2018 INFORMS O.R. & Analytics Student Team Competition

Entry Number: [2018ORASTC252]

Executive Summary

This report proposes a novel optimization framework for the portfolio optimization problem, which determines an optimal portfolio allocation across many assets. Since the seminal work of Markowitz (**?**), the so-called mean-variance model has become the pinnacle of modern financial decision making. The idea behind the mean-variance model is to investigate an optimal trade-off between risk and return, which often can be formulated as a convex optimization problem. The computational merit of classical mean-variance model is one of major reason of the popularity of the model.

In practice, however, the classical Markowitz model lacks of many desired features. Recently, many variants of Markowitz model were introduced to make the models more realistic. Unfortunately, most of them are not easy to solve unlike Markowitz model due to non-convexity. Moreover, with advance of globalization, the number of assets to consider is tremendously increasing, which is making the problem even harder to solve. Traditional approach for solving the portfolio optimization problems is based on mathematical programming. The modeling power of mixed integer programming is so powerful, most of the portfolio problems can be easily formulated which can be fed to off-the-shelf MIP solvers. The resulting formulations, however, are very hard to solve due to non-convexity and/or existence of integer decision variables. For this reason, extensive research works have been devoted for developing heuristic methods (**??**).

The portfolio optimization problem concerned in this report is one of the most general forms, motivated by a real-life investment company. It is distinguished from the classical Markowitz model in several ways:

- The risk and return of a portfolio are measured against *benchmark* weights.
- The number of assets with meaningful weights (e.g., greater than 0.001) must be limited by a given range.
- The model provides parameters and constraints that can be used for controlling activeness and/or volatility of portfolio compared to the benchmark index.
- The formulation for the problem is non-convex and involves integer decision variables, which makes the problem computationally intractable.

The following sections will present detailed analysis on the problem and data.

For the last several years, we witnessed beginning of artificial intelligence (AI) era, ignited by deep neural networks (DNNs, **?**). Neural networks (NNs) are widely used in financial applications such as prediction and detecting of financial crisis. There are, however, little previous works that utilize the NNs for solving portfolio optimization problems. Common wisdom is the NNs are very good in classification and approximation of functions, but not adequate in solving optimization problems with many complicated constraints. In this report, we present a novel optimization approach by taking bests of two approaches: the NNs and the mathematical programming. We see that the contributions of our work as:

- We develop a three-step heuristic algorithm for solving the portfolio optimization problem.
- The first step of the algorithm utilizes a variant of DNN, while the remaining steps involve solving of convex optimization problems.
- The algorithm is well-scaled horizontally by employing more computing power.
- Extensive computational experiments that show efficiency of our approach.

In particular, the proposed approach enables us to take into account uncertainty of important parameters. We also present how our algorithm can be extended to deal with worst-case realization by using robust optimization (**?**).

Team Makeup & Process

Our team consists of three undergraduate students majoring industrial and management engineering. Our advisor is an assistant professor in our department whose research field is Operation Research. We three worked together on a variety of projects, including data mining projects and optimization modeling projects during the classes in college. The three of us enjoy challenging very much, and when we heard about the INFORMS competition from the advisor, we immediately decided to give a try. We expected that this would be a fantastic opportunity to learn about a real-life optimization modeling and we now can say that the expectation is met even more satisfactory. After building of the team, we have been doing the project through the weekly meeting. Every week, we discussed and studied anything relevant to the project, for example:

- Acquire the background knowledge of the financial field to understand the problem.
- Analyze existing research to identify trends in the latest portfolio optimization.
- Discuss the direction of the methodology after implementing the algorithm using well-known methods

- Various attempts to develop good algorithm

One of us has patented an algorithm to solve the Chinese Postman Problem, and applied it to the problems of the real world and developed the mobile application and started the business. Because of his experience in developing algorithms, his works were focused on solving the portfolio optimization through the formulation based with various optimization tools. Another team member has experience in refining and analyzing real data from various companies, and is especially interested in deep learning. This team member's background could be the idea for the project and develop a deep learning based algorithm which makes our algorithm distinguished. Our algorithm includes various technical contents such as parallel processing and data processing. Our last team member contributes to make our heuristic algorithm faster and more powerful with his information system development experience. The advisor gives guidance on the algorithm presented through the weekly meetings, and his mentorship leads us to the right direction. Above all, the teamwork is the most important factor on developing the right algorithms by maximizing each experience and background. So, it can be said that we solved another optimization problem that has ourselves as decision variables.

From understanding the problem to developing the algorithm, all process were a series of challenges. Our first challenge was to understand and analyze the business model correctly. The actual data given are very different from what is commonly used in the classroom, but we tried to solve it by analyzing actual stock data as well as related papers through weekly seminars. Our second major challenge was to develop a methodology that fits the problem requirements through a deep understanding of the problem. In particular, the number of stocks to be considered may be large in the real world, so we devised a method of separating the process of selecting the set of assets to invest and determining the weights of the assets so as not to be sensitive according to the problem size.

We keep challenging ourselves, and we grow up through trial and error during four months. As a result, we believe that we have developed an algorithm that is unique and practically usable.

Framing the Problem

Portfolio optimization is finding the optimal proportion of each asset in an investors portfolio. In the problem given, the investors construct an investment portfolio to minimize risk and, at the same time, to maximize the expected return. The investment portfolio is also required to satisfy not only the well-known Markowitz formulation constraints, but also the extension constraints that reflect the real-world requirements. First, we take a look at the meaning of each constraints in the formulation presented and give a suggestion on how to handle the constraints. The parameters and decision variables are defined as follows:

N : set of assets

w_i^{bench} : weight of benchmark of asset $i \in N$

$\Omega \in \mathcal{R}^{|N|\times|N|}$: return covariance matrix for all assets in the universe, which contains the relationships between each pair of historical return series. Ω is a symmetric matrix where the diagonal elements are the variances of the asset returns, the off-diagonal elements are covariances.

α_i : Expected return score of asset $i \in N$, higher values are expected to have higher future returns.

β_i : measure of return sensitivity relative to the equity market return as a whole of asset $i \in N$

S : set of sectors

N_s : set of assets of sector $s \in S$

K : set of MCAPQ (Market Cap Quintile)

N_k : set of assets of MCAPQ $k \in K$

w_i : decision variable, weight of asset $i \in N$

d_i : decision variable, difference between weight and the benchmark of asset $i \in N$

For a given asset $i \in N$, d_i is defined as

$$d_i = w_i - w_i^{bench}. \tag{1}$$

Then, the mathematical formulation for the portfolio selection problem is stated as:

$$\begin{aligned}
\text{(P)} \quad \min\ & d^T\Omega d - \lambda d^T\alpha && && (2)\\
\text{s.t. } & w_i \geq 0, && \forall i \in N, && (3)\\
& \sum_{i\in N} w_i = 1, && && (4)\\
& -0.05 \leq d_i \leq 0.05, && \forall i \in N, && (5)\\
& -0.1 \leq \sum_{i\in N_s} d_i \leq 0.1, && \forall s \in S, && (6)\\
& -0.1 \leq \sum_{i\in N_k} d_i \leq 0.1, && \forall k \in K, && (7)\\
& -0.1 \leq \sum_{i\in N} \beta_i d_i \leq 0.1, && && (8)\\
& 50 \leq card(w_i \neq 0) \leq 70, && && (9)\\
& 0.6 \leq 1 - \sum_{i\in N} \min\{w_i, w_i^{bench}\} \leq 1, && && (10)\\
& 0.05 \leq \sqrt{d^T\Omega d} \leq 0.1, && && (11)
\end{aligned}$$

where $card(w_i \neq 0)$ represents the number of weights that are greater than a given threshold 0.001. The objective function (2) minimizes the sum of total risk ($d^T \Omega d$) and the negative of expected return ($d^T \alpha$), weighted by parameter λ that controls the trade-off between the risk and expected return. Constraints (3) - (8) are linear, each of which represents a specific business requirement. Constraints (5) ensures that $d_i \in [-0.05, 0.05]$ for any $i \in N$. When the lower bound of the weight is bigger than 0.001, it should be included in the set of assets to invest. For each set of sector $s \in S$ and MCAPQ $k \in K$, the sum of the active share must be in the range of $[-0.1, 0.1]$. Not only the number of assets included in each sector and MCAPQ is different, but also the sum of bench weights is different. Constraint (9) restricts the number of positive weights, which makes the problem difficult because of the non-convex function $card(w_i \neq 0)$. Nonetheless, constraint (9) can be linearized by introducing some binary decision variables as shown in the following proposition.

