

Experiments in Robust Portfolio Optimization

Daniel Bienstock

January 2007

Abstract

We present experimental results on portfolio optimization problems with return errors under the robust optimization framework. We use several a histogram-like model for return deviations, and a model that allows correlation among errors, together with a cutting-plane algorithm which proves effective for large, real-life data sets.

1 Introduction

This paper presents numerical experiments using cutting-plane algorithms to solve complex non-convex robust optimization models arising in portfolio optimization. The models we study are motivated by realistic considerations, and are nominally NP-hard; however we show that using modern optimization methodology one can solve large, real-life models quite efficiently.

Robust optimization is an approach for optimization under uncertainty that does not rely on a stochastic model to model the distribution of data. Often one can view a robust optimization model as a two-stage game; in the first stage a one chooses values for decision variables, while in the second stage an adversary (or 'nature') picks values for data parameters from a given uncertainty set – the process is adversarial in that the worst-case data distribution will be chosen.

Robust optimization is sometimes criticized for being overly conservative. It may also not be conservative enough, since we may not effectively guard against data realizations outside of the uncertainty set. The only protection against this particular problem is to (somehow) enlarge the uncertainty set, but this of course may make us even more conservative, possibly in an unexpected way. Furthermore, in an abstract sense, a robust optimization model tends to give the same 'weight' to all data realizations, which may be unrealistic in practice. This difficulty is reduced in so-called 'ambiguous chance-constrained' models, but not completely removed.

These issues are well-known, and clearly one should, whenever possible, refine an uncertainty set to focus on the deviations of significance. To some degree, a choice of an uncertainty model is a reflection of risk aversion.

In this paper we focus on two types of uncertainty sets. The first type allows a modeler to specify frequencies of deviations (from nominal values) as a function of the magnitude of the deviation; essentially outlining a rough histogram of deviations. The second type models correlation among deviations and gives rise to value-at-risk and conditional value-at-risk problems. Both types of uncertainty sets explicitly include non-convexities observed from prior data.

We present cutting-plane algorithms for both problems. The algorithms run a sequence of two-step iterations; in the first step we solve an "implementor problem" which picks values for the decision variables, while the second step solves an "adversarial problem" which finds the worst-case data corresponding to the decision variables just selected by the implementor problem. The adversarial problem, in both cases, is a mixed-integer program. We present theoretical evidence and computational experience using large, real-life data, to show that, in fact, the adversarial problem is *not at all* difficult in spite of its being NP-hard.

In the case of the histogram model, our cutting-plane algorithms find near-optimal solutions in a few minutes of computation, and frequently in just a few seconds. In the case of our VaR and CVaR models, our algorithms are always fast and accurate, often only requiring seconds of computation.

The focus of this paper is restricted to the methodology, implementation and experimental testing of the performance of our algorithms. Of course, the real-life asset management implications

are of interest. But the practical deployment of *any* portfolio management approach necessarily includes a strategy for rebalancing positions; in our context such a strategy would clearly need to incorporate a flavor of ‘robustness’ (i.e., we need a dynamic scheme for trading and settling trades that hedges against prediction errors) for otherwise the overall approach is short-changed, and any experiments that fail to account for this fact would be inaccurate. The ‘laboratory’ testing of such dynamic schemes, using asset prices from past observations, is problematic, because it is difficult to properly estimate the so-called market impact of trades. Nevertheless, this is a topic that we will take up in future work.

1.1 Robust Optimization

The general Robust Optimization approach is due to Ben-Tal and Nemirovski, see for example [BN98]-[BN00]. A similar approach has been developed in [EGL97], [EGOL98]. By now a large body of elegant work exists; the approach has been refined, extended and used in many applications, see [AZ05], [BGGN04], [BGNV05], [BS03], [BPS03], [BT06]. [BBN06] presents a framework for compensating for errors that fall outside of the uncertainty region being considered.

Some similar elements can be found in the literature on *adversarial queueing theory* [BKRSW96] and in economics.

It is useful to consider an example extracted from [BN99]. Suppose we want to solve a linear program with n variables and m constraints where the constraint matrix is uncertain, i.e. we are given a set \mathcal{A} containing all possible realizations of the constraint matrix. The canonical problem considered in [BN99] is of the form:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b, \quad \forall A \in \mathcal{A}, \end{aligned} \tag{1}$$

in other words, we want to find a solution of minimum cost which is feasible for all realizations of the constraint matrix – the constraints are viewed as “hard”. A case considered in [BN99] considers the case of *row-wise* uncertainty. In this model, we are given, for $1 \leq i \leq m$, an n -vector \bar{r}_i and an $p(i) \times n$ matrix E^i (for some $p(i) \geq 0$). We say that an $m \times n$ matrix A is in \mathcal{A} , if

$$\text{for } 1 \leq i \leq m, (a_{i1}, a_{i2}, \dots, a_{in}) = \bar{r}_i^T + u_i^T E^i, \text{ for some } u_i \in R^{p(i)} \text{ with } \|u_i\| \leq 1. \tag{2}$$

Each condition (2) applies independently to each row; the vectors \bar{r}_i^T can be interpreted as “mean” or nominal values of the rows, the matrices E^i give the scale and geometry of the deviations, and the condition $\|u_i\| \leq 1$ describes how the adversary is constrained. Note that the deviations from the nominal values are row-wise, i.e. the adversary can affect different rows of the matrix independently.

It is seen that under condition (2) the robust LP (1) can be rewritten as

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & \end{aligned} \tag{3}$$

$$\sum_j \bar{r}_{ij} x_j \geq b_i + \|E^i x\|, \text{ for } 1 \leq i \leq m. \tag{4}$$

This problem, in turn, can be solved using conic programming techniques.

Bertsimas and Sim [BS03] present a different model of robustness in linear programming, and an elegant theoretical analysis of its behavior. In the main model in [BS03], for each entry (i, j) of the constraint matrix we are given quantities \bar{a}_{ij} and $\delta_{ij} \geq 0$, and for each row i of the matrix, we are given an integer $\Gamma_i \geq 0$. The uncertainty model is as follows: every entry a_{ij} of A satisfies

$$\bar{a}_{ij} - \delta_{ij} \leq a_{ij} \leq \bar{a}_{ij} + \delta_{ij}, \tag{5}$$

and for every row i of A , at most Γ_i entries satisfy $a_{ij} \neq \bar{a}_{ij}$. In other words, the values \bar{a}_{ij} can be viewed as estimates of the a_{ij} , the δ_{ij} are maximum deviations from the estimates, and the Γ_i indicate how many deviations can occur in any row. It is shown in [BS03] that the robust LP (1) can be formulated as a linear program, albeit a larger one. It can also be seen that the uncertainty model is equivalent to the following:

- (a) every entry a_{ij} equals one of $\bar{a}_{ij} - \delta_{ij}$, \bar{a}_{ij} , $\bar{a}_{ij} + \delta_{ij}$,
- (b) in every row i , exactly Γ_i entries satisfy $a_{ij} \neq \bar{a}_{ij}$,
- (c) if e.g. $x \geq 0$ then (a) can be sharpened; every entry a_{ij} equals either $\bar{a}_{ij} - \delta_{ij}$ or \bar{a}_{ij} .

A final point regarding the approach in [BS03] is that the robust formulation can be large. If, on average, in each row k coefficients are uncertain ($\delta_{ij} > 0$) then the robust formulation will have more than km variables and constraints.

These examples contain an ingredient frequently found in the Robust Optimization literature – the robust problem is formulated as a single, comprehensive convex optimization problem, in particular, a second-order cone program or a linear program. The benefit of this approach is that we obtain a problem that is theoretically “tractable”, i.e. it can be solved in polynomial time.

A disadvantage of this comprehensive approach is that the convex optimization problem may nevertheless prove difficult to solve in practice. Certainly there exist real-world families of linear programs which in nominal (certain) form are already difficult enough; the robust versions of such linear programs should prove even more difficult. Another disadvantage is that not every uncertainty model is amenable to this approach. This is significant in that from a user’s standpoint, a particular an uncertainty model may be justifiable in terms of observed data, and furthermore the choice of an uncertainty model (in particular, which kind of data deviations are viewed as relevant) may reflect that user’s risk aversion level.

1.2 Portfolio Optimization

Portfolio allocation problems have been of interest to optimizers since their initial development by Markowitz [M52], [M59]. The typical problem to be solved is of the form

$$\min \quad \kappa x^T Q x - \bar{\mu}^T x \tag{6}$$

$$\text{s.t.} \quad Ax \geq b. \tag{7}$$

Here, x is an n -vector of “asset weights”, Q is an $n \times n$ symmetric, positive-semidefinite matrix (the covariance of returns matrix) and $\bar{\mu}$ is an n -vector (the vector of expected asset returns). The linear inequalities $Ax \geq b$ generally (but not always) include the constraints $\sum_j x_j = 1$ and $x \geq 0$, and may also include other portfolio construction requirements. In addition, $\kappa \geq 0$ is the so-called risk-aversion multiplier, a quantity that must be picked by the user. The solution to 7 is a set of asset allocations, with (hopefully) high overall expected return (the quantity $\bar{\mu}^T x$) and low variance of return (the quantity $x^T Q x$).

The quantity κ is a *risk aversion parameter*. For $\kappa = 0$ we obtain a solution of maximum return, while as $\kappa \rightarrow +\infty$ we converge to a minimum variance solution. Frequently, users of portfolio optimization choose κ using idiosyncratic criteria.

These two extreme cases can be refined. On the one hand, we have the problem

$$\begin{aligned} \max \quad & \bar{\mu}^T x \\ \text{s.t.} \quad & Ax \geq b, \end{aligned} \tag{8}$$

$$x^T Q x \leq \Theta, \tag{9}$$

where $\Theta \geq 0$ is a given constant, and on the other hand we have

$$\begin{aligned} \min \quad & x^T Q x \\ \text{s.t.} \quad & Ax \geq b, \\ & \bar{\mu}^T x \geq \mu_0. \end{aligned} \tag{10}$$

where μ_0 is given. All three variants occur in practice, although in our experience (7) is the most common. A different approach is that of maximizing the *Sharpe ratio*, i.e. solving

$$\begin{aligned} \max \quad & \frac{\bar{\mu}^T x - r_0}{\sqrt{x^T Q x}} \\ \text{s.t.} \quad & Ax \geq b, \end{aligned} \tag{11}$$

where r_0 is a reference value (the risk-free return). It can be seen that if $Ax \geq b$ includes the constraint $\sum_j x_j = 1$, (11) reduces to a convex quadratic program.

