European Journal of Operational Research 221 (2012) 407–416

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Decision Support

Robust ranking and portfolio optimization

Tri-Dung Nguyen [a,*], Andrew W. Lo [b]

[a] School of Mathematics and School of Management, University of Southampton, Southampton SO17 1BJ, UK
[b] Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02142, USA

ARTICLE INFO

Article history:
Received 14 December 2010
Accepted 13 March 2012
Available online 29 March 2012

Keywords:
Uncertainty modelling
Network flows
Portfolio optimization
Ranking
Mixed integer programming

ABSTRACT

The portfolio optimization problem has attracted researchers from many disciplines to resolve the issue of poor out-of-sample performance due to estimation errors in the expected returns. A practical method for portfolio construction is to use assets' ordering information, expressed in the form of preferences over the stocks, instead of the exact expected returns. Due to the fact that the ranking itself is often described with uncertainty, we introduce a generic robust ranking model and apply it to portfolio optimization. In this problem, there are n objects whose ranking is in a discrete uncertainty set. We want to find a weight vector that maximizes some generic objective function for the worst realization of the ranking. This robust ranking problem is a mixed integer minimax problem and is very difficult to solve in general. To solve this robust ranking problem, we apply the constraint generation method, where constraints are efficiently generated by solving a network flow problem. For empirical tests, we use post-earnings-announcement drifts to obtain ranking uncertainty sets for the stocks in the DJIA index. We demonstrate that our robust portfolios produce smaller risk compared to their non-robust counterparts.

1. Introduction

Ranking problems are common in real life for those applications that involve comparison and evaluation of objects such as in asset selection. A ranking between two objects simply provides information about which of the two objects is preferable. In portfolio construction, managers often have preferences on the assets they are investing. They might prefer one stock (or one sector of stocks) over other stocks (or other sectors). The ranking of the assets could depend on individual preferences or on empirical evidence. There is an extensive literature on accounting anomalies such as the post-earnings-announcement drift Ball and Brwon [3] and Fama–French risk factors Fama and French [15] that can be used to select stocks and to obtain their ranking. The main advantage of using ranking over the traditional mean–variance portfolio construction is in avoiding the estimation of the expected returns, which is a very challenging task.

Although the idea of using ranking is quite intuitive, and it is easy to build trading strategies from a given ranking, the ranking itself needs to be estimated and there is uncertainty associated with this. Hence, we often see ranking expressed in the form of confidence intervals. For example, it is more natural for a manager to believe an asset to be in the top five/top ten, or in the bottom five/bottom 10 rather than to claim it to be the best (or the worst) one. Moreover, ranking can derive from many different sources, e.g. each analyst might come up with a different ranking, so there is always uncertainty. Consider a situation where an investor wants to invest in a long-short portfolio of ten assets based on the assets' ranking. Suppose asset X is among these ten assets and is predicted to have a ranking between second and sixth. If we use the average ranking, which is fourth in this case, then the investor is likely to invest a positive proportion of funding on this asset because it is in the top half. However, this investment strategy does not prepare for the worst case scenario when the asset has the sixth rank. This motivates us to develop a robust ranking model to find the optimal policy, e.g. the portfolio weights, to maximize an objective function, even for the worst realization of the ranking within the uncertainty set. This robust optimization framework for handling uncertainty differs from the traditional stochastic programming approach in several ways and there have been debates on which one is more suitable. We refer the readers to Ben-Tal et al. [5] and Bertsimas et al. [8] for the discussions on why robust optimization is more appropriate in some applications. We find robust optimization an attractive approach in finance because investors have become more and more risk-averse and are more concerned about worst-case secenario due to the recent global financial crisis and the current Eurozone crisis. Notice, however, that our robust ranking framework is applicable to others areas such as betting on horse races and in sport in general, and for search engines when deciding the list of websites to display after each search.

The robust ranking model is a minimax mixed integer programming problem, which is often hard to solve. We apply the

* Corresponding author. Address: Room 10005, Building 54, Southampton University, Southampton SO17 1BJ, UK. Tel.: +44 (0)23 8059 7759; fax: +44 (0)23 8059 5147.
E-mail address: T.D.Nguyen@soton.ac.uk (T.-D. Nguyen).

0377-2217/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2012.03.023

constraint generation method where constraints are efficiently generated using a network flow model. We show that the robust ranking model can be solved efficiently through numerical results. We also apply the robust ranking model to portfolio optimization, where the ranking uncertainty set is obtained from post-earnings-announcement drifts. In the next Subsection, we will review the existing literature related to our robust ranking model and portfolio optimization.

1.1. Literature review

The traditional mean–variance portfolio optimization framework of Markowitz [21] aims to find an optimal portfolio to maximize a utility function whose input parameters include the expected returns and the covariance matrix. Since then, many variants of the mean–variance portfolio optimization framework and their corresponding computational techniques have been proposed to incorporate practical constraints such as short selling restriction and transaction costs (see, for example, [17,19]). Nevertheless, many researchers, including Michaud [22], and Chopra and Ziemba [13], have shown the poor out-of-sample performance of the mean–variance optimal portfolios due to estimation errors on the expected returns. Due to this drawback, many practitioners have used nonparametric ranking methods for portfolio optimization. The main advantage of using ranking is to avoid the difficult task of estimating the expected returns. Almgren and Chriss [2] use assets' ranking instead of assets' returns to model the portfolio optimization problem. Although the ranking that Almgren and Chriss used was easy to obtain, it is still prone to estimation error. We present a robust ranking model in which the ranking is allowed to be in an uncertainty set. This research is along the lines of the robust optimization approaches, which are very prevalent. The earliest work on robust optimization was Soyster [26] in 1973. The key for an attractive robust optimization model is to ensure the construction of the uncertainty set is intuitive and captures well the uncertainty nature of the parameters while the problem is still computationally tractable. Ben-Tal and Nemirovsk [4], and Bertsimas and Sim [6] presented a series of papers on robust optimization with different frameworks for controlling the conservative levels for different problem structures. Since then, the robust optimization frameworks have been applied to many different applications, such as inventory control and dynamic pricing (see [1,7,8,24]). For the most updated theory and development of the field, see Beyer and Sendhoff's comprehensive review [9].

In the field of robust portfolio optimization, Natarajan et al. [23] and Chen et al. [12] assume only the knowledge of the expected returns and the covariance matrix, while not restricting the stocks' returns on any fixed distribution. In contrast, Goldfarb and Iyengar [16], and Bienstock [10] develop different frameworks for the uncertainty sets building around the expected returns and the covariance matrix. Goldfarb and Iyengar use a factor model for the assets' returns, and the uncertainty arrives naturally from the estimation of the parameters of the factor model. This construction leads to a first norm uncertainty set for the expected returns and a second norm uncertainty set for the covariance matrix. Finally, they transform the robust model into a second-order cone program (SOCP) which is computationally tractable using primal–dual interior-point methods (see Lobo et al. [20]). Bienstock, on the other hand, constructs the uncertainty set around the level of deviations of the assets' returns from their expected values. These deviations are both in terms of the number of "misbehavior" assets, as well as their total returns. This model becomes a mixed integer optimization problem. Bienstock then uses a decoupling technique to solve it efficiently.

Our robust model is different from all the aforementioned methods, due to the difference in the parameter space of the problem. Since the parameter in the ranking problem is the ranking itself, which is discrete, the uncertainty set in our case is discrete. This combinatorial characteristic makes the problem very difficult to solve in general. We exploit the nice property of the ranking and apply the constraint generation method to solve the robust ranking problem. The constraint generation process is done through solving a transportation problem. Our method of controlling the conservative level is also different from the earlier works. We incorporate the deviation of the rankings from a nominal ranking into the objective function itself. This method in effect creates a non-uniform uncertainty set where those rankings that are further from the nominal ranking are penalized more compared with those that are closer.