Proposition 1 *Constraint* (9) *can be replaced by the following set of constraints:*

$$
\begin{aligned}
& y_i \geq w_i, && \forall i \in N, \\
& y_i \leq w_i + 0.999, && \forall i \in N, \\
& 50 \leq \sum_{i \in N} y_i \leq 70, && \\
& y_i \in \{0, 1\}, && \forall i \in N.
\end{aligned}
$$

Constraint (10) restricts the active share of the portfolio between 0.6 and 1. Because $w_i \geq 0$ and $w_i^{bench} \geq 0$ for any $i \in N$, the constraint $1 - \sum_{i \in N} \min\{w_i, w_i^{bench}\} \leq 1$ is redundant, which results in the following proposition.

Proposition 2 *Constraint* (10) *can be replaced by the following set of constraints:*

$$
\begin{aligned}
& 0.6 \leq 1 - \sum_{i \in N} v_i, && \\
& w_i - z_i \leq v_i \leq w_i, && \forall i \in N, \\
& w_i^{bench} - 1 + z_i \leq v_i \leq w_i^{bench}, && \forall i \in N, \\
& z_i \in \{0, 1\}, && \forall i \in N.
\end{aligned}
$$

Constraint (11) ensures the tracking error (TE) to be in the given range $[0.05, 0.1]$. In general, a higher TE means the portfolio would be more volatile, while a lower TE indicates the portfolio would closely follow the benchmarks. Constraint (11) consists of two inequalities: $0.05 \leq \sqrt{d^T \Omega d}$ and $\sqrt{d^T \Omega d} \leq 0.1$, where the latter is a convex constraint. The former inequality, however, is non-convex. Tracking error and active share are measures of how different the investment portfolio is from the benchmark, a composite indicator derived from several accumulated criteria. Assume two distinct investment portfolios, the absolute difference between portfolio weights and bench weights are the same, and consider the case where portfolio weight is biased toward one or two assets. Since

the two have the same absolute value of the active weight, the active share would be the same. Likewise, active share ignores diversification, so it is necessary to see tracking error considering not only benchmark weight but also risk between assets. In the problem, tracking error is used as an indicator of risk in objective function to be minimized. Therefore, the optimal solution is the value that the tracking error converges to the lower bound value while the value of return is high.

By using Proposition 1 and 2, we can formulate a mixed integer non-linear programming (MINLP) problem. There are several solvers for obtaining global optimal solutions of the MINLP problems such as BARON (**?**), Couenne (**?**), and Bonmin (**?**). Those solvers typically incorporate outer approximation/generalized Bender's decomposition into branch-and-bound framework. Existence of many binary decision variables and non-convex constraints, however, makes solving of the formulation directly using the off-the-shelf solvers intractable.

Data

The data from Principal consists of 3 parts which are timeseries data, riskmodels data, and result template data. Timeseries data and riskmodels data contain the parameters for portfolio optimization, and result template data contains indicators for evaluating solution quality after deriving the final solutions. The details of each data for the portfolio optimization problem are listed as follows.

- **Time Series Data** : It contains the parameter values needed to construct the formulation for each period. At this time, not only does it contain unique information of assets like SEDOL or Sector that do not change, but it also includes Alpha score (α), Beta (β), Bench Weight (w_{bench}), and MCAPQ, which describe the characteristics of the assets at each period.
- **Riskmodels Data** : The risk is composed of diagonal elements representing the risk of each asset and off-diagonal elements representing the risk between the two assets. It is important that not only the variances of the return values of the diagonal elements are minimized, but also the large assets of the omega between the two assets are not selected together.
- **Result Template Data** : It contains 4 weekly forward returns, and is used when calculating portfolio performance.

1 Data Pre-processing and Rescailing

There are some difficulties in applying the data given from Principal directly to the model. Therefore, we present the following data preprocessing process. To make the data composed of three parts more flexible, a process of data preprocessing formed dataframe using the Pandas Library (`http://pandas.pydata.org`) in Python. In particular, as rebalancing is carried out, it is configured to extract not all time series data, but only the data needed for the iteration. The parameters

Correlations	W_{bench}	α	β	Variance	Covariance
W_{bench}	1	0.04	-0.17	-0.04	-0.08
α		1	-0.07	0.03	-0.03
β			1	0.22	0.5
Variance				1	-0.16
Covariance					1

Figure 1: The Correlation of Parameters

used in model were extracted from data frames formed for each period, when each parameter (Alpha Score(α), Beta(β),4 Weekly Returns (r), etc) was set with the 'SEDOL' index as the key value. This dictionary data type is proper to consider a list of assets that may change every period. The parameter Omega (Ω) is specified as the covariance matrix for a given data. In this case, the index in the columns and rows are 'SEDOL', the key value of the time series dictionary derived from the above, and are constructed in the form of full matrix. Taking the first period data as an example, we realize that the range of Ω is (from -0.024 to 0.616) and the range of α is (from -2.1e-05 to 1.9e-05). Especially, Ω is less influential because it is considered with a very small value of active weight. Because the value of data is small, it may cause a calculation error due to the limit value of Python, and the value may be ignored in optimization tool, CPLEX. Therefore, we prevent the possible round-off errors by multiplying α and Ω by 10000. By multiplying α and Ω to 10000, we multiply both sides of the constraints (11) equally by 10000 after squaring both sides.

2 The Analysis of the Data

We analyze the correlation between each parameters given to figure out how much each parameter impacts on others. The parameters we take a look are w_{bench}, α, and β from the time series data and Ω from the riskmodel data. We obtain the covariance matrix which contains diagonal and off-diagonal elements, according to the problem definition, each of the data is variances of the asset returns and covariances, so we see the value of the correlation coefficient,respectively. Figure 1 shows the correlation coefficient between the parameters of w_{bench}, α, β, $Variance$, and $Covariance$. Overall, there is no clear correlation between most parameters. Especially, the w_{bench} which contains company's know-how for a long time and α are not correlated with all other parameters. One interesting fact is that there is no correlation between diagonal values of $\Omega(Variance)$ and off-diagonal elements ($Covariance$). For a number of assets, they are set to 0 for the off-diagonal and 0.5 for the diagonal elements, which means that the risk of the asset itself is high, but the covariance between other assets is low. These values have a large effect on the correlation coefficient.