Factor models are widely used in practice to describe the structure of the matrix Q . In brief, a factor model will give rise to a decomposition

$$Q = VMV^T + D \tag{12}$$

where for some value k (typically $k \ll n$), V is $n \times k$ and M is $k \times k$ (and symmetric, positive-semidefinite), and D is a diagonal matrix with nonnegative entries. Using (12), (7) becomes

$$\min \quad \kappa \left(\sum_j d_j x_j^2 + w^T M w \right) - \bar{\mu}^T x \tag{13}$$

$$\text{s.t.} \quad Ax \geq b, \tag{14}$$

$$V^T x - w = 0. \tag{15}$$

Here, w is an r -vector of additional variables.

There have been many research efforts applying robust optimization techniques to portfolio problem; here we will only review a short list. One of the earliest examples appears in [LVBL85]. Goldfarb and Iyengar [GI04] give a robust optimization approach for portfolio optimization using factor models. They give robust versions problems (8), (10), and (11), using an uncertainty model where the matrices V and D belong to ellipsoids, and each entry $\bar{\mu}_j$ belongs to some interval I_j of the reals.

Their model makes the standard statistical assumption that returns have a multivariate normal distribution. Their procedure can be viewed as a methodology for immunizing the statistical estimation procedure that gives rise to the estimates F and $\bar{\mu}$ against idiosyncrasies of the observed data – still assuming, of course, that the observed data is drawn from the underlying multivariate normal distribution. In all cases, their robust optimization models give rise to SOCPs (second-order cone programs).

Tütüncü and Koenig [TK04] consider interval sets describing the uncertainty in returns and variance. Ceria and Stubbs [CS06] consider a model where returns are uncertain lie in a known ellipsoid, and furthermore the sum of deviations of returns from their expected values is assumed to be zero; i.e. the adversary “gives back” exactly the same as is taken away.

2 Uncertainty models for portfolio optimization

In this section we consider the portfolio optimization problem (6)-(7). We give two robust optimization models that address uncertainty in the returns. Unlike the work in [GI04], we do not seek to produce a more robust statistical estimation procedure – rather, our models can be seen as driven by past observations of “errors”, i.e. deviations from return estimates.

As before, we denote by $\bar{\mu}$ the estimate for the expected returns vector, which is an input to our problem. On the other hand, we denote by μ the vector of “random” returns, i.e. the returns that occur when data is realized.

As noted before, robust optimization problems can often be cast in the general form $F^* = \min_{x \in X} \max_{d \in D} f(x, d)$. Our generic method will be an adaptation of the classical cutting-plane procedure, which can also be viewed (abstractly) as a version of Benders procedure [B62]. In this we run an iterative algorithm which maintains and updates a subset \tilde{D} of D . Initially, $\tilde{D} = \emptyset$.

BASIC ALGORITHM: IMPLEMENTOR-ADVERSARY

Output: values L and U with $L \leq F^* \leq U$, and $x^* \in X$ such that $L = \max_{d \in D} f(x^*, d)$.
Initialization: $D = \emptyset$, $L = -\infty$, $U = +\infty$.

Iterate:

- 1. Implementor problem:** solve $\min_{x \in X} \max_{d \in \tilde{D}} f(x, d)$, with solution x^* .
Reset $L \leftarrow \max_{d \in \tilde{D}} f(x^*, d)$.
- 2. Adversarial problem:** solve $\max_{d \in D} f(x^*, d)$, with solution d^* .
Reset $U \leftarrow \min \{f(x^*, d^*), U\}$.
- 3. Test:** If $U - L$ is small, exit, else reset $\tilde{D} \leftarrow \tilde{D} \cup d^*$ and go to **1**.

The Basic Algorithm constitutes a generic template of all the algorithms we will consider in this paper. Typically, the implementor problem will be a convex program, while the adversarial problem will be a mixed-integer program, and each run of the adversarial problem will generate a cut (or cuts) to be added to the formulation for the implementor problem. Practice with cutting-plane algorithms (in particular, in the context of Benders’ Decomposition applied to Stochastic Programming) indicates that the careful use of heuristics to provide an effective starting point for the algorithm (as opposed to using $\tilde{D} = \emptyset$) can significantly speed up convergence – we will make use of this idea in one of our models. It is worth pointing out that the adversarial problem need be solved to proved optimality only at the last iteration (to prove termination of the overall algorithm).

In order to simplify the exposition, we will make the following

Assumption 2.1 For $1 \leq j \leq n$, we have $\bar{\mu}_j > 0$,

although our algorithms can be simply extended in case the assumption does not hold.

2.1 Histogram model

We will first provide a formal definition of the model, henceforth called the *histogram* model, and then discuss it. In the histogram model the adversary produces a vector η which embodies the main structural features of the return shortfalls we consider, plus another vector, ω , used to handle small, idiosyncratic errors. As before, n indicates the number of assets, i.e. the number of x variables. Our model uses the following parameters:

- (H.1) Integers $K \geq 0$ and $H \geq 0$,
- (H.2) Values $0 = \gamma_0 < \gamma_1 < \gamma_2 < \dots < \gamma_K \leq 1$. For $1 \leq i \leq K$, integers n_i, N_i , with $0 \leq n_i \leq N_i \leq n$.
- (H.3) For each $1 \leq h \leq H$, a value $\Lambda_h \geq 0$ and a set T_h of assets.
- (H.4) A value $0 \leq \Omega \leq 1$.

The uncertainty model is given by the conditions:

- (H.5) The random returns vector, μ , is computed as $\mu_j = \eta_j - \bar{\mu}_j \omega_j$. We impose $\mu \geq 0$. The vectors μ , η and ω are such that:
- (H.6) For $2 \leq i \leq K$, the number of assets j satisfying $(1 - \gamma_i)\bar{\mu}_j < \eta_j \leq (1 - \gamma_{i-1})\bar{\mu}_j$ is at least n_i and at most N_i . At least n_1 assets j satisfy $(1 - \gamma_1)\bar{\mu}_j < \eta_j \leq \bar{\mu}_j$, and at most N_1 assets satisfy $(1 - \gamma_1)\bar{\mu}_j < \eta_j < \bar{\mu}_j$. No assets j satisfy $\eta_j \leq (1 - \gamma_K)\bar{\mu}_j$.
- (H.7) $\sum_j \omega_j \leq \Omega$,
- (H.8) For $0 \leq h \leq H$, $\sum_{j \in T_h} \mu_j \geq \Lambda_h \sum_{j \in T_h} \bar{\mu}_j$.

2.1.1 Comments on the model

- (i) (H.6) describes the core of our model. Suppose, for the time being, that we use $\Omega = 0$. Then $\mu = \eta$, and (H.6) amounts to an approximate histogram description of (downwards) errors in the returns. More specifically, the γ_i describe percentage drops in returns. So, for example, we might use $K = 4$, $\gamma_1 = 0.1\%$, $\gamma_2 = 0.2\%$, $\gamma_3 = 0.5\%$ and $\gamma_4 = 1\%$. Or, if we are more risk averse, we might choose $\gamma_1 = 1\%$, $\gamma_2 = 2\%$, $\gamma_3 = 5\%$ and $\gamma_4 = 10\%$. (The *scale* of the deviations that we want to immunize against are a reflection of our risk aversion). In general, we would expect that K would be rather small (say, $K \leq 10$). The case $i = 1$ is handled separately in (H.6) so that the number of assets j with $\eta_j = \bar{\mu}_j$ (i.e., no shortfall, modulo ω_j) is unlimited. This feature is needed because, for example, we might have $\sum_i N_i < n$ (this could be avoided by e.g. setting $N_1 = n$, but then we would allow many assets to lose a fraction γ_1 of their return).
- (ii) (H.7) can be used to accommodate further (smaller) idiosyncratic errors in the returns. In addition, we will primarily use this feature to test the sensitivity of our model to variations in parameters.
- (iii) (H.8) represents a means for limiting the power of the adversary. A simple case of this constraint is that where T_h is the set of all assets. More generally, we could use (H.7) to limit the decrease in returns in the top *deciles* (suggested to us by R. Tütüncü [T06]).
- (iv) Likewise, the constraints in the model imply that the adversarial return vector, μ , is nonnegative. This is not a requirement and we only include it to avoid being overly conservative; nevertheless we can allow negative returns through simple modifications of the algorithms given below.

We will refer to the sets T_h as *tiers*, and we will say that asset j is in *sector* i , $2 \leq i \leq K$, if $(1 - \gamma_i)\bar{\mu}_j < \eta_j \leq (1 - \gamma_{i-1})\bar{\mu}_j$, and in *sector* 1 if $(1 - \gamma_1)\bar{\mu}_j < \eta_j < (1 - \gamma_0)\bar{\mu}_j = \bar{\mu}_j$. Note that if j is in no sector then $\eta_j = \bar{\mu}_j$. Also note that for $i > 1$, the number of assets in sector i between n_i and N_i ; and the number of assets in sector 1 is at most N_1 but may be smaller than n_1 .

A natural question is how a modeler would construct a histogram model. The following procedure can be applied:

- (a) Fix reasonable values for K and the γ_i ,
- (b) Using past data, compute an estimate of the average \bar{n}_i , and the standard deviation $\bar{\delta}_i$, of the number of assets in sector i ,
- (c) For some constant $\psi_i \geq 0$, let $n_i = \bar{n}_i - \psi_i \bar{\delta}_i$, and $N_i = \bar{n}_i + \psi_i \bar{\delta}_i$. For example, we might choose $\psi_i = 2$. In general, the choice of ψ_i is a measure of risk aversion.
- (d) The tiers can be chosen as deciles, as well as a tier consisting of all the assets. Many choices are reasonable here. The Λ_h quantities are, again, a measure of risk aversion (the smaller they are, the more risk-averse the model becomes).