1.2. The robust ranking problem

Consider n objects with a ranking $\boldsymbol{R} = (R_1, \ldots, R_n)$, which is a permutation of the set $\{1, \ldots, n\}$. For example, in the case $R_i = i$, $\forall i \in \{1, \ldots, n\}$, object i is ranked ith in the set. We want to find a weight vector $\boldsymbol{\omega}$ that belongs to some generic constraint set $\mathcal{P}(\boldsymbol{\omega})$ to maximize some generic objective function $f(\boldsymbol{\omega}, \boldsymbol{R})$. An example of the constraint is $\mathcal{P}(\boldsymbol{\omega}) = \{\boldsymbol{\omega} | \boldsymbol{\omega}^t \boldsymbol{e} = 1\}$ to make sure the weights are normalized and an example of the objective function is $f(\boldsymbol{\omega}, \boldsymbol{R}) = \boldsymbol{\omega}^t \boldsymbol{R}$ to maximize the weighted average ranking. If the ranking $\boldsymbol{R}$ is known, the problem becomes:

$$\begin{aligned} \max_{\boldsymbol{\omega}} \quad & f(\boldsymbol{\omega}, \boldsymbol{R}) \\ \text{s.t.} \quad & \boldsymbol{\omega} \in \mathcal{P}(\boldsymbol{\omega}). \end{aligned}$$

In this case, the optimization problem can be solved easily for many classes of objective function f and constraint set $\mathcal{P}$. However, in many situations, the ranking is unknown. Instead, we assume that we only know the ranking belongs to some known polytope of the form:

$$\mathcal{U}(\boldsymbol{R}) = \{\boldsymbol{R} \in \Pi(1, \ldots, n) | R_i \in Q_i, \forall i \in \{1, \ldots, n\}\},$$

where $\Pi(1, \ldots, n)$ is the set of all the permutations of the set $(1, \ldots, n)$ and Q_i is a subset of the set $\{1, \ldots, n\}$. For example, Q_i could be a confidence interval $[l_i, u_i]$ such that the constraint $R_i \in Q_i$ is equivalent to $l_i \leqslant R_i \leqslant u_i$. The robust ranking problem aims to find the weight vector $\boldsymbol{\omega}$ to maximize some objective function $f(\boldsymbol{\omega}, \boldsymbol{R})$ for the worst realization of the ranking $\boldsymbol{R}$ within the uncertainty set $\mathcal{U}(\boldsymbol{R})$ as shown in Model 1:

Model 1: General Robust Ranking (**GRR**) Model

$$\begin{aligned} \textbf{GRR} := \max_{\boldsymbol{\omega}} \quad & [\min_{\boldsymbol{R} \in \mathcal{U}(\boldsymbol{R})} f(\boldsymbol{\omega}, \boldsymbol{R})] \\ \text{s.t.} \quad & \boldsymbol{\omega} \in \mathcal{P}(\boldsymbol{\omega}). \end{aligned}$$

This problem is very challenging to solve in general because the set $\mathcal{U}(\boldsymbol{R})$ is typically a very large discrete set. Even if we know the weight vector, solving for the worst realization of the ranking would be a combinatorial problem and could be very difficult. We will show that, for a large class of the objective function f and the constraint set $\mathcal{P}$, we can solve this robust ranking problem efficiently by using the constraint generation algorithm and network flows.

1.3. Paper structure

This paper aims to solve the robust ranking problem in general. We will provide the general conditions on the objective

function and the constraint set for the algorithm to work. However, since the computation performance depends on the specific objective function $f(\boldsymbol{\omega},\boldsymbol{R})$, we will provide the algorithm for the general objective function, but demonstrate its usage, its intuitions, as well as its performance, through the specific objective functions in the robust portfolio optimization problem. We will show how to apply the constraint generation method to solve the generic robust ranking problem in Section 2.2. This method iteratively solves the relaxed robust ranking problem with a smaller ranking uncertainty set, and then adds the worst possible ranking corresponding to the relaxed optimal weights to the relaxed uncertainty set. The algorithm stops when the relaxed uncertainty set is stable.

For this algorithm to work, we need to find efficient methods for generating additional constraints and for solving the relaxed problem. The constraint generation problem is analyzed in Section 2.3, and the relaxed problem in Section 2.4. We extend our robust ranking model to control the degree of conservatism in Section 2.5 and to allow group ranking in Section 2.6. We show how to apply our robust ranking model to portfolio optimization in Section 3.

We present numerical results in Section 4 with the overall computation time and the number of iterations in Section 4.1. In Section 4.2, we present an empirical study where we apply our generic robust ranking model to portfolio optimization with the universe of stocks in the Dow Jones Industrial Average Index. The ranking uncertainty set is obtained from the post-earnings-announcement drifts as shown in Section 4.2.1. We show the procedure for running an out-of-sample test in Section 4.2.3 and the empirical results in Section 4.2.4. We conclude our paper in Section 5.

1.4. Contributions

- We develop a generic robust ranking model that is applicable to decision making situations where objects' ranking contains uncertainty. Our model is distinguished from other robust optimization models in its discrete uncertainty set. The resulting robust ranking problem becomes a mixed-integer minimax problem and is often very challenging to solve.
- We use the constraint generation method to solve the problem. Although the general framework for the constraint generation algorithm is not new, the key to its success lies in the capability to generate violating constraints efficiently. We achieve this by showing that the constraint generation problem can be transformed into a network flow problem and can thus be solved very efficiently.
- We apply our ranking model into portfolio optimization and perform empirical tests. The ranking uncertainty is obtained based on stocks' post-earnings-announcement drifts. We provide computational results to demonstrate the algorithm performance.

2. Solving the robust ranking problem

2.1. Notations

In general, we use bold fonts for vectors and normal fonts for scalars. Specifically, we use the following notations:

- $n \in \mathbb{Z}$: The number of objects (e.g. the number of assets in the robust portfolio optimization example).
- $\Pi(1,\ldots,n)$: The set of all the permutations of the set $(1,\ldots,n)$.
- $\boldsymbol{R} \in \Pi(1,\ldots,n)$: A ranking vector.
- $R_j \in [1,\ldots,n]$: The ranking of object j for all $j \in [1,\ldots,n]$.
- $\mathcal{U}(\boldsymbol{R})$: The uncertainty set of the ranking $\boldsymbol{R}: \mathcal{U}(\boldsymbol{R}) = \{\boldsymbol{R} \in \Pi(1,\ldots,n) | R_i \in Q_i, \forall i \in \{1,\ldots,n\}\}$, where Q_i is a subset of $\{1,\ldots,n\}$.
- $\mathcal{S}^{(k)} \subset \mathcal{U}(\boldsymbol{R})$: Relaxed uncertainty set used in the algorithm.
- $\boldsymbol{R}^{(k)} \in \Pi(1,\ldots,n)$: Ranking vectors, which are produced through the algorithm.
- $\mathbf{r} \in \mathbb{R}^n$: Expected returns of the assets.
- $r_i \in \mathbb{R}$: The expected return of asset i.
- $\mathcal{U}(\mathbf{r})$: The uncertainty set of the assets' returns.
- $\boldsymbol{\omega} \in \mathbb{R}^n$: A vector of weights.
- $\omega_i \in \mathbb{R}$: The weight allocated for object i.
- $\mathcal{P}(\boldsymbol{\omega}) \in \mathbb{R}^n$: Feasible space for the weight vector $\boldsymbol{\omega}$.
- $f(\boldsymbol{\omega},\boldsymbol{R})$: Objective function.
- $\delta(\boldsymbol{\omega})$: Worst objective value given $\boldsymbol{\omega}$.
- $\boldsymbol{\omega}^{(k)} \in \mathbb{R}^n$: Weight vectors produced through the algorithm.
- $\boldsymbol{e}$: A unit vector of size n of all 1.
- $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$: The covariance matrix of the assets.

The inputs to the generic robust ranking problem **GRR** include the number of objects n, the uncertainty set $\mathcal{U}(\boldsymbol{R})$, the objective function $f(\boldsymbol{\omega},\boldsymbol{R})$ and the feasible space $\mathcal{P}(\boldsymbol{\omega})$. The output is the robust weight vector $\boldsymbol{\omega}$.