Table 1: The results of solving (P1) by CPLEX

B&B time (sec)	# of feasible	GAP(%)	Tracking Error	Active Share	Violated Constraints
30	0	-	-	-	-
60	0	-	-	-	-
90	2	100.3%	0.0018	0.72	(11) TE
120	2	100.3%	0.0012	0.64	(11) TE
150	2	100.3%	0.0018	0.72	(11) TE
180	8	100.6%	0.0004	0.50	(10) AS, (11) TE

Methodology Approach & Model Building

1 Preliminary Tests

In order to assert the difficulty of the problem (P), we conducted a test using the following simplified formulation:

$$
\begin{aligned}
\text{(P1)} \quad \min \ & (2) \\
\text{s.t.} \ & (1), (3)-(8), \\
& y_i \geq w_i, && \forall i \in N, && \text{(9-1*)} \\
& y_i \leq w_i + 0.999, && \forall i \in N, && \text{(9-2*)} \\
& 50 \leq \sum_{i \in N} y_i \leq 70 && && \text{(9-3*)} \\
& y_i \in \{0,1\}, && \forall i \in N, && \text{(9-4*)} \\
& d^T \Omega d \ \leq 0.01. && && \text{(11*)}
\end{aligned}
$$

Constraints (9-1*)-(9-4*) are due to Proposition 1. Constraint (11*) is obtained by squaring both sides of constraint (11). Note that the problem (P1) is a relaxation of (P) due to lack of constraint (10) and non-convex part of constraint (11). The problem (P1) is a mixed integer quadratic constrained quadratic programming (MIQCQP) problem that can be solved by off-the-shelf solvers such as CPLEX. The decision variables y_i is 1 if asset $i \in N$ is selected (i.e., $w_i \geq 0.001$). We solved (P1) for the time period of January 2007 with CPLEX 12.6 on a MacPro machine with Intel Xeon 3.5GHz CPU.

Table 1 shows the number of integer feasible solutions found by branch and bound for a given time. The column GAP represents the difference (%) from the best low bound, which is given as $\frac{\text{obj. value of best integer solution-best lower bound}}{\text{obj. value of best integer solution}} \times 100$. Large GAPs shown in the table imply that the problem (P1) is pretty hard to solve, even it is significantly relaxed from the original problem (P). Non of the test cases could find an optimal solution within 3 minutes, which means that solving the formulations (P1) or (P) directly using off-the-shelf solvers is hopeless.

The calculated values of AS and TE from the best integer solutions are also shown in the table. We found that among constraints for the AS and TE, the constraint for the TE ($0.05 \leq \sqrt{d^T \Omega d}$) is

more prone to violate, which is understandable by noting that the TE is supposed to be minimized from the objective function. Based on the results of the preliminary experiments, the following observations were derived.

(O1) Due the existence of many binary decision variables, it is very hard to solve problem (P) as it is.

(O2) Once the binary decision variables were fixed, solving (P1) is easy (e.g., less than a second).

(O3) The constraints for the TE and AS must be addressed separately to ensure feasibility.

(O4) In particular, TE values of the portfolios must be increased to satisfy constraint (11).

Motivated by the above observations, we propose a three-step heuristic approach. The first step determines a set of assets to invest. Pre-determination of the set of assets enables us to eliminate most of binary decision variables from the problem (P1). The second step involves finding of feasible solutions under the given set of assets. The basic idea for this is to solve (P1) after fixing the binary decision variables y_i according to the given set of assets. When the solution if infeasible to either (10) or (11), we add some constraints to the formulation (P1) to make the portfolio have higher TE value. The last step improves the feasible portfolio by repeatedly updating the pre-determined set of assets. In the following sections, we explain each step in details.

2 Step 1: Selection of Initial Set of Assets to Invest

This step needs to select an initial set of assets to invest. Ideally, selection of assets to invest must be able to consider all constraints of (P) and the objective function. Our computational experiments discourage any mathematical programming based approach for this purpose, because of too many binary decision variables. We propose a generative adversarial network (GAN) based approach for this. Before presenting the detailed algorithm developed, we first give a rather brief introduction on the GAN.

2.1 Generative adversarial networks (GANs)

A GAN is a variant of deep neural net (DNN) architectures comprised of two networks (**?**). One neural network, called **Generator (G)**, generates new data instances, while the other, **Discriminator (D)**, decides whether each instance of data is real (real images) or fake (generated by G). Both neural networks learn to alternate with each other. The G trains to generate data similar to real data in order to deceive D, and the D trains to distinguish real data from fake data generated by G. Figure 2 illustrates a basic structure of GANs. **G** generates fake data when arbitrary noise z is given as an input through a NN composed of layers. Let $D(x)$ denote an output of **D** with

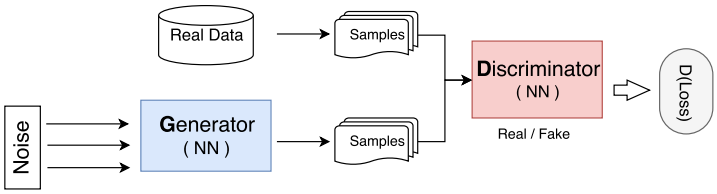

Figure 2: A strucutre of basic GANs

x as an input x. **D** is trained as $D(x) = 1$ whenever x is real otherwise $D(x) = 0$. **G** tries to make $D(x) = 1$ for sample x generated by **G**, and **D** tries to make $D(G(z)) = 0$ for any samples generated by $G(z)$, where z is a randomly generated value. Therefore, the loss function $V(D, G)$ of GAN can be expressed as follows (**?**):

$$\min_G \max_D V(D,G) = \mathbb{E}_{x:\text{real}}\left[\log D(x)\right] + \mathbb{E}_{z:\text{randomly generated}}\left[\log(1 - D(G(z)))\right]$$

Typical GANs have two NNs, which are trained iteratively. Each training iteration fixes either **G** or **D**, and updates the other un-fixed NN using well-known backpropagation algorithm. Contrary to ordinary DNNs that are good at classification of any given data, GANs are able to generate artificial data that look like real ones. GANs are actively studied and utilized for image-related applications especially with incorporating convolutional neural networks (CNN). There are little previous studies using GANs for solving combinatorial optimization problems (**?**). To the best of our knowledge, this study is the first attempt for solving the portfolio optimization problem using GANs. The basic idea is using the generating power of GANs to populate good candidates for the initial sets of assets to invest.

2.2 Generating Initial Sets of Assets using GAN

In general, training of the discriminator **D** requires a large number of labeled samples. For our portfolio problem, we propose a discriminator that does not need to be trained but determines if generated portfolio is feasible or not. Specifically, the loss function $D(w)$ for a given portfolio weight vector w is defined as:

$$D(w) = 1 - \tanh(L(w)), \tag{12}$$

where $L(w)$ is a "measure" of infeasibility of the portfolio against the problem (P). Without a loss of generality, we represent jth constraint of (P) by $\underline{b}_j \le f_j(w) \le \overline{b}_j$. Let $M_{(l)}$ denote the set of indices of constraint (l). Then, the discriminator **D** checks the feasibility of the following system of inequalities:

$$\underline{b}_j \le f_j(w) \le \overline{b}_j, \quad \forall j \in M_{(l)}, l = 4, 5, 6, 7, 8, 10, 11, \tag{13}$$

$$50 \le \sum_{i \in N} \tanh(w_i) \le 70, \tag{14}$$

$$(w - w^{bench})^T \Omega (w - w^{bench}) - \lambda (w - w^{bench})^T \alpha \le Z, \tag{15}$$

Figure 3: The strucutre of GAN for portfolio optimization

where Z is a given parameter that represents the best objective value of feasible solutions of (P). Note that $\sum_{i\in N}\tanh(w_i)$ of (14) is differentiable and an approximation of $card(w)$. Let $relu(x) := \max\{x, 0\}$ (i.e., *rectified linear* activation function). Then, we define

$$\begin{aligned} L(w) = & \sum_{l=4,5,6,7,8,10,11} \sum_{j\in M_{(l)}} \left[relu(\underline{b}_j - f_j(w)) + relu(f_j(w) - \overline{b}_j)\right] \\ & + relu\left(\sum_{i\in N}\tanh(w_i) - 70\right) + relu\left(50 - \sum_{i\in N}\tanh(w_i)\right) \\ & + relu\left((w - w^{bench})^T\Omega(w - w^{bench}) - \lambda(w - w^{bench})^T\alpha - Z\right). \end{aligned}$$

In other words, $L(w)$ is the sum of the degrees of violations of the constraints and objective value. If a sample portfolio w is feasible to all constraints *and* its objective value is no worse than the previous best, $L(w)$ becomes zero.