The histogram model can be viewed as a refinement of a pure interval model, i.e. one where we assume that $l_j \leq \mu_j \leq u_j$ for given constants l_j and u_j , for each j . Rather than simply assuming that μ_j lies in this range, using a histogram model we can specify a finer set of subintervals, together with an estimation of the number of assets whose return can fall in appropriate of subintervals.

There are many possible variations of the model:

- Rather than specifying percentage drops in returns, one could model decrease in return as multiples of e.g. the standard deviation of return. Here we would say that asset j is in *sector* i ($1 \leq i \leq K$) if $\bar{\mu}_j - \gamma_i \delta_j < \eta_j \leq \bar{\mu}_j - \gamma_{i-1} \delta_j$, where $\delta_j \geq 0$ is a given quantity. If δ_j is the standard deviation of μ_j , then we could use $\gamma_i = \frac{i}{2}$, i.e. we model deviations in ranges of one-half of a standard deviation.
- Make the sector boundaries depend on the asset, that is to say, say that asset j is in *sector* i if $(1 - \gamma_{j,i})\bar{\mu}_j < \eta_j \leq (1 - \gamma_{j,i-1})\bar{\mu}_j$. Thus, the number of sectors remains fixed, but the values $\gamma_{j,i}$ must be chosen by the modeler. Using a rule of the form $\gamma_{j,i} = \frac{\gamma_i}{\mu_j}$ yields the model in the previous paragraph.
- Classify the assets into a number of categories, and have a separate constraint (H.4) for each category. For example, we could have assets of high, medium and low volatility. We would say that asset j , in category m , is in sector i ($1 \leq i \leq K^m$), if $(1 - \gamma_i^m)\bar{\mu}_j < \eta_j \leq (1 - \gamma_{i-1}^m)\bar{\mu}_j$.

Even though the algorithms and implementations given below are in terms of the basic histogram model (H.1 - H.8) the extension to these variants is straightforward.

The simplest version of the model is that where there is one sector and $n_1 = 0$, no tiers ($H = 0$), and $\Omega = 0$. This case of the problem can be handled using the algorithm in [BS03]. As discussed in Section 1, in this case the model simplifies: *exactly* N_1 assets will be in sector 1. If $x \geq 0$, then exactly N_1 assets j will have return $\mu_j = (1 - \gamma_1)\bar{\mu}_j$ while for all other assets $\mu_j = \bar{\mu}_j$. When we have a model with tiers, and multiple sectors, the approach in [BS03] cannot be used and the simplification no longer holds: the number of assets in any sector i can be (at optimality) strictly between n_i and N_i , and we can have assets j with $(1 - \gamma_i)\bar{\mu}_j < \mu_j < (1 - \gamma_{i-1})\bar{\mu}_j$.

2.2 Solving the histogram model

Given a histogram model, the robust mean-variance portfolio optimization problem we are interested in can be written as

$$\min_x \max_{\mu} \left\{ \kappa x^T Q x - \mu^T x \right\}, \quad (16)$$

where the min is over the asset vectors and the max, over the uncertainty set. The above can be written as

$$\min_x \left\{ \kappa x^T Q x - \min_{\mu} \mu^T x \right\} = \min_x \left\{ \kappa x^T Q x - \mathcal{A}(x) \right\},$$

where $\mathcal{A}(x)$ denotes the worst-case return achieved by the asset vector x ; i.e. the smallest value $\mu^T x$ over all return vectors μ that are described by the model.

All of the variants of the traditional mean-variance problem (e.g. minimize variance subject to a return lower bound, maximize return subject to a variance upper bound, maximize Sharpe ratio) have a robust version, and our techniques are easily adapted to handle them – we have chosen (16) because it already incorporates all the features we are interested in.

In the rest of this section we will show how to compute $\mathcal{A}(x)$ as the solution to a mixed-integer program that in practice proves very easy. Furthermore, we can provide some theory that at least provides an indirect explanation for this ease of solution. Then we will show how to improve our implementor-adversary algorithm scheme to obtain an effective solution procedure.

Consider the following mixed-integer program:

$$\min \sum_j x_j \mu_j \quad (17)$$

$$\text{Subject to: } \mu_j - \eta_j + \bar{\mu}_j \omega_j = 0, \quad \forall j, \quad (18)$$

$$\eta_j + \sum_{i=1}^K \delta_{ij} = \bar{\mu}_j, \quad \forall j, \quad (19)$$

$$\gamma_{i-1} \bar{\mu}_j y_{ij} \leq \delta_{ij} \leq \gamma_i \bar{\mu}_j y_{ij}, \quad \forall i, j, \quad (20)$$

$$y_{ij} = 0 \text{ or } 1, \text{ all } i, j, \quad (21)$$

$$\sum_{i=1}^K y_{ij} \leq 1, \quad \forall j, \quad (22)$$

$$n_i \leq \sum_j y_{ij} \leq N_i, \quad \forall i, \quad (23)$$

$$\sum_{j \in T_h} \mu_j \geq \Lambda_h \sum_{j \in T_h} \bar{\mu}_j, \quad \forall h, \quad (24)$$

$$\sum_j \omega_j \leq \Omega, \quad (25)$$

$$0 \leq \mu, \delta \text{ and } \eta, \omega \text{ free.} \quad (26)$$

In this formulation, μ is the adversarial return vector, $y_{ij} = 1$ if asset j is in sector i , and δ_{ij} equals the deviation from the mean return in asset j when it is in sector i (and it equals 0, otherwise). Finally, ω_j and η_j are, respectively, the idiosyncratic error in the return of asset j , and the return without the idiosyncratic error (see (H.4), (H.5), (H.7)). Constraint (22) ensures that at most one sector is picked for any asset, while (23) and (20) enforce the histogram structure. (18) and (19) define the adversarial returns, while (24) are the tier constraints.

Lemma 2.2 *The value of the above mixed-integer program is $\mathcal{A}(x)$.*

Proof. Suppose first that (μ, η, ω) is feasible for the histogram model. For $1 \leq i \leq K$ and $1 \leq j \leq n$, if asset j is in sector i , we set $y_{ij} = 1$ and $\delta_{ij} = \bar{\mu}_j - \eta_j$. In addition, if the number of assets in sector 1 is $q < n_1$, we choose $n_1 - q$ additional assets j with $\eta_j = \bar{\mu}_j$ and set $y_{1j} = 1$, $\delta_{1j} = 0$. This is possible by the way (H.6) was stated in the case $i = 1$. For all other pairs i, j we set $y_{ij} = \delta_{ij} = 0$. We will show $(\mu, \eta, \omega, y, \delta)$ is feasible for (18)-(26); for which purpose we only need to argue that (19)-(23) are satisfied.

(22) is clear because any asset is, by definition, in at most one sector. By construction of the y values, (23) holds. By definition of sectors, (20) holds. For given j , (19) is clear if $y_{ij} = 1$ for some (and hence, one) i , if j is in *no* sector then $\eta_j = \bar{\mu}_j$ and again (19) holds. Thus, indeed, $(\mu, \eta, \omega, y, \delta)$ is feasible for (18)-(26), and therefore $\mathcal{A}(x)$ is an upper bound on the value of the mixed-integer program.

Conversely, suppose now that we have a vector $(\mu, \eta, \omega, y, \delta)$ feasible for (18)-(26). Suppose that for each pair i, j with $y_{ij} = 1$ we have $\delta_{ij} < \gamma_i \bar{\mu}_j$; then (μ, η, ω) is feasible for the histogram model. If, on the other hand, for some pair \check{i}, \check{j} with $y_{\check{i}\check{j}} = 1$ we have $\delta_{\check{i}\check{j}} = \gamma_{\check{i}} \bar{\mu}_{\check{j}}$, then for $\epsilon > 0$, small, we can reset $\delta_{\check{i}\check{j}} \leftarrow \delta_{\check{i}\check{j}} - \epsilon$, and correspondingly, reset $\eta_{\check{j}} \leftarrow \eta_{\check{j}} + \epsilon$ and $\mu_{\check{j}} \leftarrow \mu_{\check{j}} + \epsilon$. Thus the change *increases* $\mu_{\check{j}}$, and since $\gamma_{\check{i}-1} < \gamma_{\check{i}}$ the new vector is feasible for the mixed-integer program if ϵ is small enough. Proceeding in this manner with every such pair \check{i}, \check{j} we will obtain a vector (μ, η, ω)

feasible for the histogram model; while the change in $\sum x_j \mu_j$ is proportional to ϵ . ■

As noted above, the robust portfolio optimization problem that we consider can be written as

$$\mathcal{H} \doteq \min \quad \kappa x^T Q x - \mathcal{A}(x) \tag{27}$$

$$\text{s.t.} \quad A x \geq b. \tag{28}$$

To solve this problem we adapt our basic implementor-adversary template. In the case of the histogram model, each run of the implementor problem generates a vector of asset weights x^* , and each run of the adversarial problem produces a vector μ of returns.

2.2.1 The implementor problem for the histogram model

Consider iteration h of the cutting-plane algorithm. Let $\mu_{(i)}$, $1 \leq i \leq h - 1$, be the return vectors produced by the prior runs of the adversarial problem. Then the implementor problem at iteration h is a convex quadratic program:

$$\min \quad \kappa x^T Q x - r \tag{29}$$

$$\text{s.t.} \quad A x \geq b, \tag{30}$$

$$r - \mu_{(i)}^T x \leq 0, \quad 1 \leq i \leq h - 1.$$

We expect that in a successful application of the cutting-plane approach, the overall number of iterations will be small. Further, each run of the implementor problem will be “not very different” than the previous. Since simplex-like methods for quadratic programming can be warm-started, we therefore expect that each implementor problem should be quickly solved.

2.2.2 The adversarial problem for the histogram model

Let x^* be a given vector of asset weights. Then the adversarial problem corresponding to x^* is precisely the mixed-integer program (17)-(26) described above. We will discuss this problem in the next section.

2.2.3 An improved algorithm for the histogram model

When using a cutting-plane algorithm such as that in our implementor-adversary template, an immediate concern involves the number of iterations that will be needed for convergence. As we will see below, when properly initialized with an appropriate heuristic the algorithm will require very few iterations; and in many cases the default version of the algorithm (without heuristics) already converges in few iterations.

In view of the previous section, however, our algorithm will have to solve a possibly large mixed-integer program at each iteration. A central question is, therefore, how difficult do we expect the adversarial problem to be?

