanjul-Peyro, L. and Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, 207:55–69.

Fiterman, A. E. and Timkovsky, V. G. (2001). Basket problems in margin calculation: Modelling and algorithms. *European Journal of Operational Research*, 129:209–223.

Guastaroba, G. and Speranza, M. G. (2012). Kernel Search: An application to the index tracking problem. *European Journal of Operational Research*, 217:54–68.

Gutin, G., Yeo, A., and Zverovich, A. (2002). Traveling salesman should not be greedy: Domination analysis of greedy-type heuristics for the TSP. *Discrete Applied Mathematics*, 117:81–86.

Jacobs, L. W. and Brusco, M. J. (1995). A local search heuristic for large set-covering problems. *Naval Research Logistics Quarterly*, 42:1129–1140.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C. (1989). Optimization by simulated annealing: An experimental evaluation; Part I, graph partitioning. *Operations Research*, 37:865–892.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C. (1991). Optimization by simulated annealing: An experimental evaluation; Part II, graph coloring and number partitioning. *Operations Research*, 39:378–406.

Ledoit, O. and Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88:365–411.

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

Roll, R. (1992). A mean/variance analysis of tracking error. *The Journal of Portfolio Management*, 18:13–22.

Ruiz, R. and Stützle, T. (2007). A simple an effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177:2033–2049.

Ruiz, R. and Stützle, T. (2008). An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 187:1143–1159.