The generator **G** is a deep NN structure consisting of one input layer, three hidden layers, and one output layer. **G** tries to minimize $D(w)$ for the sample $w = G(z)$ generated from the random input noise $z \sim \mathcal{U}(-1, 1)$. Unlike the typical GAN structure, GAN for portfolio optimization has **G** of NN structure, but **D** is not, and only **G** is required to train. The loss function $V(G)$ of GAN can be expressed as follows :

$$\max_G \mathbb{E}_{z\sim\mathcal{U}(-1,1)}\left[\log(D(G(z)))\right]$$

We trained the generator **G** for 2000 iterations by using the standard stochastic gradient algorithm (**?**). The size of mini batch for the gradient calculation was 128. During the iterations, any sample portfolio w is considered as feasible if $D(w) > 0.995$. Figure 3 illustrates the structure of GAN for the portfolio optimization.

After the training finished, we let the generator **G** produce many sample portfolios. Let $\hat{w}$ denote the best portfolio generated in terms of $D(w)$, the set of assets to invest $\hat{N}$ is defined as

$$\hat{N} = \{i \in N \mid \hat{w}_i \geq 0.0001\}. \tag{16}$$

We emphasis that even the generator gives portfolio weight w as output, we do not use it directly as a solution. It is well-known that the training of NNs does not converge to global optima, and generated portfolio weights often do not satisfy the constraints completely. Nonetheless, we expect any assets with larger weight values are good candidates for the set of assets to invest.

3 Step 2: Finding a Feasible Solution under Given Set of Assets

Let $\hat{N}$ denote the set of assets to invest given from the previous step. We consider the following problem:

$$
\begin{aligned}
\text{(P2)} \quad \min \ & (2) \\
\text{s.t.} \ & (1), (3)-(8), \\
& w_i \le 0, && \forall i \in N \backslash \hat{N}, \\
& w_i \ge 0.001, && \forall i \in \hat{N}, \\
& d^T \Omega d \le 0.01.
\end{aligned}
$$

The problem (P2) finds a portfolio weight w such that only assets in $\hat{N}$ are greater or equal to the threshold value of 0.001. It is a convex QCQP problem. As indicated by **O2**, solving the problem (P2) is fairly easy, but the obtained solution is often not feasible to (P). In this section, we present a heuristic algorithm based on bisectional search to find the feasible solutions using (P2). According to **O3** and **O4**, TE value $d^T \Omega d$ is often too small so does not satisfy the lower bound (i.e., $0.0025 > d^T \Omega d$). To increase the TE value, it is required to make the weight w be more unlike w^{bench}. In other words, the difference between the two weights must be increased to some extent to satisfy the constraint. Consider a solution of (P2) w^0 that is infeasible to (P) for a given set of assets $\hat{N}$. Let $N^+ := \{i \in N \mid w_i^0 > 1/|\hat{N}|\}$, and $W^k = \sum_{i \in N^+} w_i^k$. We add the following constraint to (P2):

$$\sum_{i \in N^+} w_i \ge (1 + W^0)/2. \tag{17}$$

We solve the problem (P2) with added constraint, which gives a solution w^1. If w^1 is also feasible to (P), we modify the right-hand-side value of (17) as

$$\sum_{i \in N^+} w_i \ge (W^1 + W^0)/2.$$

Otherwise, w^1 is not feasible to (P), we modify the right-hand-side value of (17) as

$$\sum_{i \in N^+} w_i \ge (1 + W^1)/2.$$

We repeat the above procedure until either $|1 - W^k|$ or $|W^k - W^{k-1}|$ is less than a given tolerance 0.001.

An example is shown in Figure 4. Assume that $W^0 = 0.6$. First, the solution is searched by dividing the interval between $0.8 (= \frac{1+W^0}{2})$ and 1.0. If the new solution is feasible to (P), narrow the range and search again in the left section ([0.7, 0.8]). Otherwise, we increase the sum of weights of $\hat{N}$ even more to [0.9,0.1]. One of the feasible solutions, obtained from the bisectional search with the lowest objective value, is returned.

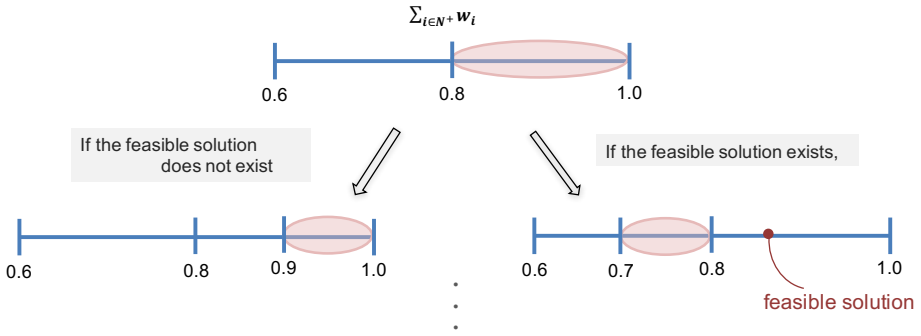

Figure 4: An Example of Bisectional Search

4 Step 3: Local Search for Improving Solutions

As a result of Step 2, the feasible solutions are obtained. At this point, the solutions are improved by a local search heuristic. The idea of the heuristic algorithm is to replace an asset, namely *leaving asset*, in the set of asserts to invest by another assert, namely *entering asset*, in outside the investing set, if the exchange seems beneficial in terms of objective function value. Conceptually, our local search heuristic mimics the well-known simplex algorithm. We maintain a set of assets whose weights can be positive, as basis in the simplex algorithm, and to improve the objective value we repetitively do exchanging operation, as pivoting operation in the simplex method. A difference from the simplex algorithm is instead of reduced cost our heuristic employs a simplified method to determine which is to enter and to leave.

Exchanging operation: For a given set of assets $\hat{N}$, we set $\hat{N} \leftarrow \hat{N} \backslash \{l\} \cup \{e\}$, where

$$\text{Entering asset: } e = \arg\max_{i \in N \backslash \hat{N}} \left\{ \sum_{j \in \hat{N}} \Omega_{ji} + \Omega_{ii} - \lambda \alpha_i \right\} \tag{18}$$

and

$$\text{Leaving asset: } l = \arg\max_{i \in \hat{N}} \left\{ \sum_{j \in \hat{N}: j \neq i} \Omega_{ji} + \Omega_{ii} - \lambda \alpha_i \right\}. \tag{19}$$

The exchanging operation is repeated until the given time limit is reached. During the exchanging operations, the feasible solution with the best objective value is returned as a solution.