We will first provide evidence to show that, frequently, the answer to this question is “not difficult at all”. As a byproduct of our answer we will describe a slightly different uncertainty model that can be solved in polynomial time, and which gives rise to a polynomial-time heuristic for the overall robust optimization problem which frequently proves extremely effective. We will then outline how the adversarial problem could in principle prove nontrivial. Finally, we will describe our complete algorithm for the robust optimization problem.

Returning to the adversarial problem, in the Appendix we prove the following result:

Theorem 2.3 *Suppose $x \geq 0$, $\bar{\mu} \geq 0$, and $\Omega = 0$. For every fixed K and H , and for every $\epsilon > 0$, there is an algorithm that runs in time polynomial in ϵ^{-1} and n , and that finds a solution to the adversarial problem (17-26) such that $\bar{\mu} x - \mathcal{A}(x)$ is approximated within multiplicative error $\leq \epsilon$.*

We stress that we state this theorem only for the sake of theoretical completeness. However, it is important to consider its implications. First, the algorithm in the theorem does not yield a strict separation method (see [GLS93]). Nevertheless, in a practical application we do expect K and H to be relatively small numbers and the assumption in the theorem is reasonable, and therefore, even though the adversarial problem can be shown to be NP-hard from a theoretical standpoint it is 'easy' to approximate the optimum return shortfall that the adversary can achieve, at least when $\Omega = 0$. This is not unexpected since the adversarial problem is closely related to the standard 0-1 knapsack problem – which is only truly hard when the coefficients require many bits of precision, and in any case it is only hard if exact optimality is required. To put it in a different way, the adversarial problem is non-convex, but it is not truly 'combinatorial' – there is a vast gulf between problem (17- 26) with K and H relatively small and, say, a graph coloring problem. A better interpretation of the theorem is that, put simply, we should not expect that the adversarial problem will be hard to solve using a modern mixed-integer programming solver. This fact is amply borne out by our computational experience.

However, the main point here is that the techniques underlying Theorem 2.3 strongly suggest that the adversarial problem should be very closely approximated by its *linear programming relaxation*. This is the problem obtained by replacing constraint (21) with $0 \leq y_{ij} \leq 1$, for all pairs i, j . In summarized form, we obtain a linear program of the form

$$\mathcal{L}(x) \doteq \min \quad \sum_j x_j \mu_j \tag{31}$$

Subject to:

$$M_1 \mu + M_2 \eta + M_3 \delta + M_4 y + M_5 \omega \geq \psi, \tag{32}$$

where the M_i are appropriate matrices and ψ is an appropriate vector (we stress that $0 \leq y_{ij} \leq 1$ is included in this system). Without proof, we state a simple result concerning the above linear program (also see Lemma A.3 in the Appendix).

Lemma 2.4 *Suppose $\Omega = 0$ and that the tiers T_h are pairwise disjoint. Then there is an optimal solution to (31)-(32), such that the number of y_{ij} variables taking fractional value is at most $H(K + 1)$.*

In fact, the Lemma holds under much more general conditions than disjointness of the tiers. In general, we should expect that the number of fractional y_{ij} in an optimal solution to the LP (31)-(32) will grow slowly as a function of HK , and thus, since we expect both H and K to be rather on the small side, small compared to n .

In the linear programming relaxation we allow the y_{ij} to take fractional values; hence we should have

$$\mathcal{L}(x) \leq \mathcal{A}(x),$$

but as discussed, we expect that, frequently,

$$\mathcal{L}(x) \approx \mathcal{A}(x). \tag{33}$$

In our experiments, this is precisely the case. In turn this suggests a different approach, that of using the linear program (31)-(32) as the definition of an alternative uncertainty model. We will refer to this model as the *relaxed* histogram model.

In summary, thus, we have two robust optimization problems. First, the histogram model, which as stated before produces the robust optimization problem

$$\begin{aligned} \mathcal{H} = \min \quad & \kappa x^T Q x - \mathcal{A}(x) \\ \text{s.t.} \quad & Ax \geq b. \end{aligned}$$

and the relaxed histogram model,

$$\begin{aligned} \mathcal{R} &\doteq \min && \kappa x^T Qx - \mathcal{L}(x) \\ &\text{s.t.} && Ax \geq b. \end{aligned}$$

where as before, $Ax \geq b$ are the portfolio construction techniques. We have that

$$\mathcal{H} \leq \mathcal{R}, \tag{34}$$

since the adversary is more powerful in the relaxed model, but as before we expect that, frequently

$$\mathcal{H} \approx \mathcal{R}. \tag{35}$$

Again, this is often the case. In our experiments using real data, \mathcal{R} frequently approximates \mathcal{H} correctly to at least two or three digits. What is more, the relaxed robust optimization problem can be solved in polynomial time. This follows from strong LP duality – the argument is well-known but we will state it for future reference.

Lemma 2.5

$$\mathcal{R} = \min \quad \kappa x^T Qx - r \tag{36}$$

$$\text{s.t.} \quad Ax \geq b, \tag{37}$$

$$r - \psi^T \alpha \leq 0, \tag{38}$$

$$x - M_1^t \alpha = 0, \tag{39}$$

$$\alpha^T (M_2, M_3, M_4, M_5) = 0, \tag{40}$$

$$0 \leq \alpha. \tag{41}$$

■.

The relaxed robust optimization model is in polynomial time, but the quadratic program (36)-(41) is significantly larger than the original, nominal quadratic program (it has more than Kn variables and constraints) and can prove significantly more difficult to solve. Nevertheless, in our experiments the overall running time is tolerable.

Now we turn to the less positive aspects of the relaxed model. Even though we expect (33) to hold, there is no theoretical guarantee that it will. In this context, it is worth considering a simple version of the adversarial problem for the histogram model. Suppose $K = 1$ (one segment), suppose there is a single tier ($H = 1$) consisting of all the assets, and suppose $\Omega = 0$. Writing $\delta_j = \bar{\mu}_j - \mu_j$ for each j , the adversarial problem for the histogram model reduces to:

$$\begin{aligned} &\max && \sum_j x_j \delta_j \\ &\text{Subject to:} && \sum_j \delta_j \leq (1 - \Lambda_1) \sum_j \bar{\mu}_j, \\ &&& 0 \leq \delta_j \leq (1 - \gamma_1) y_j, \\ &&& \sum_j y_j \leq N_1, \\ &&& y_j = 0 \text{ or } 1, \text{ all } j \end{aligned}$$

This is the *cardinality constrained knapsack problem*, which has been previously studied [B96], [DFN02]. It is NP-hard, and one can generate examples where there is a large gap between the value of this mixed-integer program and its linear programming relaxation.

Thus, in principle, one *can* generate examples where (33) will not hold, and, likewise, examples where (35) does not hold. In our experience, however, these have been rather artificial examples; nevertheless the fact remains that, from a theoretical perspective, the histogram model and the relaxed histogram model could give rise to significantly different problems. Moreover, even if (35) *does* hold, this fact can only be known *a posteriori* if we use the basic implementor-adversary algorithm in order to compute \mathcal{H} , which can be much more computationally expensive than computing \mathcal{R} (especially in difficult models with relatively large number of sectors K).

In order to handle these issues in a way that is theoretically complete and also efficient in practice, we will amend our basic implementor-adversary template so that

- (a) We first solve the relaxed model – this requires polynomial time.
- (b) In case (35) holds, we efficiently take advantage of this fact to quickly solve the histogram model.
- (c) In case (35) does not really hold, we still have a valid algorithm.

This approach is described in the next section.

2.2.4 The amended algorithm

Our amended algorithm makes use of the following (easy) observations:

Lemma 2.6 *Let \hat{x} be an optimal solution to the QP (36)-(41), and let $\hat{\mu}$ be optimal dual variables for constraints (39). Then*

- (i) $\hat{\mu}$ is a returns vector that is feasible for the relaxed histogram model, and
- (ii) \hat{x} is an optimal solution to the QP

$$\min \quad \kappa x^T Qx - r \tag{42}$$

$$s.t. \quad Ax \geq b, \tag{43}$$

$$r - \hat{\mu}^T x \leq 0. \tag{44}$$

Proof. (i) Follows from weak duality, and (ii) follows from the first-order optimality conditions for (36)-(41).■

Lemma 2.7 *Let $\check{\mu}$ be any returns vector that is feasible for the histogram model. Then*

$$\mathcal{H} \geq \min \quad \kappa x^T Qx - r \tag{45}$$

$$s.t. \quad Ax \geq b, \tag{46}$$

$$r - \check{\mu}^T x \leq 0. \tag{47}$$

Proof. Trivial, $\check{\mu}$ is just one returns vector for the histogram model.■

We can now describe our algorithm, in outline:

AMENDED IMPLEMENTOR-ADVERSARY ALGORITHM

Step 1: Solve the QP (36)-(41); let \hat{x} be an optimal solution to the QP and let $\hat{\mu}$ be optimal dual variables for constraints (39).

Step 2: Find a returns vector $\check{\mu}$ that is feasible for the histogram model, and such that $\check{\mu}$ is “close” to $\hat{\mu}$.

Step 3: Run the basic implementor-adversary algorithm initialized with $D = \{\check{\mu}\}$.

Omitting (for the time being) a precise description of Step 2, here is what the algorithm achieves. Assuming that, indeed, $\mathcal{L}(\hat{x}) \approx \mathcal{A}(\hat{x})$, then we should be able to find a vector $\check{\mu}$ in Step 2 such that

$$\hat{\mu}^t \hat{x} \approx \check{\mu}^T \hat{x}.$$

In fact, it should be the case (using Lemma 2.6 (ii)) that \hat{x} “nearly” satisfies the first-order optimality conditions for the QP in Lemma 2.7, i.e.

$$r(\check{\mu}) \doteq \min \quad \kappa x^T Qx - r \tag{48}$$

$$\text{s.t.} \quad Ax \geq b, \tag{49}$$

$$r - \check{\mu}^T x \leq 0. \tag{50}$$

In summary, then, we will have

$$r(\check{\mu}) \leq \mathcal{H} \leq \mathcal{R} \tag{51}$$

(where Lemma 2.7 gives the first inequality, and the second inequality is (34)), while at the same time the difference between \mathcal{R} and $r(\check{\mu})$ is “small”. But, according to Step 3 of our amended algorithm, $r(\check{\mu})$ is the value of the first implementor problem. Thus, already in the first iteration of the basic implementor-adversary approach, we will have “tight” lower and upper bounds on the robust optimization problem. In subsequent iterations the lower bound can only improve (as we add more cuts to the implementor problem), and of course, \mathcal{R} , always remains a valid upper bound. An extreme example of this scheme is that where $\hat{\mu}$ is itself feasible for the histogram model – in this case $\check{\mu} = \hat{\mu}$ and the amended algorithm will terminate immediately.