2.2. The constraint generation algorithm

Let $\delta(\boldsymbol{\omega}) = \min_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})} f(\boldsymbol{\omega},\boldsymbol{R})$. The robust ranking model 1 can be rewritten as follows:

$$\begin{aligned} \max_{\boldsymbol{\omega},\delta} \quad & \delta \\ \text{s.t.} \quad & \delta \leqslant f(\boldsymbol{\omega},\boldsymbol{R}) \forall \boldsymbol{R} \in \mathcal{U}(\boldsymbol{R}) \\ & \boldsymbol{\omega} \in \mathcal{P}(\boldsymbol{\omega}) \end{aligned}$$

This new model has a simpler objective function compared to the original robust ranking formulation. However, the set of constraints is much larger. This reformulated model motivates us to apply the constraint generation method. In this method, we relax the robust ranking problem with a smaller set of constraints $\mathcal{S}$ and solve for the relaxed optimal solution. We then check if all the constraints are satisfied. This is done by solving a sub-problem of finding the worst realization of the ranking that corresponds to the relaxed optimal solution and checking if the worst ranking is already in the relaxed ranking list. If not, we add the violating constraints to the relaxed constraint set. This process is done until no more constraints are violated. The formal procedure of the constraint generation method is shown in Algorithm 1:

Algorithm 1. The Constraint Generation Method for the Robust Ranking Problem

- **Initialization step**: Find an initial weight vector $\omega^{(0)}$, set the relaxed uncertainty set $\mathcal{S}^{(0)} \equiv \emptyset$ and set $k = 0$.
- **Iterative steps**:
  - **a.** Solve the **constraint generation problem**:
    $$\boldsymbol{R}^{worst} = \text{argmin}_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})}[f(\boldsymbol{\omega}^{(k)},\boldsymbol{R})]$$
  - **b.** If $\boldsymbol{R}^{worst} \in \mathcal{S}^{(k)}$, terminate the algorithm. Otherwise, add R^{worst} to the relaxed uncertainty set, i.e. $\mathcal{S}^{(k+1)} = \{\mathcal{S}^{(k)}, \boldsymbol{R}^{worst}\}$.
  - **c.** Set $k = k + 1$ and solve the **relaxed problem**:
    $$\begin{aligned} \boldsymbol{\omega}^{(k)} &= \text{argmax}_{\boldsymbol{\omega}\in\mathcal{P}(\boldsymbol{\omega})}[\min_{\boldsymbol{R}\in\mathcal{S}^{(k)}} f(\boldsymbol{\omega},\boldsymbol{R})], \delta^{(k)} \\ &= \max_{\boldsymbol{\omega}\in\mathcal{P}(\boldsymbol{\omega})}[\min_{\boldsymbol{R}\in\mathcal{S}^{(k)}} f(\boldsymbol{\omega},\boldsymbol{R})] \end{aligned}$$
  - **d.** Go back to step **a**.

Notice that once the optimal solution of the relaxed problem meets all the constraints of the original problem, that relaxed optimal solution is an optimal solution of the original problem. This occurs when $\delta^{(k)} = \min_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})}[f(\boldsymbol{\omega}^{(k)}, \boldsymbol{R})]$. Formally, we can express the stoping condition as stated in the following Proposition:

Proposition 1. *$\omega^{(k)}$ is an optimal solution of the original robust ranking problem if and only if the relaxed problem produces a ranking that has already been in the relaxed uncertainty set, i.e. $\boldsymbol{R}^{worst} \in \mathcal{S}^{(k)}$ (which is also equivalent to $\mathcal{S}^{(k)} \equiv \mathcal{S}^{(k+1)}$).*

Proof. Let ω_{opt} be the optimal solution of the original problem. We have:

$$\begin{aligned}
\min_{\boldsymbol{R}\in\mathcal{S}^{(k)}} f(\boldsymbol{\omega}^{(k)}, \boldsymbol{R}) &\geqslant \min_{\boldsymbol{R}\in\mathcal{S}^{(k)}} f(\boldsymbol{\omega}_{opt}, \boldsymbol{R}) && (1)\\
&\geqslant \min_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})} f(\boldsymbol{\omega}_{opt}, \boldsymbol{R}) && (2)\\
&\geqslant \min_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})} f(\boldsymbol{\omega}^{(k)}, \boldsymbol{R}) && (3)\\
&= f(\boldsymbol{\omega}^{(k)}, \boldsymbol{R}^{worst}) && (4)\\
&\geqslant \min_{\boldsymbol{R}\in\mathcal{S}^{(k+1)}} f(\boldsymbol{\omega}^{(k)}, \boldsymbol{R}) && (5)
\end{aligned}$$

Here, (1) is due to the fact that $\omega^{(k)}$ is an optimal solution of the relaxed robust ranking problem, (2) is because $\mathcal{S}^{(k)} \subset \mathcal{U}(\boldsymbol{R})$, (3) is because ω_{opt} is an optimal solution of the original robust ranking problem, (4) comes directly from our definition of R^{worst}. Finally, (5) is because $\boldsymbol{R}^{worst} \in \mathcal{S}^{(k+1)}$. Notice that if we extract only the first and the last terms in this chain of inequalities, we obtain:

$$\min_{\boldsymbol{R}\in\mathcal{S}^{(k)}} f(\boldsymbol{\omega}^{(k)}, \boldsymbol{R}) \geqslant \min_{\boldsymbol{R}\in\mathcal{S}^{(k+1)}} f(\boldsymbol{\omega}^{(k)}, \boldsymbol{R})$$

Thus, if $\mathcal{S}^{(k)} \equiv \mathcal{S}^{(k+1)}$ then all the inequalities, including (2), must hold and hence $\omega^{(k)}$ is an optimal solution of the original problem. On the other hand, if $\omega^{(k)}$ is an optimal solution, then it is straightforward to show $\boldsymbol{R}^{worst} \in \mathcal{S}^{(k)}$. □

Proposition 1 shows that the constraint generation algorithm never enters a loop of reintroducing worst rankings before the final iteration. In addition, since the uncertainty set contains a finite number of possible ranking $\boldsymbol{R}$, the constraint generation algorithm will terminate at an optimal solution. This is independent on the starting weight vector $\boldsymbol{\omega}^{(0)}$. However, we need to make sure that we can solve the subproblems for generating constraints and for solving the relaxed problems efficiently. In the next two Subsections, we will show the general conditions for these two problems to be computationally tractable.

2.3. Solving the constraint generation problem

The constraint generation problem is often difficult to solve because of its combinatorial structure. In order to solve this problem efficiently, we need to make an assumption on the separability of the objective function.

Assumption 1. We assume that:

(a) the objective function is separable in R, i.e. $f(\boldsymbol{\omega}, \boldsymbol{R}) = \sum_{i=1}^{n} g_i(\boldsymbol{\omega}, R_i)$, and
(b) the uncertainty set has the permutation property, i.e. each object is assigned with one and only one rank and each rank is given to one and only one asset.

We will show in Section 3 that the separability property is indeed satisfied in many models for portfolio optimization. In other applications such as deciding the ordering of the websites to be displayed by search engines or ranking in sports, the objective function can be expressed in the form $f(\boldsymbol{\omega}, \boldsymbol{R}) = \sum_{i=1}^{n} g_i(\omega_i, R_i)$ to measure the discrepancy between the decision variable $\boldsymbol{\omega}$ and the uncertain ranking $\boldsymbol{R}$. In these cases, the separability assumption holds.

It is very interesting to notice that once Assumption 1 holds, we can transform the constraint generation problem into a transportation problem as we will show in Proposition 2. The implication of this proposition is that, for separable objective functions, the constraints can be generated efficiently for the constraint generation algorithm. Notice that, the restriction on the permutation property of the uncertainty set is only used for Proposition 2 and can be extended for the case of group ranking where many objects can obtain the same rank in Section 2.6.

Proposition 2. *The constraint generation problem is equivalent to a network flow problem if Assumption 1 holds.*

Proof. Consider a transportation network that has n source and n sink nodes. This network has inflows of (+1) in each source node and outflows of (−1) in each sink node. The cost between source i and sink j is given by $g_i(\omega,j)$ if $j \in Q_i$, where Q_i is the list of possible ranks that object i can have, and is infinity otherwise. Under the permutation property in Assumption 1, each asset is paired with one and only one rank and vise versa. Therefore, each feasible solution of the transportation problem can be viewed as an assignment of assets into ranks. For example, if there is a flow between source i and sink j, we can then assign object i to rank $R_i = j$. The total cost of the transportation network is $\sum_{i=1}^{n} g_i(\boldsymbol{\omega}, R_i)$ which is equal to $f(\omega,R)$. Thus, by minimizing the total cost of the transportation network, which can be done very efficiently (see Ravindra et al. [25]), we solve the constraint generation problem. □

Fig. 1 shows the transportation network for an example of 3 assets with the ranking uncertainty set $\mathcal{U}(\boldsymbol{R})$ expressed as follows:

$$\mathcal{U}(\boldsymbol{R}) = \{\boldsymbol{R} \in \Pi(1,2,3) | R_1 \in \{1,2\}, R_2 \in \{1,2,3\}, R_3 \in \{1,2,3\}\}$$

From Proposition 2, we can show the complexity of the robust ranking problem as stated in the following proposition:

Proposition 3. *The robust ranking problem is solvable in polynomial time if (a) the constraint set $\mathcal{P}(\boldsymbol{\omega})$ is a polyhedron, and (b) $f(\omega,R)$ is linear in ω and is separable in R.*