4.1 Preventing Infeasibility

The updated set of assets $\hat{N}$ can be infeasible, which means Step 2 with the updated set of assets cannot find a feasible solution to (P). During our experiments, we have noticed that in most cases

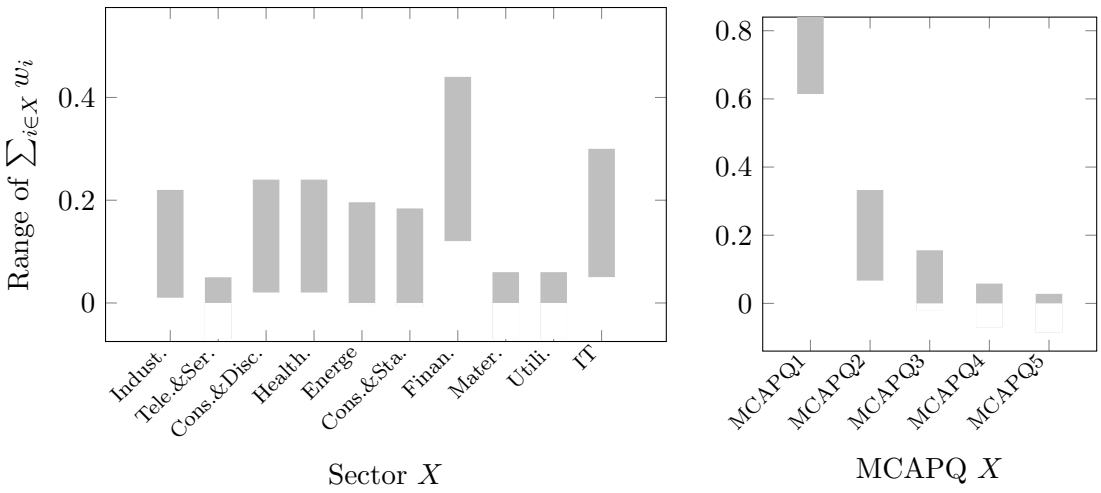

Figure 5: The sum of the weights for each Sector and MCAPQ

infeasibility occurs due to constraints (6) and/or (7). Figure 5 shows required sums of weights for sectors/MCAPQs to make constraints (6) and/or (7) with w^{bench} for January 2007. Note that the required sums of weights are significantly varying for different sectors/MCAPQs. We calculate the required sum of weights for each of sectors/MCAPQs after selecting the leaving asset. If the any sum of weights is outside of the required range, we select the entering asset only among that sector/MCAPQ. Our computational experiments show that this selective choice of entering asset greatly increase the possibility of being feasible after the exchanging operation.

5 Detailed Description of Algorithm

Figure 6 shows the flowchart of the proposed algorithm. The proposed algorithm greatly relies on the set of assets to invests that was generated by the GAN. Typically, the parameters of the GAN are initialized randomly. Because the training of NNs converges to local optima, different initialization values lead significantly distinct trained GANs. In other words, even though our GAN could take into account all constraints, the training of the GAN results in different local optima depending the starting conditions. To hedge this issue, we employed many independent GANs that were trained with different initial conditions (See Figure 7). For the parallel computing, we used Ipyparallel (`https://github.com/ipython/ipyparallel`) without any load balancing. For example, on a 8-core machine we may spawn 8 independent GANs with distinct initialization parameters, which yields 8 considerably different sets of assets. Another aspect to consider regarding the training of GAN is which NN computation framework to use. We used PyTorch (`http://pytorch.org/`) for our experiments, and training time was less than 30 seconds using only 1 CPU core. We expect the training time can be reduced by using GPU or cloud computing service.

Figure 8 illustrates the entire algorithm. When every process finishes Step 2, the obtained solutions are compared in the solution check stage. We pick top 50% solutions in terms of the objective value

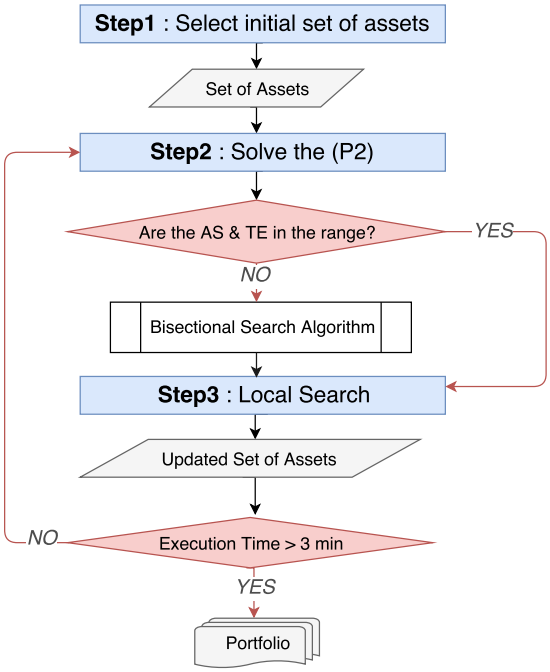

Figure 6: A Flowchart of Proposed Algorithm

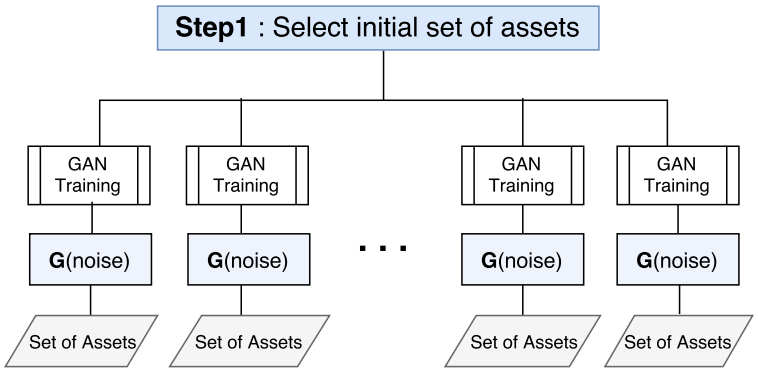

Figure 7: Parallel Training of Independent GANs

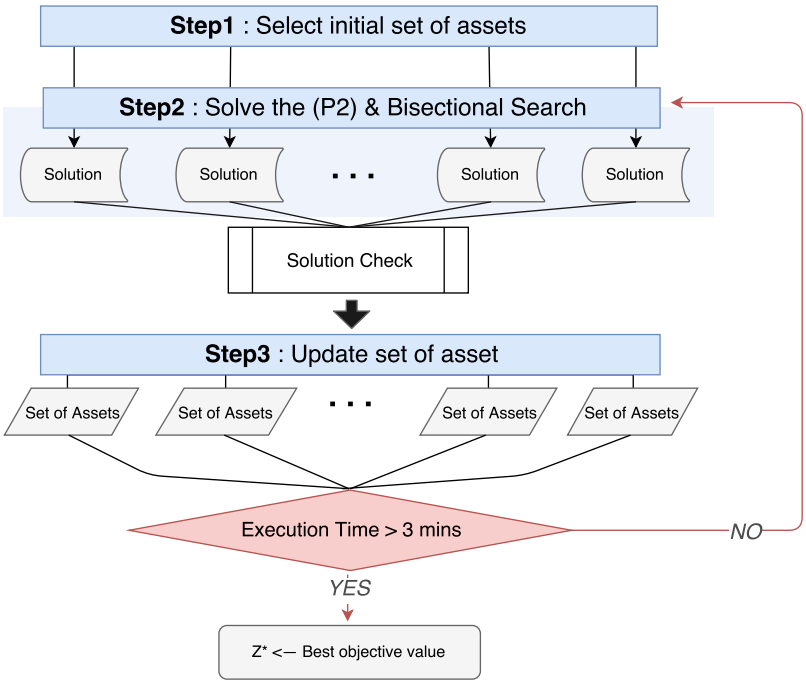

Figure 8: A Flowchart of Hybrid Approach

and execute Step 3 on these solutions. For the remaining solutions, we discard them and initiate Step 1 from scratch with different initializers. The algorithm terminates when the given time limit (3 minutes) reaches. After finishing all procedures, the algorithm returns the best portfolio solution.