To complete the description of the amended algorithm, we must specify how Step 2 is to be carried out. In our implementation, we solve the following problem, with variables $\epsilon_j, \check{\mu}_j, \eta_j, \omega_j, 1 \leq j \leq n$, and $y_{ij}, \delta_{ij}, 1 \leq i \leq K, 1 \leq j \leq n$:

$$\min \quad \sum_j \epsilon_j \tag{52}$$

Subject to:

$$\epsilon_j \geq |\hat{\mu}_j - \check{\mu}_j| \quad \forall j, \tag{53}$$

$$(\check{\mu}, \eta, \omega, y, \delta) \text{ feasible for (18) – (26)}. \tag{54}$$

In other words, we minimize the L_1 norm of the error entailed in approximating $\hat{\mu}$ with a return vector $\check{\mu}$ feasible for the histogram model. Thus, the above problem is a mixed integer program. However, we do not need to solve it to complete optimality – in our implementation, we set a limit on the proved accuracy of the computed solution.

As a final remark, we point out that there is an alternative algorithm for the robust optimization problem under the histogram model, which progressively tightens the relaxed formulation by using polyhedral cuts (to separate from the convex hull of return vectors feasible under the histogram model). In terms of the formulation (36)-(41), this approach amounts to a *column-generation* approach, where at each iteration we prove a tighter (i.e., smaller) upper bound on \mathcal{H} . From a practical standpoint, we would still need to prove good lower bounds on \mathcal{H} , for example using a returns vector computed as in the previous paragraph. We did not implement this algorithm; a computational hurdle will be the need to solve a large QP at each iteration.

2.3 Computational experiments with the histogram model

In our implementation all quadratic programs, linear programs, and mixed-integer programs are handled using a commercial solver [CP10].

The quadratic program (36)-(41) in Step 1 is solved using the dual solver in [CP10] – the barrier solver can prove faster, but the “dual crossover step”, needed so as to make Step 2 of the amended algorithm faster, can become slow. Subsequent instances of the implementor’s problem are solved using the primal QP solver in [CP10], warm started at the prior optimum.

All quadratic programs are solved using default tolerances in [CP10] (e.g. 1e-06 optimality tolerance). The mixed-integer program in the adversarial problem is also solved using default tolerances; note that in this case the optimality tolerance is 1e-04. We enforced early termination of the mixed-integer program (52)-(54) solved in Step 2 of the amended algorithm, as soon as the absolute gap between upper and lower bounds proved by the MIP solver was less than $1e-04 \hat{\mu}^T \hat{x}$.

The termination criterion for the amended algorithm uses a tolerance parameter, $0 < \tau \leq 1$, as follows. Consider a given iteration of the algorithm. Let \tilde{x}, \tilde{r} denote the optimal solution to the implementor problem, and let r^{adv} denote the value of the adversarial problem, i.e. the minimum return that \tilde{x} attains under the histogram model. Then the algorithm terminates any of the following conditions apply:

- (a) If $|r^{adv}| > 1e-06$ and $\tilde{r} - r^{adv} \leq \tau|r^{adv}|$,
- (b) If $|r^{adv}| \leq 1e-06$ and $\tilde{r} - r^{adv} \leq \tau(1 + |r^{adv}|)$,
- (c) If $|\kappa\tilde{x}^T Q\tilde{x} - \tilde{r}| > 1e-06$ and $\tilde{r} - r^{adv} \leq \tau|\kappa\tilde{x}^T Q\tilde{x} - \tilde{r}|$,
- (d) If $|\kappa\tilde{x}^T Q\tilde{x} - \tilde{r}| \leq 1e-06$ and $\tilde{r} - r^{adv} \leq \tau(1 + |\kappa\tilde{x}^T Q\tilde{x} - \tilde{r}|)$.

Conditions (a) and (b) capture the case where the implementor has approximated the worst-case adversarial return sufficiently closely. Likewise, conditions (c) and (d) are active when the error in the worst-case return estimation is small compared to the overall objective value.

All our testing was performed on real-life problem instances of the factor model (13) – the only added data are the parameters in the uncertainty model. The following table summarizes information about the underlying data sets we used – for each data set we performed a number of studies by varying the parameters of the uncertainty model.

	A	B	C	D	E	F	G	H	I
n	500	500	499	499	703	1338	2019	2443	2464
rows	20	20	20	140	108	81	140	153	153
factors	19	19	19	139	60	41	139	152	152

Table 1: Data set parameters

In the table, “rows” indicates the total number of constraints in the problem, i.e. constraints (14) and “factors” is the number of factors (see eq. (15)).

2.3.1 Measuring performance.

The purpose of our initial set of tests is to document the performance of our algorithm. In Table 2 we consider a variety of simple problem instances. In each of these instances

- (a) There is one segment ($K = 1$), $\gamma_1 = 0.10$, $n_1 = 0$ and $N_1 = \lfloor n/10 \rfloor$.
- (b) There is one tier ($H = 1$). This tier consists of all the assets, and $\Lambda_1 = 0.05$.
- (c) $\Omega = 0$,
- (d) $\tau = 1e-03$.

In Table 2, “time” is the overall running time, in seconds, “step1QP” is the time spent on solving the QP in Step 1, “stepMIP” is the time spent on solving the MIP in Step 1, “iters” is the number of iterations of the basic implementor-adversary algorithm (encountered in Step 3 of the amended

ID	time	step1QP	step1MIP	iters	impT	advT
A	0.68	0.36	0.02	7	0.12	0.14
B	0.41	0.32	0.01	0	0.03	0.00
C	0.48	0.33	0.01	3	0.09	0.05
D	31.70	30.42	0.01	0	1.25	0.00
E	2.28	2.08	0.01	0	0.18	0.00
F	1.79	1.62	0.01	0	0.09	0.03
G	193.70	179.41	0.08	0	14.21	0.00
H	24.86	16.59	0.04	0	8.22	0.00
I	53.46	33.74	0.04	28	16.46	2.36

Table 2: Base runs with the histogram model

algorithm), “impT” is the cumulative time spent solving implementor problems in Step 3 and “advT” is the cumulative time spent solving adversarial problems in Step 3.

As we can see from the table, in every single run, the Step 1 QP running time dominates by a wide margin. Also note the modest time spent on solving MIPs, especially in the case of adversarial problems – this confirms the remarks made above regarding the simplicity of these problems. Further, with the exception of data set I, every other run required very few (or no) iterations of the implementor-adversary procedure within Step 3. Data set I frequently proves problematic; from our testing it would appear that it is the structure of the factor covariance matrix M that causes problems.

The next set of runs constitute more of a stress-test on the algorithm. For data sets A, F, H and I, we run the algorithm on examples where:

- (a) There are six tiers. Tier 1 consists of all the assets, and $\Lambda_1 = 0.05$. For $2 \leq h \leq 6$, tier h consists of the $(h - 1)^{st}$ return decile, e.g. tier 2 consists of the top 10% highest return assets, tier 3 consists of the next 10% assets, and so on. In this case we have $\Lambda_h = 0.1$, for all $h \geq 2$, i.e. in each additional tier we allow the total percentage return loss to be as high as 10%.
- (b) The number of segments is varied. We use, for $K = 2, 3, 4$, K segments, where segment i ($1 \leq i \leq K$) has $\gamma_i = \frac{i}{K}$, $n_i = 0$ and $N_i = \lfloor \frac{n}{10K} \rfloor$.
- (c) As before, we use $\Omega = 0$ and the tolerance $\tau = 1e-03$.

Our choice of the histogram model in (b) implies that large errors are just as likely as small errors. This is probably an unrealistic assumption; however it is exactly this feature that can make problems more difficult (or, put more accurately, the more difficult problems become even more so). The results are given in Table 3, where “01vars” is the number of 0/1 variables in each mixed-integer program ($= Kn$). All other headings in Table 3 have the same meaning as in Table 2.

Note the weak dependence of running time (or number or iterations) on K . Also, on the difficult data set I, the time spent solving QPs clearly dominates. On instances A and I (and to some degree, F) the heuristic in Step 1 of the amended algorithm fails; and in fact the running time is longer than it would be if we used the basic implementor-adversary setup directly. As it turns out, using a heuristic variation of the folklore “smoothing” technique for cutting-plane algorithms (see, Nesterov [Ne05], [BI06], Nemirovski [N04] for modern theoretical results) produces much faster running times on the I instances, with small degradation on the easier instances. For lack of space we will not describe these results.

Our next set of results in this section concern the speed of convergence of the algorithm, and also serve as a better form of stress-test of our approach. The uncertainty model is more complex:

- There are $K = 10$ segments, and for $1 \leq i \leq 10$,

$$N_i = \left\lfloor \frac{1/i}{\sum_{j=1}^{10} 1/j} n \right\rfloor, \quad (55)$$

	K	time	step1QP	step1MIP	iters	impT	advT	01vars
A	2	23.49	0.66	7.25	120	8.78	6.67	1000
	3	50.41	0.76	31.16	99	5.58	12.74	1500
	4	60.83	4.88	14.86	89	4.88	14.86	2000
E	2	1.67	1.38	0.03	1	0.21	0.05	2676
	3	1.97	1.58	0.09	1	0.20	0.07	5352
	4	2.21	1.77	0.15	1	0.20	0.10	5352
F	2	9.82	2.30	0.14	27	1.1	6.06	2676
	3	25.13	2.83	0.40	52	2.85	18.72	4014
	4	19.59	3.47	1.11	21	1.03	13.69	5352
G	2	68.86	55.65	0.16	1	12.27	0.17	4038
	3	99.59	83.02	0.34	1	11.34	0.22	6057
	4	133.05	119.8	0.46	1	11.68	0.33	8076
H	2	12.33	7.75	0.08	1	3.62	0.08	4886
	3	13.37	8.43	0.10	1	3.82	0.10	7329
	4	14.66	9.40	0.16	1	3.96	0.14	9772
I	2	523.28	42.68	11.35	318	375.37	91.91	4928
	3	725.59	47.06	123.67	328	403.95	123.67	7392
	4	1013.66	59.34	121.52	381	517.67	314.69	9856

Table 3: Tests with uniform errors, $2 \leq K \leq 4$ and $H = 6$

$$n_i = \left\lfloor \frac{N_i}{2} \right\rfloor, \quad (56)$$

$$\gamma_i = \frac{1}{11 - i}. \quad (57)$$

Thus, the segment structure allows the adversary to produce larger displacements approximating a “heavy tail”.