Proof. Under the assumption that the constraint set $\mathcal{P}(\boldsymbol{\omega})$ is a polyhedron, and $f(\omega,R)$ is linear in ω and is separable in R, the

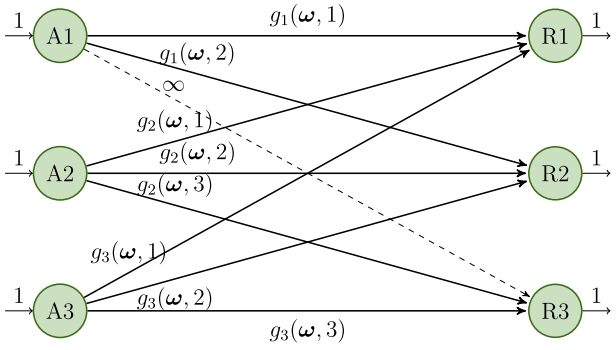

Fig. 1. The constraint generation problem is reformulated as a transportation problem. The path connecting source node i to sink node j implies that asset i is assigned to rank j. Suppose the uncertainty set is: $\mathcal{U}(\boldsymbol{R}) = \{\boldsymbol{R} \in \Pi(1,2,3) | R_1 \in \{1,2\}, R_2 \in \{1,2,3\}, R_3 \in \{1,2,3\}\}$. This means asset 1 has the first or second ranking while assets 2 and 3 have all the possible ranking. The cost of assigning asset i to rank j is $g_i(\boldsymbol{\omega},j)$ if j is a candidate rank for asset i. Since asset 1 cannot be in the third rank, the cost of the flow from asset 1 to rank 3 is set to infinite. The total cost of the network flow problem is equal to the objective function $f(\boldsymbol{\omega}, \boldsymbol{R}) = \sum_{i=1}^{n} g_i(\boldsymbol{\omega}, R_i)$.

reformulated robust ranking model is a linear programming problem with an exponentially large number of constraints. From Proposition 1, identifying a violating constraint for a given ω can be done in polynomial time (i.e. by solving an assignment problem). This means we have an 'oracle-polynomial time' to solve the separation problem. Therefore, the robust ranking problem can be solved in polynomial time using the ellipsoid method (see Theorem 6.4.9 in Grotschel et al. [18]). □

Proposition 3 provides a nice theoretical result on the complexity of the problem. However, since ellipsoid algorithm does not perform well in practice, we use the constraint generation algorithm described in Algorithm 2.2. Although it is possible to modify this algorithm using the bundle method such that theoretical convergence properties can be derived (see Briant et al. [11]), this technique is quite complicated. In addition, our numerical results in Section 4.2.4 show that the current simple version of the constraint generation method can converge in reasonable time.

2.4. Solving the relaxed problem

The relaxed problem is equivalent to:

$$\begin{aligned} \max_{\boldsymbol{\omega},\delta} \quad & \delta \\ \text{s.t.} \quad & \delta \leqslant f(\boldsymbol{\omega},\boldsymbol{R}), \forall \boldsymbol{R} \in \mathcal{S} \\ & \boldsymbol{\omega} \in \mathcal{P}(\boldsymbol{\omega}) \end{aligned}$$

For a small set $\mathcal{S}$, this problem is computationally tractable for many classes of objective function $f(\boldsymbol{\omega},\boldsymbol{R})$ and constraint set $\mathcal{P}(\boldsymbol{\omega})$. For example, if $f(\boldsymbol{\omega},\boldsymbol{R})$ is a concave function and $\mathcal{P}(\boldsymbol{\omega})$ is a convex set in $\boldsymbol{\omega}$, the relaxed problem is a convex optimization problem. Notice that even when the objective function is non-convex, it might be still possible to transform the relaxed problem into a convex optimization problem. For example, we will show in Section 3.2 that, although the Sharpe ratio objective function in the robust portfolio optimization problem is non-convex, we can exploit its homogeneous property and transform the relaxed problem into a quadratic programming problem which can be solved efficiently.

2.5. Extension to control conservative levels

One of the main criticisms for robust optimization is its overconservatism at its optimal solution. The robust optimal solution usually corresponds to the cases where the parameters take extreme values in their boundary, instead of corresponding to some nominal ranking $\overline{\boldsymbol{R}}$ that we mostly expect. We propose a model where we incorporate the deviations of the parameters from their nominal values into the objective function itself. This protects the regions of parameters that are closer to the nominal value more than those that are further off.

Model 2: Robust Ranking Model with Non-Uniform Uncertainty Set

$$\begin{aligned} \max_{\boldsymbol{\omega}} \quad & \left[\min_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})} [f(\boldsymbol{\omega},\boldsymbol{R}) + \gamma|\boldsymbol{R}-\overline{\boldsymbol{R}}|_{L1}]\right] \\ \text{s.t.} \quad & \boldsymbol{\omega} \in \mathcal{P}(\boldsymbol{\omega}). \end{aligned}$$

In this new model, we penalize the objective function if the realized ranking R_i is different from its nominal ranking $\overline{R}_i$. Here, $\gamma \geqslant 0$ and $\gamma|\boldsymbol{R}-\overline{\boldsymbol{R}}|_{L1}$ represents the belief about how far the actual ranking could go from the nominal ranking (here, $|\cdot|_{L1}$ denotes the L1 norm). If γ is small, then the actual ranking could be anywhere in the uncertainty set. For example, with $\gamma = 0$, we have the original robust ranking formulation. If γ is large, we have a strong belief that the actual ranking will be around the nominal ranking and we want the robust model to protect that region more. In other words, γ can be viewed as the 'concentration' parameter to represent how 'dense' the uncertainty set $\mathcal{U}(\boldsymbol{R})$ is surrounding the nominal ranking $\overline{\boldsymbol{R}}$.

Proposition 4. *The constraint generation problem arisen in Algorithm 1 for the "robust ranking model with non-uniform uncertainty set" shown in Model 2 is equivalent to a network flow problem if Assumption 1 holds.*

Proof. Notice that the new objective function has been changed to:

$$f'(\boldsymbol{\omega},\boldsymbol{R}) = f(\boldsymbol{\omega},\boldsymbol{R}) + \gamma|\boldsymbol{R}-\overline{\boldsymbol{R}}|_{L1} = \sum_{j=1}^{n} g'_j(\boldsymbol{\omega},R_j)$$

where $g'_j(\boldsymbol{\omega},R_j) = g_j(\boldsymbol{\omega},R_j) + \gamma|R_j - \overline{R}_j|$. Since the new objective function is still separable in R, we can use the same method shown in Proposition 2 to transform the constraint generation problem into a network flow model by changing the cost between source i to sink j to $g'_i(\boldsymbol{\omega},j)$. □

The constraint generation algorithm is still the same. However, we need to modify the transportation model to adjust this change in the objective function. Consider a simple case of $n = 3$ and suppose:

$$\mathcal{U}(\boldsymbol{R}) = \{\boldsymbol{R} \in \Pi(1,2,3) | R_1 \in \{1,2\}, R_2 \in \{1,2,3\}, R_3 \in \{1,2,3\}\}$$
$$\overline{\boldsymbol{R}} = \{1,2,3\}$$

Fig. 2 shows the modified network in detail. For example, if the actual rank of object 3 is 1 (or 2) instead of its nominal value 3, we would include a penalty of 2γ (or γ) into the objective function. Hence, the cost of assigning asset 3 to rank 1 (or 2) has been changed from $g_3(\boldsymbol{\omega},1)$ to $g_3(\boldsymbol{\omega},1) + 2\gamma$ (or $g_3(\boldsymbol{\omega},2) + \gamma$).

Remark 1. The choice of γ can be done through cross-validation. The set of possible candidates for γ can be found by comparing the relative values of the optimal value $f(\omega^*,R^{worst})$ of the robust model when $\gamma = 0$ and the corresponding deviation $|\boldsymbol{R}^{worst} - \overline{\boldsymbol{R}}|_{L1}$. Then we can set:

$$\gamma = \alpha * \frac{f(\boldsymbol{\omega}^*,\boldsymbol{R}^{worst})}{|\boldsymbol{R}^{worst} - \overline{\boldsymbol{R}}|_{L1}}$$

for some choice of α. With $\alpha = 1$, we treat the risk of being at the boundary of the uncertainty set and the risk of changing from the nominal value equally. Letting $\alpha < 1$ would mean we are more conservative. Letting $\alpha > 1$ would mean we believe more in the nominal value.