Analytics Solution and Results

1 Experiments for Each Step

The proposed algorithm has a sequential process and roles assigned to each step are as follows:

- Step 1: Populating good initial sets of assets (GAN).
- Step 2: Obtaining feasible solutions under the given set of assets (Bisectional search algorithm).
- Step 3: Improving feasible solutions obtained at the previous step (Local search algorithm).

In this section, we report computational results that confirm validity of each step.

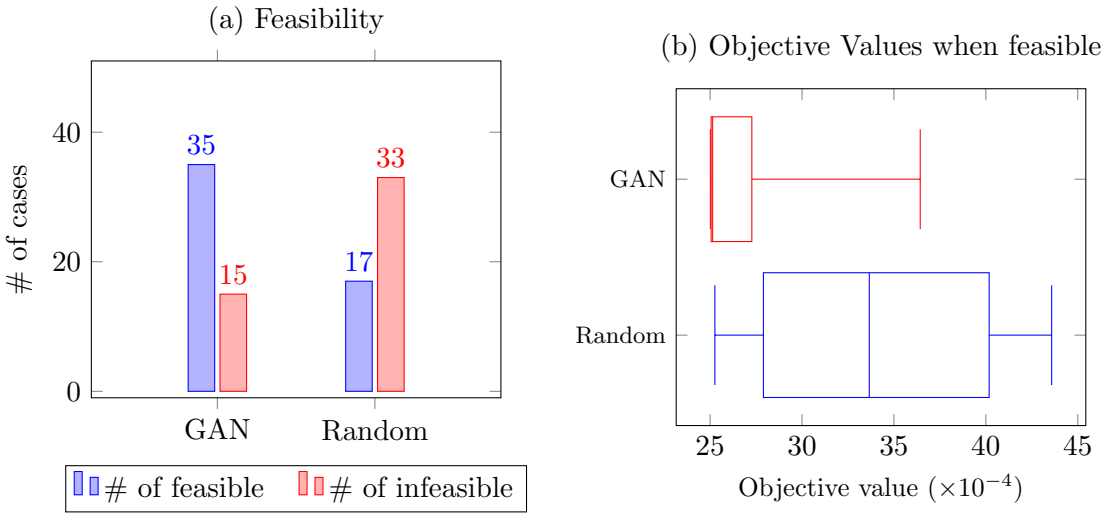

Figure 9: Experiments Results for Step 1

1.1 Step 1 : Select Initial Set of Assets

We conducted the following experiments to show the performance of our GAN-based initial set selection process. We compared the GAN-based approach to a simple random set selection method. The experiment settings are as follows:

- Number of experiments : 50 times
- Training time for GAN : 20 seconds
- Neural network library : PyTorch (`http://pytorch.org/`)
- Data of the first period (January 2007)
- Baseline method: randomly selecting 60 assets for each initial set

We injected the populated sets from two methods into Step 2 (Bisectional search algorithm), and evaluated the feasibility and objective values if feasible. Figure 9 compares the results of two methods. The numbers of feasible cases vs. infeasible cases out of 50 trials are shown in Figure 9(a). It is clearly shown that the GAN-based approach outperforms the random selector. When a feasible set is found, the expected objective value is much better for the GAN-based approach as shown in Figure 9(b).

Insights and Suggestion: the GAN is composed of NNs, and the quality of results, especially training time, heavily depend on computing environment. Therefore, we recommend that Principal could improve the training speed and quality by using GPU-accelerated computing or through

some advanced technologies such as Amazon Cloud services. Moreover, the structure of generator **G** is worth of further investigating because the numbers of hidden layers and units are of crucial importance in training of GAN. We also found that the fully-trained GANs often produce considerably different initial sets because of different initial conditions. We would recommend employing parallel training of multiple GANs with distinct initial parameters, which results in much divergent initial sets.

1.2 Step2 : Solving (P2) & Bisectional Search

In Step 2, we solve the convex QCQP problem (P2) with a given set of assets $\hat{N}$. Because the problem (P2) relaxes two non-convex constraints, we need to warrant the feasibility of the solution obtained. As we stated before, we do a bisectional search to find a feasible solution by increasing the sum of weights of a subset assets. Figure 10 shows the changes of the TE and AS values during the search algorithm. We found that the AS values are rarely violating the constraint (red line in the figure), while the TE values are typically well below from the required lower bound. In Figure 10, is is shown that the TE values are fluctuating significantly because of the added constraint (17). The black squares represent finding of feasible solutions. It is evident that during the bisectional search, many feasible solutions can be found. We return the feasible solution with the best objective value.

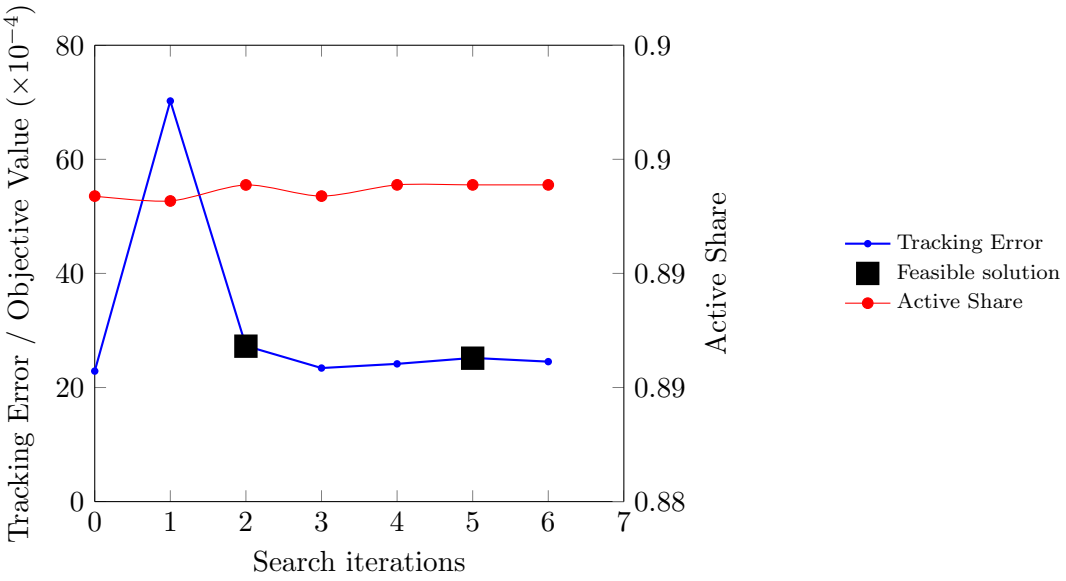

Figure 10: The Results of Bisectional Search

Insights and Suggestion: Even the bisectional search is able to find good feasible solutions quickly, the added constraint (17) can make the feasible solutions somewhat restrictive because the set of assets to increase N^+ is fixed. One can try different choice for N^+, or update the set N^+ on the fly during search.