- We have six tiers, as in Table 3.
- We used $\Omega = 0$, and the tolerance $\tau = 5\text{e-}04$.

τ	A	B	C	D ^b	E [*]	F	G [*]	H	I
5.0e-02	214.53	14.81	144.86	122.53	11.77	274.64	136.43	11.93	140.29
1.0e-02	223.21	15.49	144.86	122.53	14.66	356.98	224.56	11.93	140.29
5.0e-03	254.73	16.03	162.41	126.63	34.16	363.84	601.63	11.93	140.29
1.0e-03	300.88	35.23	183.12	157.49	64.61	469.75	764.85	11.93	140.29
5.0e-04	361.20	37.92	216.52	167.40	73.87	598.94	924.63	11.93	140.29

Table 4: Convergence time on heavy-tailed instances, $K = 10$, $H = 6$

Table 4 shows the CPU time (in seconds) required by the algorithm to prove various levels of accuracy (shown in the leftmost column) using the bounds obtained as the run proceeded. Columns

highlighted with “*” correspond to problems where the large QP in Step 1 could not be solved, while the column highlighted with “b” corresponds to a problem where the Barrier solver could handle the QP in Step 1, but not so for the simplex solvers. It is worth pointing out that the default tolerance in the underlying MIP solver [CP10] is 1e-04 – this can be tightened, but sometimes at a steep computational cost. As a result, we left this tolerance unchanged; and consequently the overall error tolerance in our algorithm should probably be set slightly coarser than 1e-04.

However, an additional point regarding Table 4 is the fact that, for pragmatic purposes, the tolerance $\tau = 1e-03$ is probably excessively fine, and $\tau = 1e-02$ should (more than) suffice. In fact, the parameters that embody the uncertainty model (*any* uncertainty model) must be regarded as “soft”, and it is not clear what benefit is derived from using a very sharp tolerance value. As we can see, the time savings achieved by using $\tau = 1e-02$ over $\tau = 1e-03$ are often substantial. In addition, it is to be expected that the quality of the solution at $\tau = 1e-02$ is, a posteriori, much better than the 1% estimate.

In Table 5 we describe additional tests using $K = 10$. Here we used the same tier structure as for the results in Table 4, $\Omega = 0$, and for $1 \leq i \leq K$,

$$N_i = \left\lfloor \frac{1/i^2}{\sum_{j=1}^{10} 1/j^2} n \right\rfloor, \quad (58)$$

$$n_i = \left\lfloor \frac{N_i}{2} \right\rfloor, \quad (59)$$

$$\gamma_i = \frac{i}{10}. \quad (60)$$

We used the tolerance $\tau = 1e-03$. For this test, we selected the more difficult data sets from Table 4. As before, on data set G the QP in Step 1 could not be solved, and Step 1 was bypassed. The column headings are as in Table 3.

	time	step1QP	step1MIP	iters	impT	advT	01vars
A	327.04	2.52	211.72	135	12.27	100.24	5000
C	29.32	3.01	9.35	27	1.02	15.76	4990
F	74.06	13.57	15.96	27	2.47	41.42	13380
G *	681.12	–	–	19	64.7	615.54	20190
I	124.82	93.38	22.58	1	4.17	2.46	24640

Table 5: Additional tests with $K = 10$, $H = 6$ (* = Step 1 bypassed)

2.3.2 Qualitative tests.

A question of interest is the sensitivity of the model to changes in the parameter Ω . Recall that in the histogram model the return μ_j includes an error term of the form $\bar{\mu}_j \omega_j$, where $\omega \geq 0$ and $\sum_j \omega_j \leq \Omega$. For $\Omega = 0, 1, \dots, 20$, Figure 1 plots the percentage increase in the value of the robust optimization problem, as compared to the case $\Omega = 0$, using data set E (with $K = 3$ and 3 tiers). Since this data set has $n = 703$, the case $\Omega = 20$ allows a simultaneous error of up to $\approx 28\%$ in every asset (modulo the tier constraints) – these are very large deviations. As we can see from the figure, the rate of growth of the objective value is moderate, and appears to slow down for larger Ω .

In Table 6 we describe a different type of experiments involving Ω . For each data set in the table, we first solve the robust optimization problem with $\Omega = 0$, save its solution, and then compute the worst-case behavior of that vector for positive values of Ω . If x^0 is the solution to the problem with $\Omega = 0$, and we denote by r^k the worst-case return earned by x^0 when $\Omega = r$, the table reports the ratio r^k/r^0 . The data sets used the same tier structure, and the first three sectors as sets used in table 4.

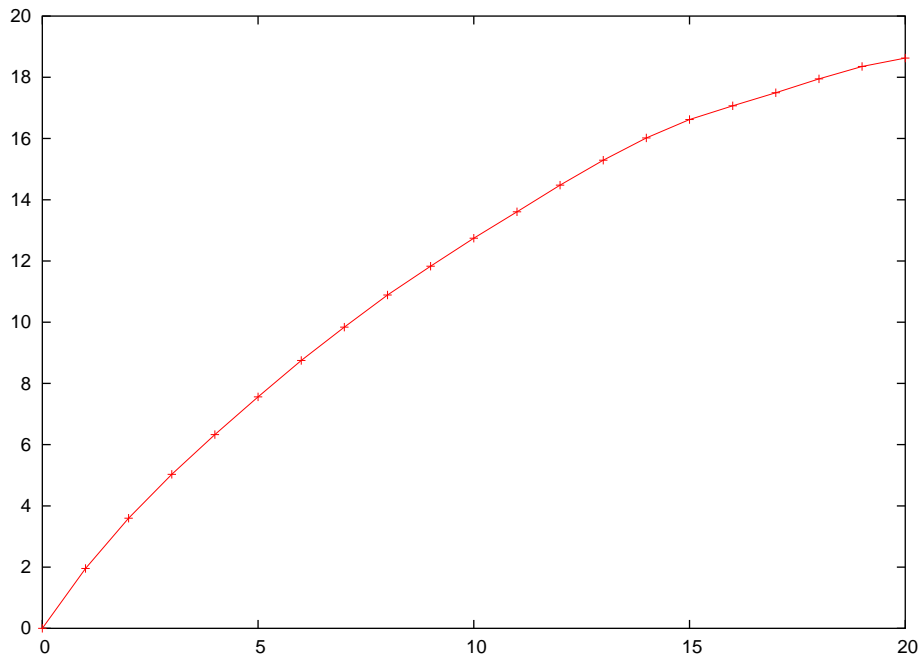


Figure 1: Value of robust optimization problem as a function of parameter estimation error

Ω	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.10	0.20
A	0.999	0.998	0.998	0.997	0.996	0.995	0.995	0.994	0.993	0.992	0.984
C	0.999	0.999	0.999	0.998	0.998	0.998	0.997	0.997	0.996	0.996	0.992
F	0.997	0.995	0.992	0.990	0.987	0.985	0.982	0.980	0.977	0.974	0.949
I	0.999	0.998	0.997	0.997	0.997	0.995	0.995	0.994	0.993	0.992	0.985

Table 6: Impact of Ω on worst-case return

As we can see from the table, the impact of Ω on solution quality seems moderate; it is either linear or sublinear.

Figure 2 shows the behavior of the basic implementor-adversary algorithm (no Steps 1 and 2) on data set I – this is the same run as on Table 4. The figure shows a snapshot taken every ten iterations. In each snapshot, the figure shows, in green, the implementor’s estimate of the worst case return that could be achieved by the asset vector produced at that iteration; in red, the actual worst-case return for that vector (as computed by the adversarial problem); and in blue, the nominal return value achieved by the same vector (i.e., the return computed using the nominal returns $\bar{\mu}$). All returns have been scaled so that the initial nominal return equals 1.00.

The figure shows that the implementor problem quickly converges to a near-optimal robust return estimate. At the same time, the nominal return value, after an initial dip, is essentially constant. Thus, the asset vectors produced after roughly the first 5% of the iterations are essentially equivalent, in terms of nominal return value. But, as the red curve shows, these asset vectors significantly differ from one another in their robustness until approximately 40% of the iterations have taken place. Finally, the dip in nominal return value mentioned above may be taken as a proxy for the cost of imposing robustness on the problem – though we caution that the min-max problem that has been solved is (28), and not a min-max return problem. At the same time, the gap between the final robust value (approximately 0.84) and the initial adversarial problem value (approximately 0.22) is the price of *non-robustness*.