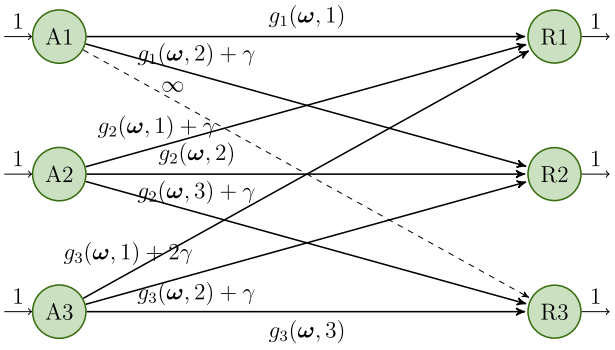

Fig. 2. The network flow problem when the conservative level is incorporated.

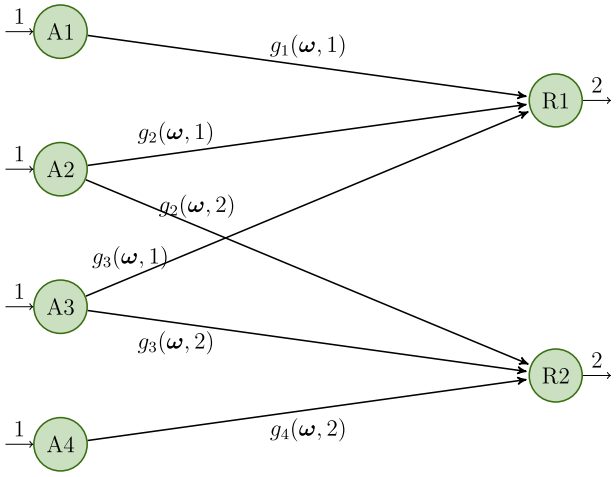

Fig. 3. The network flow problem when allowing group ranking in the uncertainty set.

2.6. Extension to allow group ranking

So far, we have assumed there is a complete ranking for all the objects. However, in many situations, objects are ranked into groups or tiers where there is no ranking information between those objects within the same group. For example, instead of having n different ranks for n assets, we can use only 10 ranks and divide assets into deciles. We can easily extend the robust ranking problem to allow robust group ranking. The uncertainty in this case comes from the fact that some assets might not be in their predicted tiers. Let K be the number of tiers. The algorithm for solving this problem is still unchanged. However, we need to modify the transportation model to be a network flow model with n source and K sink nodes. Let us take an example with $n = 4$ assets and $K = 2$ tiers. Asset 1 is predicted to be in the first tier. Assets 2 and 3 are either in the first or the second tier. Asset 4 is in the second tier. The corresponding network flow model is modified according to Fig. 3.

We have presented the general robust ranking model, its computational method using the constraint generation algorithm and its extensions. In the next section, we will show how to apply our robust ranking model to portfolio optimization.

3. Robust ranking models for portfolio optimization

In this Section, we will apply our generic robust ranking model to portfolio optimization. We will show different choices of the objective functions and the constraint sets. We will present their computational performance in Section 4.1 and their empirical results in Section 4.2. Consider a portfolio optimization problem where an investor forms a portfolio $\boldsymbol{\omega}$ out of n assets based on their ranking information which belongs to an uncertainty set $\mathcal{U}(\boldsymbol{R})$. The ranking information is obtained by using an accounting anomaly on the post-earnings-announcement drift (see Ball and Brown [3]). The choice of the objective function f and the constraint set $\mathcal{P}(\boldsymbol{\omega})$ could vary from investor to investor. We present two possible models in the next Subsections.

3.1. Model I: Robust max weighted ranking with nonnegative weight constraint

$$\begin{aligned} \max_{\boldsymbol{\omega}} \quad & \left[\min_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})} \boldsymbol{\omega}^t\boldsymbol{R}\right] \\ \text{s.t.} \quad & \boldsymbol{\omega}^t\boldsymbol{e} = 1 \\ & \boldsymbol{\omega} \geqslant 0. \end{aligned}$$

In this model, we want to find the optimal portfolio weight $\boldsymbol{\omega}$ to maximize the weighted average ranking $\boldsymbol{\omega}^t\boldsymbol{R}$ for the worst realization of the ranking $\boldsymbol{R}$ that lies in the uncertainty set $\mathcal{U}(\boldsymbol{R})$. The constraints $\boldsymbol{\omega}^t\boldsymbol{e} = 1$ and $\boldsymbol{\omega} \geqslant 0$ are for the normalization and short-selling restriction respectively. In this case, the objective function f is separable in ranking and hence we can apply the constraint generation method.

$$f(\boldsymbol{\omega}, \boldsymbol{R}) = \boldsymbol{\omega}^t\boldsymbol{R} = \sum_{j=1}^{n} g_j(\boldsymbol{\omega}, R_j)$$

where $g_j(\boldsymbol{\omega}, R_j) = \omega_j R_j$.

3.2. Model II: Robust max weighted ranking with risk constraint

$$\begin{aligned} \max_{\boldsymbol{\omega}} \quad & \left[\min_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})} \boldsymbol{\omega}^t\boldsymbol{R}\right] \\ s.t. \quad & \boldsymbol{\omega}^t\boldsymbol{\Sigma}\boldsymbol{\omega} \leqslant 1. \end{aligned}$$

In this model, we want to find the optimal portfolio weight $\boldsymbol{\omega}$ to maximize the average ranking for the worst realization of the ranking $\boldsymbol{R}$ that lies in the uncertainty set $\mathcal{U}(\boldsymbol{R})$. The constraint set $\boldsymbol{\omega}^t\boldsymbol{\Sigma}\boldsymbol{\omega} \leqslant 1$ allows the total portfolio risk to be bounded above. This model is very similar to a mean–variance portfolio optimization model, except that we use the ranking $\boldsymbol{R}$ in the place of the expected returns $\boldsymbol{\mu}$. This robust model is a hybrid between a nonparametric model using ranking, and a parametric model using the covariance matrix. The main criticism for using the mean–variance portfolio optimization is in the estimation error from the expected returns but not from the covariance matrix. Hence, we could retain the good features from the mean–covariance model by adding the risk constraint to the robust ranking model.

It is interesting to provide a reformulation of Model II as stated in the following Proposition.

Proposition 5. *The following two models (Model II and Max-Sharpe) are equivalent:*

Model II

$$\begin{aligned} \max_{\boldsymbol{\omega}} \quad & \left[\min_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})} \boldsymbol{\omega}^t\boldsymbol{R}\right] \\ \text{s.t.} \quad & \boldsymbol{\omega}^t\boldsymbol{\Sigma}\boldsymbol{\omega} \leqslant 1 \end{aligned}$$

Max-Sharpe

$$\begin{aligned} \max_{\boldsymbol{\omega}} \quad & \left[\min_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})} \frac{\boldsymbol{\omega}^t\boldsymbol{R}}{\sqrt{\boldsymbol{\omega}^t\boldsymbol{\Sigma}\boldsymbol{\omega}}}\right] \\ \text{s.t.} \quad & \boldsymbol{\omega}^t\boldsymbol{e} = 1 \end{aligned}$$

in the sense that optimal solution from one model can be derived from the other.

The detailed proof of Proposition 5 is in Appendix A.2. The reformulated model looks like a max Sharpe version. In fact, if we could transform a ranking into the equivalent returns, we could have a mapping from the ranking uncertainty set into a discrete expected return uncertainty set, then Model II would be exactly the robust max Sharpe model. For example, suppose we decide to use the Almgren and Chriss method in [2] to transform the complete ranking $R_1 \geqslant R_2 \geqslant, \ldots, \geqslant R_n$ into the equivalent returns c, where $c_1 \geqslant c_2 \geqslant, \ldots, \geqslant c_n$. Then, for any asset i with ranking information $l_i \leqslant R_i \leqslant u_i$, its return r_i belongs to the set $\{c_{l_i}, c_{l_i+1}, \ldots, c_{u_i}\}$. Thus, the uncertainty set for the returns is constructed as follows:

$$\mathcal{U}(\mathbf{r}) = \{\mathbf{r} | r_i \in \{c_{l_i}, c_{l_i+1}, \ldots, c_{u_i}\}, \forall i \in [1, \ldots, n]\}$$

The robust max Sharpe portfolio optimization problem can be formulated as:

$$\max_{\omega} \quad \min_{\mathbf{r}\in\mathcal{U}(\mathbf{r})} \frac{\sum_{i=1}^{n}\omega_i r_i}{\sqrt{\boldsymbol{\omega}^t\boldsymbol{\Sigma}\boldsymbol{\omega}}}$$
$$\text{s.t.} \quad \boldsymbol{\omega}^t\boldsymbol{e} = 1$$

We can then apply the same constraint generation method to solve this model. In addition to the max Sharpe objective function, it is also possible to do robust ranking for many other types of objective functions in the portfolio optimization problem such as a quadratic utility function.