1.3 Step 3: Local Search for Improving Solutions

The feasible solution obtained from Step 2 is improved in this step. A good asset that might reduce the objective value is identified by (18), and added to the current set of assets to invest. Conversely, the asset that might make the objective value higher is removed by using (19). Figure 11 shows the changes of objective values of the solutions with repeating the exchanging operations for 60 seconds. It is clear that the objective values are kind of "saturated" around 0.0025. This behavior is because the term $d^T \Omega d$ approaches to 0.0025 due to the TE constraint (11). We found that compared to $d^T \Omega d$, another term in the objective function $d^T \alpha$ contribute very little, which results in the objective values of many good solutions being around 0.0025.

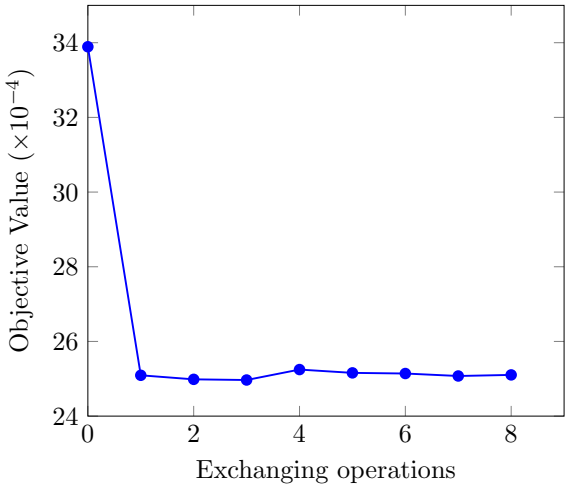

Figure 11: The Results of Local Search

Insights and Suggestion: The given scaling choice of Ω and α implies that the "true" optimal portfolio should be those with minimum risk. That is, even the objective function has two terms for the risk and return, the risk should be considered more seriously than the return. This observation motivates a goal programming based approach. For example, we may add constraint $d^T \Omega d \leq 0.0025 + \epsilon$, and maximize only the return $d^T \alpha$, where ϵ is a very small value.

2 Computational Experiments

In this section, we report results of computational experiments for solving portfolio optimization with the methodology we present. The experiment was conducted with the provided data, and no future data was used at the point time concerned. For every rebalance to rebalance, maximum of 3 minutes of computational time was used for the whole computation.

Experiment environment : The experiments were performed on an Intel Xeon 3.5 GHz MacPro machine with 32GB memory and ILOG CPLEX 12.6 were used as an MIP solver, and PyTorch for Python 3.6 was used as a deep learning framework.

Table 2: The Computational Results of the Algorithm Proposed (year of 2007)

Instance		**Output**					**Execution Time (sec)**				
Date	# of asset	# of assets to invest	Objective Value ($\times 10^{-4}$)	r_{Txadjt}	r_{optt}	turnover	Pre -processing	Step1	Step2	Step3	Total
01/31/2007	493	70	30.81	0.005	0.011	1.116	0.269	53.09	71.12	44.37	168.86
02/28/2007	494	70	30.80	-0.016	-0.011	1.107	0.274	53.35	70.18	42.78	166.59
03/28/2007	494	70	31.02	-0.005	0.0005	1.193	0.275	53.17	70.56	43.39	167.41
04/25/2007	494	70	30.75	0.047	0.052	1.034	0.276	53.29	70.6	43.84	168.02
05/23/2007	494	64	29.56	0.019	0.023	0.776	0.272	53.18	71.79	44.61	169.87
06/20/2007	494	60	31.82	-0.009	-0.003	1.322	0.277	53.13	70.35	42.13	165.90
07/18/2007	493	70	31.88	-0.001	0.006	1.351	0.284	53.35	70.19	43.97	167.81
08/15/2007	493	70	30.47	-0.071	-0.066	1.030	0.271	57.14	72.15	44.46	174.04
09/12/2007	493	70	29.93	0.019	0.024	0.932	0.277	53.19	70.39	44.04	167.90
10/10/2007	495	70	30.52	0.084	0.089	1.063	0.274	53.20	70.09	42.57	166.15
11/07/2007	495	70	30.32	-0.064	-0.059	1.062	0.277	53.26	71.08	44.19	168.81
12/05/2007	495	70	30.15	-0.009	-0.005	0.91	0.276	53.15	70.98	42.68	167.10
Mean		**68.66**	**30.67**	**-0.0003**	**0.005**	**1.075**	**0.275**	**53.54**	**70.79**	**43.59**	**168.20**

Computational Results : The experiment was performed for all given periods, and Table 2 shows the results of experiment for the year of 2007. In most cases, the objective value is close to the lower bound of the tracking error, which is restricted by constraint (11). Therefore, we might say our solution is near optimal. On the other hand, the value of r_{opt}, calculated by the product of the actual return r, and the portfolio weight w is relatively irregularly formed. The execution time for the proposed algorithm is also shown in the table. The execution time of Step 1, which is the process of selecting the initial set of assets using the GAN, includes the time for forming the network and the training time. Step 2 includes the setup and formulating time to solve (P2) and the time for the branch and bound by CPLEX, and the execution time of Step 3 contains all the time that occurs in the process of improving the solution by doing a local search. Therefore, we ensure that the proposed algorithm produces relatively good solutions in a rather short execution time (¡ 3 minutes). We note that the MacPro machine used for the experiments only has 6 CPU cores, which may deteriorate the computational times because we spawned total 8 processes. A more powerful machine would allow us to spend more time in each step, which would improve the quality of solutions.

The Portfolio Performance Statistics : The portfolio performance statitstics below is averaged over 10 years of data.

Portfolio Performance Statistics

3 Parameter Setting

Out approach provides many algorithmic parameters. Ideally, to optimally tune parameters, various experimental analyzes are required with real data. However, given limited information on construction methods of the data such as α, Ω, β, and bench weights. We assume that the year of 2007 is a parameter adjustment period, and adjusted our parameters based on the results of 2007.

2007-01-01 to 2016-12-31	Portfolio	Benchmark
Cumulative Return	13.1%	20.3%
Annualized Return	13.1%	20.3%
Annualized Excess Return	-7.3%	–
Annualized Tracking Error	3.7%	–
Sharpe Ratio	3.8	
Information Ratio	-7.8	–

Note that for the year of 2007, we simply used default values (e.g., 1) for all parameters, because we are now allowed to use the future outcomes to tune the past portfolios.

The Parameter λ: Parameter λ reflects how much return will be considered relative to risk in the objective function. When the value of λ is large, the return is supposed to be considered more, and when the λ value is small, the risk is considered more. We used results of the year of 2007 to analyze whether the $d^T\alpha$, which represents the return, has an influence on the overall return index (r_{opt}) as we adjusted the λ. In other words, as the λ value increases, more revenue will be taken into account, so that it can be inferred that the r_{opt} is more affected by return or risk. Therefore, we conduct the simulation by changing λ with the first year of data (see Figure 12). Then, we compare the experimental result with the actual return value to see how much the parameter λ affects the overall return. Experimental results show that the distribution of r_{opt} is not affected by λ values. We believe the causes of the experimental results as follows: First, the scale of α which is an index of return, and the scale of Ω which is an index of risk, are very different in the objective function. In other words, the α is too small, so even if the parameter λ is 10, return is still considered small. The second reason is that there is a very low correlation between α and r, further consideration of α does not improve actual returns.