Another point to be made is that after a very small number of iterations (say: 5% of the total) while still technically sub-optimal according to our tolerance, the asset vector is nevertheless very significantly more robust than the initial asset vector, i.e. the solution to the nominal problem. We regard the ability of a Benders-like algorithm to visibly improve solution quality during the first

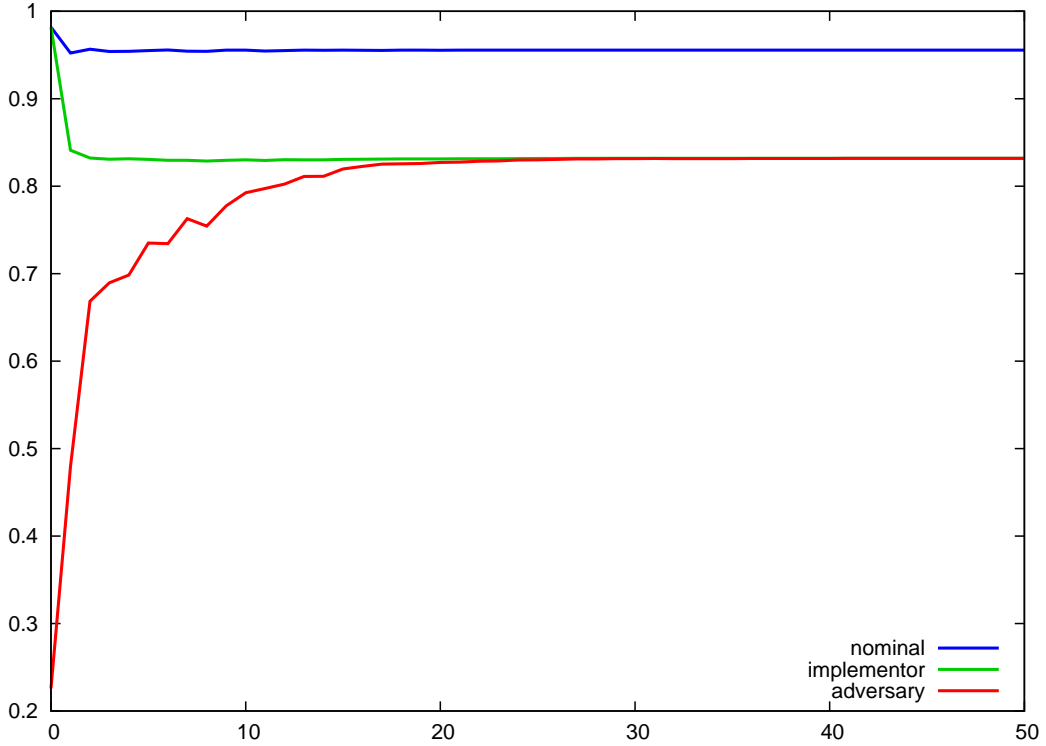


Figure 2: Typical behavior of algorithm without strengthening

few iterations (and at very small computational cost) as a very valuable feature.

Figure 3 refers to the same run, and shows the size of the support (the number of positive entries) of the asset vectors computed by the algorithm.

This figure clearly shows that after the first 30% of the iterations, the asset vectors markedly differ from the initial, nominally optimal asset vector, thus reinforcing the comments in the prior paragraph.

Figure 4 shows the behavior of the algorithm on data set H, using the same uncertainty set as in Table 4, but restricted to three tiers (one tier involving all assets, plus the two top return deciles). In this case the algorithm converges in 27 iterations and 52 seconds. In this case the price of robustness is more severe, but the price of non-robustness is equally large.

There is a useful parameter on which to estimate the impact of robustness on returns, which is similar to the Sharpe ratio. For a given asset vector x , consider the ratio

$$R(x) = \frac{\bar{\mu}^T x}{\bar{\mu}^T x - \mathcal{A}(x)},$$

where $\mathcal{A}(x)$ is as in eq. (17), i.e. the worst-case return that x can attain. Thus, in Figure 2, we have

$$R(x^{(0)}) \approx 1.30, \quad R(x^{opt}) \approx 6.53,$$

where $x^{(0)}$ is the solution to the nominal optimization problem, and x^{opt} is the optimum vector for the robust problem. Corresponding to Figure 4, we have

$$R(x^{(0)}) \approx 2.77, \quad R(x^{opt}) \approx 4.33.$$

In the final set of tests we compare several runs using data set F (with 1338 assets), and with $\tau = 1e-05$. In all these runs there are no tiers, but the runs differ in the structure of the segments. We consider the following cases, where the n_i are all zero in each case:

- (1) 1 segment, with $N_1 = 200$ and $\gamma_1 = 0.5$,
- (2) 2 segments, with $N_1 = 200$ and $\gamma_1 = 0.25$, and $N_2 = 100$ and $\gamma_2 = 0.5$,

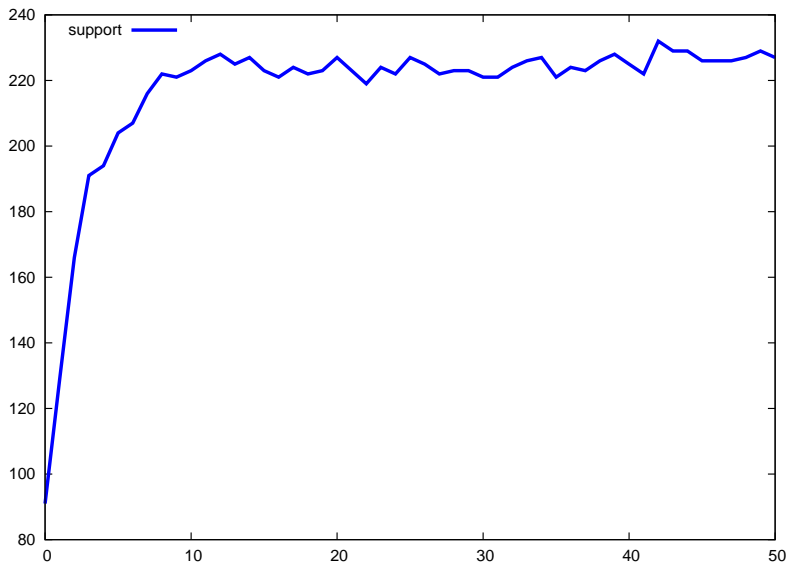


Figure 3: Growth in support

- (3) 3 segments, with $N_1 = 200$ and $\gamma_1 = 0.2$, $N_2 = 50$ and $\gamma_2 = 0.4$, and $N_3 = 50$ and $\gamma_3 = 0.8$,
- (4) 3 segments, with $N_1 = 200$ and $\gamma_1 = 0.1$, $N_2 = 50$ and $\gamma_2 = 0.16$, and $N_3 = 90$ and $\gamma_3 = 0.8$,
- (5) 2 segments, with $N_1 = 200$ and $\gamma_1 = 0.1$, $N_2 = 100$ and $\gamma_2 = 0.8$, and
- (6) 1 segment, with $N_1 = 100$ and $\gamma_1 = 1.0$.

Thus, from a convex perspective, all cases are equivalent: the adversary can, in each case, decrease returns by a total “mass” of 100. Yet, as we can see from Table 7, the six cases are structurally quite different.

Case	1	2	3	4	5	6
opt nominal	9.13348	9.07071	4.12856	4.15375	4.27876	4.27685
opt robust	4.56677	4.56543	1.61677	1.6378	1.67355	1.59757
opt support	173	194	232	266	267	267

Table 7: Impact of non-convexity

In this table, “opt nominal” is the nominal return attained by the optimal solution x^{opt} to the robust optimization problem, “opt robust” is the minimum return attained by x^{opt} under the uncertainty model, and “opt support” is the cardinality of the support of x^{opt} . It is clear from this table that the histogram structure would be difficult to replace with a single, convex uncertainty model.

3 VaR and CVaR models

In this section we consider extensions of the histogram models which are designed to capture the folklore notion that unforeseen, negative, deviations in data can happen “all at once” or in other words, seem correlated.

To this end, we will apply the “ambiguous chance constrained” framework of Erdogan and Iyengar [EI04], also see [CC05]. Here, given a decision on the part of the implementor, the adversary does not directly choose values for the data. Rather, the following process takes place:

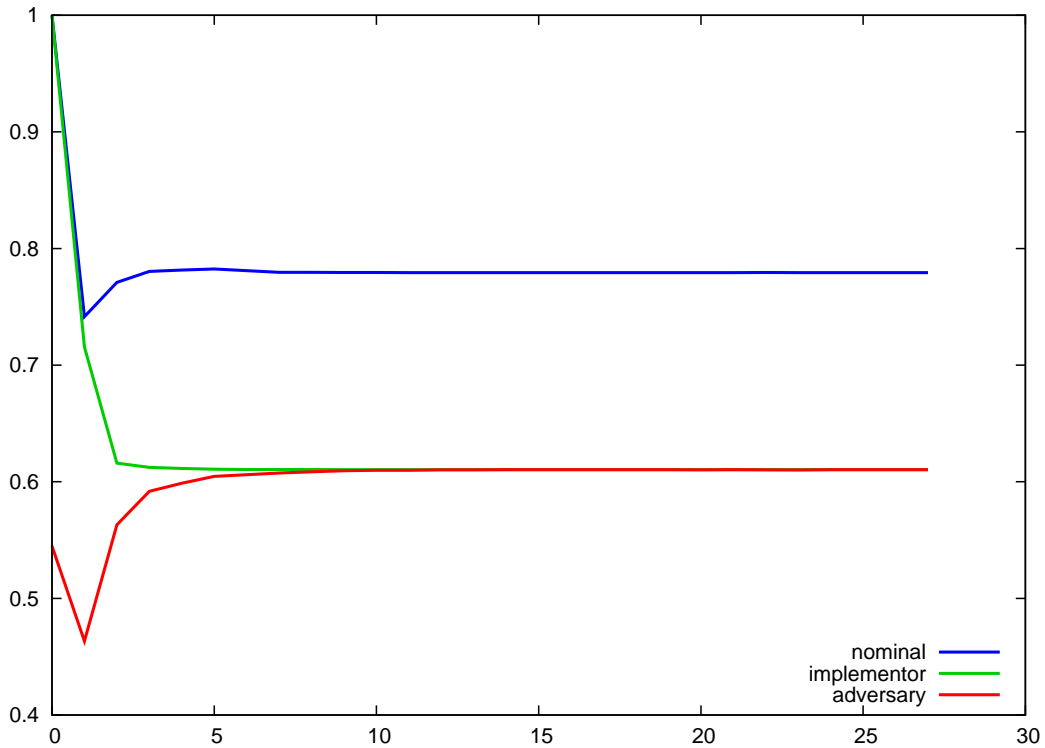


Figure 4: Behavior on data set H

1. The adversary chooses a *probability distribution* \mathbf{P} , from a family of distributions \mathcal{P} (known to the implementor) of the data.
2. A random sample is drawn from \mathbf{P} thereby setting values from the data.

As a simple example for this approach, consider a model where a single real parameter, v is known to be normally distributed, with mean $\mu_v \in [1, 2]$ and standard deviation $\sigma_v \in [0.5, 1]$. The adversary can choose any normal distribution thus restricted, but the underlying randomness allows the implementor some possibility of hedging. Similar models have been proposed in different contexts: supply chain ([GM93], [MG94], [S58]), and adversarial queueing [BKRSW96].