4. Numerical results

4.1. Computational time

Table 1 shows the computational time to solve the robust portfolio optimization problem using Model 2, the average computational times to solve the relaxed problem and the constraint generation problem (all in seconds), and the number of iterations that the algorithm takes to converge to the optimal solution. We can see that the algorithm finds the optimal solutions very fast. For example, with $N = 100$ and $k = 20$, the algorithm converges in 264 iterations, even though the size of the uncertainty set is very large (up to the order of $k^N = 20^{100}$ possible ranking vectors in the uncertainty set).

We also record the computational time for solving the relaxed problem and the constraint generation problem through iterations in Fig. 4. In general, we can see a trend of increasing computational time for solving the relaxed problem because the number of constraints increases by one through each iteration. However, it is interesting to notice that the constraint generation problem takes about the same time in every iteration. This is because the transportation model does not change through iterations.

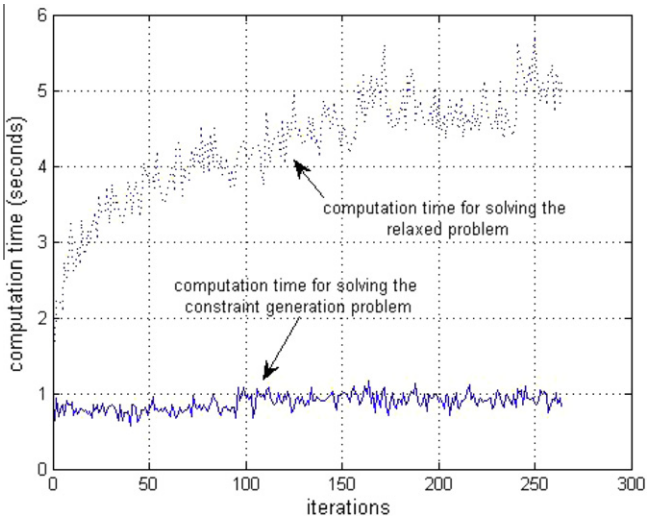

Fig. 4. Computational time for solving the relaxed problem and the constraint generation problem.

4.2. Empirical study

We apply robust ranking into portfolio optimization where the ranking is obtained from the standardized earning surprise. We first test the common belief about post-earnings-announcement drifts. We then compare the performance of the robust models with their non-robust counterparts. We test these models with stocks in the Dow Jones Industrial Average Index during the period from 2000 to 2007.

4.2.1. Post-earnings-announcement drifts

Ball and Brown [3] were the first to document the post-earnings-announcement drift effect. The authors define the standardized earning surprise as follows:

$$SUE_t = \frac{E_{a,t} - E_{e,t}}{\sigma(E_a - E_e)}$$

where $E_{a,t}$ and $E_{e,t}$ are the actual earning and the average analysts' estimates on earning for period t, and $\sigma(E_a - E_e)$ is the standard deviation of the differences between the actual and the estimated earnings. The authors show that a simple portfolio with a long side of the highest *SUE* assets and a short side of the smallest *SUE* assets would earn significant returns with high Sharpe ratio. This result provides us a method to rank assets according to their SUE_t as we will show in Section 4.2.3.

4.2.2. Data collection

- *Stock Universe*: We construct our portfolio from stocks in the Dow Jones Industrial Average Index. We avoid possible errors with the database as much as possible by (a) keeping only those stocks with consistent earning records (every March 31st, June 30th, September 30th and December 31st) and (b) removing those stocks whose issuing companies have merged, split, or changed their names. This results in a set of 14 stocks in the period from 2000 to 2007.
- *Earning Information*: The estimated earnings and actual earnings are retrieved from the IBES database from the Wharton School of Business (WRDS).
- *Stocks' Returns*: The stocks' returns are retrieved from the CRSP database from the Wharton School of Business (WRDS).
- *Time Frame*: Our data starts from 2000. Since the CRSP and IBES only provide data up to 2007, our test sample ends in 2007.

4.2.3. Trading strategy implementation

Our trading time periods start right after the quarterly earning announcement, and we hold our trading position until the end of that quarter before a new portfolio is formed. Suppose we are at

Table 1
Computational time (in seconds) for running the constraint generation algorithm with different problem sizes. We used Matlab to run simulations. We used Sedumi for solving the relaxed problem and use Matlog for solving the constraint generation problem. All the computation results were produced by an Intel(R) Xeon(R) CPU, 2.67 gigahertz and 12 gigabyte of RAM computer. Notice that the solutions found were optimal. If we allow ϵ-violation on the constraint generation problem, the algorithm is expected to run much faster.

Number of assets	10	20	20	50	75	100	100
$u_i - l_i$	4	4	10	10	10	10	20
Number of iterations	8	12	39	66	65	51	264
Average time solving the Constraint generation problem	0.004	0.031	0.039	0.280	0.469	0.75	0.88
Average time solving the relaxed problem	0.218	0.234	0.241	1.352	2.182	3.15	4.27
Total time (in seconds)	1.794	3.214	10.95	107.1	172.8	199	1360

time t, that is the beginning of a quarter, we look back for 1 year and find the Dow Jones Industrial Average constituents that are in the index for the entire year (this should provide a subset of all the constituents at time t). We find the earning estimates by analysts for the last quarter and average them to obtain $E_{e,t}$. We also find the actual earning announced by the stock issuing companies for the last quarter to obtain $E_{a,t}$. We then find the standardized earning surprise as $SUE_t = \frac{E_{a,t}-E_{e,t}}{\sigma(E_a-E_e)}$, where $\sigma(E_a - E_e)$ is the historical standard deviation of the discrepancy between the estimated and actual earnings. We rank SUE_t of all the stocks selected and obtain a ranking $\overline{R} = \{\overline{R}_1, \overline{R}_2, \ldots, \overline{R}_n\}$. The uncertainty set is set equal to: $U(R) = R|(\overline{R}_i - l_i) \leqslant R_i \leqslant (\overline{R}_i + u_i)$ for different choices of widths l and u. We run the robust models to obtain the robust portfolio weights w_t. Since we hold this portfolio for the next quarter, the portfolio return for quarter $t+1$ is $w_t^t r_{t+1}$ where r_{t+1} is the return of the assets in quarter $(t+1)$. Notice that this empirical testing method provides us with the out-of-sample performance of our robust ranking portfolios, since at any point of time, we only use the historical data to make investment decisions.

4.2.4. Empirical results

First, we test whether the post-announcement earning has any drift to stocks' returns. This is done by simply comparing a portfolio of the highest ranked SUE stocks with another portfolio of the lowest ranked SUE stocks and the equal weighted portfolio. (We include the equal weighted portfolio here because it represents a naive investment strategy without any information on the ranking. In addition, DeMiguel et al. [14] have shown that the equal weighted portfolio outperformed the sample based mean variance portfolio and many other models). This comparison is shown in Table 2 and Fig. 5. We can see the highest ranked portfolio outperforms the equal weighted and the lowest ranked portfolios (in terms of the expected returns and the Sharpe ratios). This implies that the post-announcement earning does have drift on the stock returns in this case.

Table 2
Risk and return characteristics for different strategies. Rows 2, 3 and 4 show that the high SUE portfolio indeed outperforms the equal weighted portfolio and the low SUE portfolio. This implies that the post-earning announcement does contain drift to the stocks' returns.

Model	Mean returns	Standard deviation	Sharpe ratio
Equal weighted portfolio	0.0795	0.1797	0.4424
Lowest SUE portfolio	0.0413	0.2033	0.2030
Highest SUE portfolio	0.1177	0.1829	0.6439

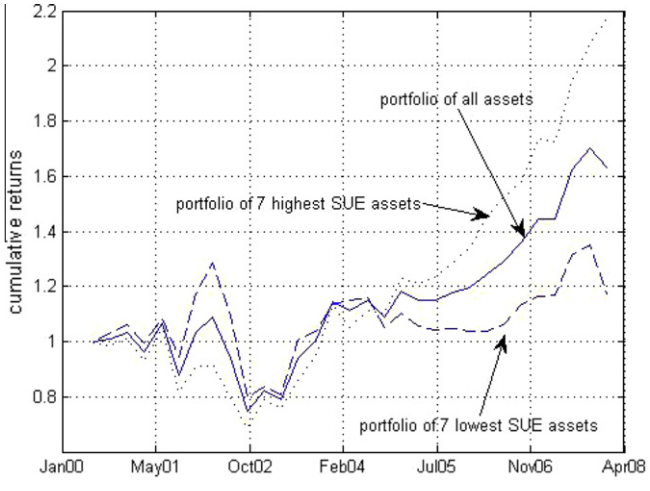

Fig. 5. This figure shows that post-earning announcement does indeed have valuable information. The blue curve (the middle one) is for the equal weighted portfolio of 14 assets that are extracted from the Dow Jones Industrial Average Index. The black curve (the lowest one) is for the equal weighted portfolio of the lowest SUE assets. The red curve (the top curve) is for the equal weighted portfolio of the highest SUE assets.