4 Rabalancing

Portfolio turnover depends on how many assets in the portfolio change from rebalance to rebalance, and the value of turnover has effects on Information Ratio (IR). Therefore, it is required to consider turnover in the portfolio optimization. We apply additional consideration for the turnover to the both GAN for finding initial set of assets and the QCQP problem for determining the weight of the selected assets. For applying it to GAN, the calculated turnover$_t(\sum_{i\in N}|w_{i,t} - w^{pre}_{i,t-1}|)$ was added to the loss function of the discriminator **D**. On the other hand, for applying it to (P2), it should

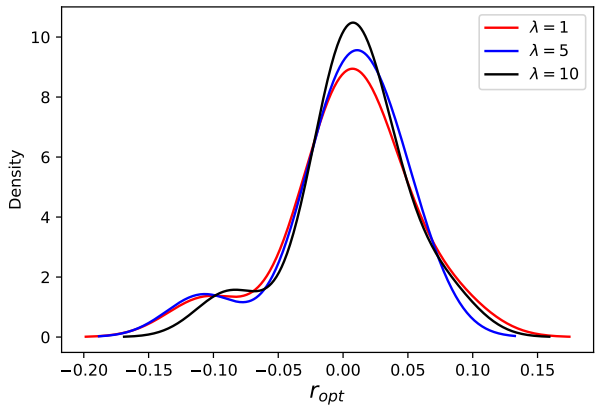

Figure 12: Kernel Density Estimation of r_{opt} for the year of 2007 with different λs

be linearized, which results in the following formulation:

$$
\begin{aligned}
\text{(P2-TO)} \quad \min \ & d^T \Omega d - \lambda d^T \alpha + \omega \sum_{i \in N} o_i \\
\text{s.t.} \ & (1), (3) - (8), \\
& w_i \leq 0, && \forall i \in N \backslash \hat{N}, \\
& w_i \geq 0.001, && \forall i \in \hat{N}, \\
& d^T \Omega d \leq 0.01. \\
& o_i \geq w_{i,t} - w_{i,t-1}^{pre}, && \forall i \in N \\
& o_i \geq -w_{i,t} + w_{i,t-1}^{pre}, && \forall i \in N
\end{aligned}
$$

The decision variables o_i is the difference between w_i of previous period $(t-1)$ and w_i of current(t) period for all asset $i \in N$. Because of constraints added, $w_{i,t} - w_{i,t-1}^{pre}$ is positive for all asset $i \in N$. The parameter ω represents the weight of turnover for objective value considering risk and return.

The Parameter ω: Experiment with a large value of ω considers turnover a lot when compared to risk-return. In the opposite side, the turnover cost is significant. Since our computational experimence with the year of 2007 showed that $\omega = 5$ would yield decent results, we set $\omega = 5$ in the following years.

5 Dealing Uncertainty of α

The results obtained earlier are derived by deterministic values which are α and Ω. However, this is a parameter value with uncertainty, so we figure out the change of α and Ω in the historical data. We first tried to secure the value of risk and alpha score for 2007 by analyzing the historical data. The analyzed data is obtained from Yahoo Finance and is historical data from 1970 to 2017.

For the first time, we are looking at the risk of an asset in 2006 with extensive historical data. We obtained data on close stock prices for approximately 400 assets from historical data among 492 assets included in January 3, 2007 in S & P 500 and derived a cross covariance matrix. And then, the correlation between the covariance matrix and the full matrix of Ω was derived. We expected a large correlation between the two matrices, but the result was not. It is judged that it is difficult to deduce Ω value from the covariance matrix derived from historical data. Therefore, we obtain the range of parameter change for each asset with the first year data received from the Principal and reflect it in the following robust optimization.

Let set $U := \{\tilde{\alpha} \in \mathbb{R}^{|N|} \mid \tilde{\alpha}_i = \alpha_i + \bar{\alpha}_i\gamma_i, \quad -1 \leq \gamma_i \leq 1, \quad \sum_i |\gamma_i| = \Gamma\}$, which contains all possible realizations of α. Our goal is to consider the worst-case realization of U, which means that the expected return of the active weight $(d^T\tilde{\alpha})$ is guaranteed above the worst-case value. The mathematical formulation addressing the uncertainty of α is stated as follows:

$$\begin{aligned} \text{(RP)} \quad & \min \ d^T\Omega d - \lambda \min_{\tilde{\alpha}\in U}\{d^T\tilde{\alpha}\} \\ & \text{s.t. } (1), (3)-(11). \end{aligned} \tag{20}$$

For a given d, the inner minimization problem is stated as

$$\min_{\tilde{\alpha}\in U}\{d^T\tilde{\alpha}\} = \min\left\{\sum_{i\in N} d_i(\alpha_i + \bar{\alpha}_i\gamma_i) : -1 \leq \gamma_i \leq 1, \forall i \in N, \sum_{i\in N}|\gamma_i| = \Gamma\right\} \tag{21}$$

$$= d^T\alpha - \max\left\{\sum_{i\in N}|d_i|\bar{\alpha}_i\gamma_i : 0 \leq \gamma_i \leq 1, \forall i \in N, \sum_{i\in N}\gamma_i = \Gamma\right\}. \tag{22}$$

As shown by **?**, the latter maximization problem has an integral polyhedron. Using the linear programming duality theory, it is easily shown that the following holds:

$$\max\left\{\sum_{i\in N}|d_i|\bar{\alpha}_i\gamma_i : 0 \leq \gamma_i \leq 1, \forall i \in N, \sum_{i\in N}\gamma_i = \Gamma\right\} = \tag{23}$$

$$\min\left\{\Gamma\pi + \sum_{i\in N}\theta_i : \pi + \theta_i \geq \bar{a}_i p_i, \ -p_i \leq d_i \leq p_i, \ \theta_i \geq 0, \ \forall i \in N\right\} \tag{24}$$

Then, the problem (RP) can be reformulated as

$$\text{(RP1)} \quad \min \ d^T\Omega d - \lambda\left[d^T\alpha - \left(\Gamma\pi + \sum_{i\in N}\theta_i\right)\right] \tag{25}$$

$$\text{s.t. } (1), (3)-(11),$$

$$\pi + \theta_i \geq \bar{\alpha}_i p_i, \qquad \forall i \in N, \tag{26}$$

$$-p_i \leq d_i \leq p_i, \qquad \forall i \in N, \tag{27}$$

$$\theta_i \geq 0, \qquad \forall i \in N. \tag{28}$$

Note that (RP1) reduces to the problem (P) when $\Gamma = 0$. Because reformulation of (RP1) involves only linear constraints and continuous decision variables, the proposed algorithm for (P) can be used for solving (RP1) without further modifications except the mathematical formulation.

Table 3: Comparision of Performance with different Γs

	Output		Execution Time (sec)				
	IR	r_{Txadjt}	Pre-processing	Step1	Step2	Step3	Total
$\Gamma = 0$	-3.47	0.005	0.28	35.53	84.49	44.27	164.59
$\Gamma = 5$	-1.32	0.007	0.26	32.61	141.34	29.04	203.27

Table 3 shows the results of the robust approach for the year of 2007. The case with $\Gamma = 0$ corresponds to the deterministic case, where no uncertainty of α is taken into account. It is noteworthy that the robust approach took comparable computational time, while the IR is significantly better. We believe that this behavior is because the robust approach could address the worst-case more wisely. Unfortunately, we could not finish complete experiments with 10 years due to the lack of time. We strongly believe that the proposed robust approach is of worth of further study.

References