Next, we compare the performances between non-robust portfolios and robust portfolios in Table 3. The third, fourth and the fifth rows show the comparison between a non-robust portfolio using the nominal ranking and two robust portfolios using Model I, shown in Section 3.1. The uncertainty set for the fourth row portfolio is set as $U(R) = R|(\overline{R}_i - 1) \leqslant R_i \leqslant (\overline{R}_i + 1)$ while that of the fifth row portfolio is set as $U(R) = R|(\overline{R}_i - 2) \leqslant R_i \leqslant (\overline{R}_i + 2)$. Rows 6, 7 and 8 show the comparison between a non-robust portfolio and two robust portfolios using Model II shown in Section 3.2. The uncertainty sets for the two robust portfolios in rows 7 and 8 are constructed in the same way with $U(R) = R|(\overline{R}_i - w_i) \leqslant R_i \leqslant (\overline{R}_i + w_i)$ where the widths $w_i = 1$ and $w_i = 2$ correspondingly. In all cases, we can see that the robust portfolios have smaller risks (standard deviations) compared to their non-robust counterparts. The expected returns for these portfolios fluctuate around their non-robust counterparts. This is because our way of constructing the uncertainty sets was very simple and we did not use any additional information. Introducing the uncertainty sets would either a) reduce the value of the information we have in the nominal ranking or b) make the robust ranking portfolios less sensitive to estimation errors, or both. In fact, we can see both of these effects in this empirical example.

Finally, we test the performance of our robust models on varying the correctness of the uncertainty sets. We construct two portfolios: the first one with additional 'good information' and the second with additional 'bad information' reflected in the uncertainty sets. These uncertainty sets are constructed as follows: $U(R) = R|(\overline{R}_i - 1 + w_i) \leqslant R_i \leqslant (\overline{R}_i + 1 + w_i)$, where $w_i = \pm 1$. Let RT be the truth ranking of the stocks in the next period. The first portfolio is set closer to the truth ranking RT while the second portfolio is set further from RT compared to the nominal ranking $\overline{R}$. In the first portfolio, we set:

$$w_i^{(1)} = \begin{cases} 1, & \text{if } \overline{R}_i < RT_i \\ -1, & \text{if } \overline{R}_i > RT_i \\ 0, & \text{otherwise} \end{cases}$$

In the second portfolio, we set $w_i^{(2)} = -w_i^{(1)}$. Table 4 shows the comparison between a robust ranking portfolio without any additional information, i.e. $U(R) = R|(\overline{R}_i - 1) \leqslant R_i \leqslant (\overline{R}_i + 1)$, and the two portfolios we have just constructed. We can see clearly that having additional good information (shown in the fourth row) produces a better portfolio than the one without it, while having additional bad information (shown in the fifth row) could downgrade the portfolio. Hence, in practice, the uncertainty set should be set carefully.

Table 3
Risk and return characteristics for different strategies. Rows 3, 4 and 5 show that a non-robust and two robust portfolios using Model I. Rows 6, 7 and 8 show a non-robust portfolio and two robust portfolios using Model II. The uncertainty sets for the robust portfolios in rows 4, 5, 7 and 8 were constructed as $U(R) = R|(\overline{R}_i - w_i) \leqslant R_i \leqslant (\overline{R}_i + w_i)$ where the widths w range from 1 to 2. We can see that robust portfolios have smaller risk compared to their non-robust counterparts.

Model	Mean returns	Standard deviation	Sharpe ratio
Equal weighted portfolio	0.0795	0.1797	0.4424
Non-robust (I)	−0.0475	0.2859	−0.1661
Robust (I), ±1	0.0352	0.2162	0.1630
Robust (I), ±2	0.0906	0.1744	0.5194
Non-robust (II)	0.2221	0.3390	0.6553
Robust (II), ±1	0.2077	0.3190	0.6512
Robust (II), ±2	0.2227	0.3305	0.6739

Table 4
Risk and return characteristics for different strategies. Row 4 is for a robust portfolio whose uncertainty set is around the nominal ranking $\overline{R}$ but skewed toward the true ranking. Row 5 is for another robust portfolio whose uncertainty set is skewed in the opposite direction compared to the nominal ranking. We can see that the 'closer' the uncertainty sets to the actual ranking, the better the corresponding robust portfolio performs.

Model	Mean returns	Standard deviation	Sharpe ratio
Equal weighted portfolio	0.0795	0.1797	0.4424
Robust (II)	0.2077	0.3190	0.6512
Robust (II) + good information	0.3869	0.3785	1.0224
Robust (II) + bad information	0.1069	0.3089	0.3459

The confidence interval for each asset could be different from others. This is the point when beliefs from both quantitative analysts, fundamental analysts and managers should be combined together to construct a good uncertainty set.

5. Conclusion

In conclusion, we have presented a robust ranking model and applied the constraint generation method to solve the problem. The construction of the uncertainty set and the method for controlling the degree of conservatism were intuitive. We exploited the special characteristic of the ranking to ensure that constraints were generated efficiently through solving a network flow model. Our method worked for a large class of the objective functions. We showed computational performance of the algorithm for the specific case of robust portfolio optimization. We found the algorithm performed very fast, even for moderately large problem sizes. Our idea of using the constraint generation method is applicable in other robust models whose uncertainty sets are discrete, as long as we can find an efficient method to generate constraints. For empirical tests, we used the post-earnings-announcement drift to construct ranking uncertainty sets. We tested our robust models for stocks in the DJIA index in the period from 2000 to 2007, and found the robust portfolios produced smaller risk compared to their non-robust counterparts.

Appendix A

A.1. Sample data

Table A1 shows the list of stocks we used in our empirical test:

Table A1
List of assets in our sample.

Symbols	Company names
'AA'	Alcoa Incorporated
'BA'	Boeing Corporation
'C'	Citigroup Incorporated
'CAT'	Caterpillar Incorporated
'GE'	General Electric Company
'GM'	General Motors Corporation
'IBM'	International Business Machines
'INTC'	Intel Corporation
'JNJ'	Johnson & Johnson
'KO'	Coca-Cola Company
'MCD'	McDonalds Corporation
'MRK'	Merck & Company
'MSFT'	Microsoft Corporation
'UTX'	United Technologies

The remaining 16 assets in the DJIA index were not included for various reasons. The companies with symbols 'AXP', 'DD', 'DIS', 'MMM', 'PG', 'XOM' were not included because they missed the earning estimates for September 2000. The companies 'HD', 'HPQ', 'WMT' were not included because their earning announcements were on a different time scale to the rest, i.e. their announcements were at the end of April, July, October and January, while the rest at the end of March, June, September and December, according to the Wharton database. The companies: 'AXP', 'JPM', 'BAC', 'CVX', 'KFT', 'PFE', 'VZ' were not included because they have merged, split, or changed their name in the last 8 years which might impede the recording of their earning and returns on the Wharton database.

A.2. Proof of Proposition 5

Proposition 5. *The following two models (Model II and Max-Sharpe) are equivalent:*

Model II

$$\max_{\boldsymbol{\omega}} \quad \left[\min_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})} \boldsymbol{\omega}^t\boldsymbol{R}\right]$$
$$\text{s.t.} \quad \boldsymbol{\omega}^t\boldsymbol{\Sigma}\boldsymbol{\omega} \leqslant 1.$$

Max-Sharpe

$$\max_{\boldsymbol{\omega}} \quad \left[\min_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})} \frac{\boldsymbol{\omega}^t\boldsymbol{R}}{\sqrt{\boldsymbol{\omega}^t\boldsymbol{\Sigma}\boldsymbol{\omega}}}\right]$$
$$\text{s.t.} \quad \boldsymbol{\omega}^t\boldsymbol{e} = 1.$$

in the sense that optimal solution from one model can be derived from the other.

Proof. Let $\alpha = \min_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})}\boldsymbol{\omega}^t\boldsymbol{R}$ and let $\beta = \min_{\boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})}\frac{\boldsymbol{\omega}^t\boldsymbol{R}}{\sqrt{\boldsymbol{\omega}^t\boldsymbol{\Sigma}\boldsymbol{\omega}}}$. The two models can be reformulated as:

Model II

$$\max_{\alpha,\boldsymbol{\omega}} \quad \alpha$$
$$\text{s.t.} \quad \boldsymbol{\omega}^t\boldsymbol{R} \geqslant \alpha, \forall \boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})$$
$$\boldsymbol{\omega}^t\boldsymbol{\Sigma}\boldsymbol{\omega} \leqslant 1$$

Max-Sharpe

$$\max_{\beta,\mathbf{x}} \quad \beta$$
$$\text{s.t.} \quad \mathbf{x}^t\boldsymbol{R} \geqslant \beta\sqrt{\mathbf{x}^t\boldsymbol{\Sigma}\mathbf{x}}, \quad \forall \boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})$$
$$\mathbf{x}^t\boldsymbol{e} = 1$$

Here, we replace R with $\mathbf{x}\in\mathbb{R}^n$ in the Max-Sharpe model to avoid confusion between the two models when we do transformation on the variables. For convenience in derivation, we use the following notation:

$$\mathcal{F}_1 = \{(\alpha,\boldsymbol{\omega}) : \boldsymbol{\omega}^t\boldsymbol{\Sigma}\boldsymbol{\omega} \leqslant 1, \boldsymbol{\omega}^t\boldsymbol{R} \geqslant \alpha, \forall \boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})\}$$
$$\mathcal{F}_2 = \{(\beta,\mathbf{x}) : \mathbf{x}^t\boldsymbol{e} = 1, \mathbf{x}^t\boldsymbol{R} \geqslant \beta\sqrt{\mathbf{x}^t\boldsymbol{\Sigma}\mathbf{x}}, \forall \boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})\}$$
$$\mathcal{F}_3 = \{(\delta,\mathbf{y}) : \mathbf{y}^t\boldsymbol{R} \geqslant \delta\sqrt{\mathbf{y}^t\boldsymbol{\Sigma}\mathbf{y}}, \forall \boldsymbol{R}\in\mathcal{U}(\boldsymbol{R})\}$$

Then model II can be rewritten as $\max_{(\alpha,\boldsymbol{\omega})\in\mathcal{F}_1}\alpha$ and the Max-Sharpe model can be rewritten as $\max_{(\beta,\mathbf{x})\in\mathcal{F}_2}\beta$. We will prove these two reformulated models equivalent by proving both of them are equivalent to $\max_{(\delta,\mathbf{y})\in\mathcal{F}_3}\delta$. Let α^*, β^*, δ^* be the optimal values of the three models correspondingly.

First, we notice that for all $(\alpha,\boldsymbol{\omega})\in\mathcal{F}_1$, we can show that $(\alpha,\boldsymbol{\omega})\in\mathcal{F}_3$. Thus, $\delta^* \geqslant \alpha^*$. Similarly, for all $(\delta,\mathbf{y})\in\mathcal{F}_3$, we have $(\delta, \frac{\mathbf{y}}{\sqrt{\mathbf{y}^t\boldsymbol{\Sigma}\mathbf{y}}})\in\mathcal{F}_1$ and this leads to $\alpha^* \geqslant \delta^*$. Combining both of these

results, we obtain $\alpha^* = \delta^*$ and conclude that model II is equivalent to $\max_{(\delta,\mathbf{y})\in\mathcal{F}_3}\delta$ because there is a transformation between the optimal solution of one model to that of the other.

Second, since $\mathcal{F}_2 \in \mathcal{F}_3$, we have $\delta^* \geqslant \beta^*$. In addition, for all $(\delta,\mathbf{y}) \in \mathcal{F}_3$, then $(\delta, \frac{\mathbf{y}}{\mathbf{y}^T\boldsymbol{e}}) \in \mathcal{F}_2$ and hence $\beta^* \geqslant \delta^*$. Combining both of these results, we obtain $\beta^* = \delta^*$ and conclude that the Max-Sharpe model is equivalent to $\max_{(\delta,\mathbf{y})\in\mathcal{F}_3}\delta$ because there is a transformation between the optimal solution of one model to that of the other. Thus, the three models are equivalent and the proof is completed. □

References

[1] E. Adida, G. Perakis, A robust optimization approach to dynamic pricing and inventory control with no backorders, Mathematical Programming 107 (1) (2006) 97–129.

[2] R. Almgren, N. Chriss, Optimal portfolios from ordering information, Journal of Risk 9 (2006) 1–47.

[3] R. Ball, P. Brown, An empirical evaluation of accounting income numbers, Journal of accounting research (1968) 159–178.

[4] A. Ben-Tal, A. Nemirovski, Robust solutions of linear programming problems contaminated with uncertain data, Mathematical Programming 88 (3) (2000) 411–424.

[5] A. Ben-Tal, L. El Ghaoui, A.S. Nemirovskiĭ, Robust Optimization, Princeton Univ. Press, 2009.

[6] D. Bertsimas, M. Sim, The price of robustness, Operations Research 52 (1) (2004) 35–53.

[7] D. Bertsimas, A. Thiele, A robust optimization approach to inventory theory, Operations Research 54 (1) (2006) 150–168.

[8] D. Bertsimas, D.B. Brown, C. Caramanis, Theory and applications of robust optimization. Arxiv, in press. 1010.5445.

[9] H.G. Beyer, B. Sendhoff, Robust optimization – a comprehensive survey, Computer Methods in Applied Mechanics and Engineering 196 (33–34) (2007) 3190–3218.

[10] Bienstock, D., 2006. Experiments with robust optimization. In: Presentation at the International Symposium on Mathematical Programming, Rio de Janeiro, Brazil.

[11] O. Briant, C. Lemaréchal, P. Meurdesoif, S. Michel, N. Perrot, F. Vanderbeck, Comparison of bundle and classical column generation, Mathematical Programming 113 (2) (2008) 299–344.

[12] L. Chen, S. He, S. Zhang, Tight bounds for some risk measures, with applications to robust portfolio selection, Operations Research 59 (4) (2011) 847–865.

[13] V.K. Chopra, W.T. Ziemba, The effect of errors in means, variances, and covariances on optimal portfolio choice, Journal of Portfolio Management (1993) 6–11.

[14] V. DeMiguel, L. Garlappi, R. Uppal, Optimal versus naive diversification: how inefficient is the $1/n$ portfolio strategy?, Review of Financial Studies 22 (5) (2009) 1915–1953.

[15] E.F. Fama, K.R. French, Common risk factors in the returns on stocks and bonds, Journal of Financial Economics 33 (1) (1993) 3–56.

[16] D. Goldfarb, G. Iyengar, Robust portfolio selection problems, Mathematics of Operations Research 28 (1) (2003) 1–38.

[17] J. Gondzio, A. Grothey, Solving non-linear portfolio optimization problems with the primal-dual interior point method, European Journal of Operational Research 181 (3) (2007) 1019–1029.

[18] M. Grotschel, L. Lovász, A. Schrijver, Geometric Algorithms and Combinatorial Optimizations, Springer-Verlag, 1993.

[19] C.C. Lin, Y.T. Liu, Genetic algorithms for portfolio selection problems with minimum transaction lots, European Journal of Operational Research 185 (1) (2008) 393–404.

[20] M.S. Lobo, L. Vandenberghe, S. Boyd, H. Lebret, Applications of second-order cone programming, Linear Algebra and its Applications 284 (1–3) (1998) 193–228.

[21] M. Markowitz Harry, Portfolio selection, Journal of Finance 7 (1) (1952) 77–91.

[22] R.O. Michaud, The markowitz optimization enigma: is' optimized' optimal?, Financial Analysts Journal 45 (1) (1989) 31–42.

[23] K. Natarajan, M. Sim, J. Uichanco, Tractable robust expected utility and risk models for portfolio optimization, Mathematical Finance 20 (4) (2010) 695–731.

[24] G. Perakis, A. Sood, Competitive multi-period pricing for perishable products: a robust optimization approach, Mathematical Programming 107 (1) (2006) 295–335.

[25] K.A. Ravindra, T.L. Magnanti, J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, 1993.

[26] A.L. Soyster, Convex programming with set-inclusive constraints and applications to inexact linear programming, Operations Research 21 (5) (1973) 1154–1157.