Clearly, such models can be used to reduce the power of the adversary. On the other hand, one needs an underlying rationale to justify the choice of a particular distribution \mathcal{P} . Moreover, it is known that the chance constrained framework can give rise to difficult optimization problems.

In the context of a Markowitz-type portfolio optimization problem, we are interested in the following generic model:

Given a vector x^* of assets,

- (r.a) The adversary chooses a vector $w \in \mathbf{R}^n$, and a probability distribution \mathbf{P} with range in $[0, 1]$. The adversary is constrained in the joint choice of w and \mathbf{P} .
- (r.b) A real $0 \leq \delta \leq 1$ is drawn from \mathbf{P} .
- (r.c) The random returns vector, μ , is computed using $\mu_j = (1 - \delta w_j) \bar{\mu}_j$, $1 \leq j \leq n$.

Instead of (r.c) we could have a model where instead $\mu_j = \bar{\mu}_j - \delta \gamma_j$, for some $\gamma_j \geq 0$. Such a model could be handled using techniques similar to those used below.

There are many possible ways in which this template can be made specific. For simplicity in this paper we have chosen the following concrete model, which we refer to this as the *random* model. We are given the following parameters as inputs to the model (i.e., they are not chosen by the adversary):

- (R.1) An integer $K \geq 1$, values $0 = \delta_0 < \delta_1 < \dots < \delta_K \leq 1$, and pairs of values $0 \leq \pi_i^l \leq \pi_i^u \leq 1$ for $1 \leq i \leq K$.
- (R.2) An integer $H \geq 1$, and for each $1 \leq h \leq H$, a value $\Gamma_h \geq 0$ and a set T_h (a “tier”) of assets and a value $0 \leq \Lambda_h$.
- (R.3) For $1 \leq j \leq n$, a value $u_j \geq 0$.

The adversary is constrained by the following rules:

- (R.4) The choice of w satisfies $0 \leq w_j \leq u_j$, for $1 \leq j \leq n$.
- (R.5) For $0 \leq i \leq K$, the adversary chooses $\pi_i = \text{Prob}(\delta = \delta_i)$, constrained by $\pi_i^l \leq \pi_i \leq \pi_i^u$.
- (R.6) For $1 \leq h \leq H$, the vectors w and π must satisfy:

$$\mathbf{E} \left(\delta \sum_{j \in T_h} \bar{\mu}_j w_j \right) \leq \Gamma_h.$$

Note that condition (R.6) may be rewritten as: $\left(\sum_{i=1}^K \pi_i \delta_i \right) \left(\sum_{j \in T_h} \bar{\mu}_j w_j \right) \leq \Gamma_h$.

Notation 3.1 Given w and π produced by the random model, we denote by $\mu^{w,\pi}$ the random vector which has $\mu_j^{w,\pi} = \bar{\mu}_j(1 - \delta_i w_j)$ with probability π_i , for $1 \leq i \leq K$ and $1 \leq j \leq n$.

The random model is best understood in contrast to the histogram model in Section 2.1. Suppose we are given a model as in (H.1)-(H.8). In particular, we are given parameters $0 \leq \gamma_1 \leq \gamma_2 \leq \dots \leq \gamma_K \leq 1$, and the histogram model provides a rough measure of the frequency that the return of an asset j can lie between $(1 - \gamma_i)\bar{\mu}_j$ and $(1 - \gamma_{i-1})\bar{\mu}_j$ for each i . This constitutes an approximation to a probability distribution that describes how much, percentage-wise, the returns deviate (downwards) from their mean values. The random model does something similar, but through the use of an explicit probability distribution (explicit, though ambiguous in that the adversary picks it). To some degree, the random model may thus be less conservative than the histogram model. The quantities w_j provide a sense of *scale*, and are limited by condition (R.6) from becoming “too large”. Also note that (R.6) is similar to condition (H.8) – it limits the total (expected) shortfall in any given tier. We have chosen (R.6) simply to demonstrate the flexibility of the algorithms we have implemented. In any case, note that when the tiers correspond to return deciles, the returns of assets in each tier will on the whole be close to one another, and (H.8) and (R.6) may nearly be equivalent for appropriate choices of the Γ_h parameters.

In what follows, for simplicity of exposition, we will make the assumption that $x \geq 0$ is among the given constraints (7).

Having chosen our uncertainty model, we can formulate an appropriate robust optimization problem. There is a very large literature on risk measures under uncertainty, see [GF04] for some recent work and a review. A central idea is that of value-at-risk (VaR), and the related concept, conditional value at risk (CVaR) (see [RU00], [KPU02], and references therein). Our definition of VaR follows that of [GI04].

Definition 3.2 Consider an asset vector x , a vector w and a distribution π as in (R.4)-(R.6), a real ν , and a tail probability $0 < \theta < 1$. Then the adversarial value-at-risk of x under the choices w, π , denoted $AVaR^{w,\pi}(x)$, is the largest real ρ such that

$$\text{Prob} \left(\nu - \sum_j \mu_j^{w,\pi} x_j \geq \rho \right) \geq \theta.$$

[In the notation $\text{AVaR}^{w,\pi}(x)$ we have not included θ or ν as they will always be understood from the context.]

In our definition, the quantity ν is a “benchmark” (a desirable returns level) and would be an input to the overall robust optimization problem. Thus, when nonnegative, the quantity $\nu - \sum_j \mu_j^{w,\pi} x_j$ is a “shortfall”. The AVaR is that critical value of the loss whose probability is at least θ . The quantity θ is also an input to the model.

Our definition AVaR differs, slightly, from the traditional definition of VaR, which is, given ν, θ, x, w, π [KPU02]:

$$\text{VaR}^{w,\pi}(x) \doteq \min \left\{ \rho : \text{Prob} \left(\nu - \sum_j \mu_j^{w,\pi} x_j \leq \rho \right) \geq 1 - \theta \right\}. \quad (61)$$

The two definitions are similar, but not identical. In many cases they essentially correspond to the same notion. We have that $\text{VaR} \leq \text{AVaR} \leq \text{CVaR}$ (defined below). If the random variable π were continuous, with positive density function, then VaR and AVaR would exactly agree. In the discrete case, if there exists q with $\sum_{i=q-1}^K \pi_i < \theta < \sum_{i=q}^K \pi_i$, then $\text{VaR} = \text{AVaR} = \nu - \sum_j \bar{\mu}_j x_j + \delta_q \sum_j w_j x_j$. Thus, the following example typifies the unique case where a difference occurs.

Example. Suppose $\nu = 1$ and $\bar{\mu}^T x = 1$. Then for $1 \leq i \leq K$, the shortfall corresponding to the i^{th} deviation equals δ_i . Consider the case with $K = 6$, where the δ_i equal 0.2, 0.4, 0.6, 0.8, 0.9, 1.0, and the π_i equal, respectively, 0.4, 0.3, 0.2, 0, 0.05, 0.05. Suppose $\theta = 0.1$. Then $\text{AVaR} = 0.9$, but $\text{VaR} = 0.6$, which is “optimistic” in the sense that with probability at least 0.1 the shortfall is it at least 0.9.

The following summarizes the situation:

- (a) The definition of AVaR and eq. (61) exactly complement each other. Thus the AVaR can be viewed as a value-at-risk from the point of view of the adversary. For related discussion on VaR-like risk measures, see [AT02]; in [AT04] AVaR is called “upper value-at-risk”.
- (b) In our computational experiments, often the π_i will be positive (at least around the AVaR) and then the difference between VaR and AVaR will be zero or small.
- (c) The key point is that the models and algorithms given below are easily adapted to handle VaR, or other risk measures for that matter. For brevity in this paper we only consider AVaR, again because of the better fit with ambiguous chance-constrained models.

Returning to our model, we have:

Lemma 3.3 *Let x, w, π, ν and θ be given. Let q be largest such that $\sum_{i=q}^K \pi_i \geq \theta$. Then $\text{AVaR}^{w,\pi}(x) = \nu - \sum_j \bar{\mu}_j (1 - \delta_q w_j) x_j$.*

Proof. For $q < i \leq K$, $\delta_q < \delta_i$ and consequently, since x, δ and w are all nonnegative,

$$\nu - \sum_j \bar{\mu}_j (1 - \delta_q w_j) x_j \leq \nu - \sum_j \bar{\mu}_j (1 - \delta_i w_j) x_j. \quad (62)$$

In other words, writing $\rho = \nu - \sum_j \bar{\mu}_j (1 - \delta_q w_j)$,

$$\text{Prob} \left(\nu - \sum_j \mu_j^{w,\pi} x_j \geq \rho \right) \geq \pi_q + \pi_{q+1} + \dots + \pi_K, \quad (63)$$

which by assumption is at least θ . At the same time, for any value $\rho' > \rho$,

$$\text{Prob} \left(\nu - \sum_j \mu_j^{w,\pi} x_j > \rho' \right) \leq \pi_{q+1} + \dots + \pi_K, \quad (64)$$

which again by assumption is $< \theta$. ■

Note that according to our definition, it could be the case that $\text{AVaR}^{w,\pi}(x) < 0$.

In the models that we will first consider the adversary will choose w and π so as to maximize AVaR. Formally

Definition 3.4 Given a vector x , denote by $\text{AVaR}^{\max}(x)$ the maximum value-at-risk that can be incurred by x under the random model, i.e. $\text{AVaR}^{\max}(x) = \max_{w,\pi} \text{AVaR}^{w,\pi}(x)$, where the maximum is taken over all choices w, π as in (R.4)-(R.6).

Likewise, the implementor wants to choose x^* so as to minimize the maximum AVaR. The values ν, θ will be fixed and known to implementor and adversary, and from a modeling standpoint can be viewed as a measure of risk aversion.

We will also consider *conditional value-at-risk* (CVaR) models.

Definition 3.5 Consider an asset vector x , a vector w and a distribution π as in (R.4)-(R.6), a real ν and a tail probability $0 < \theta < 1$. Let $1 \leq q \leq K$ be smallest such that $\sum_{i=1}^q \pi_i \geq 1 - \theta$. Then the conditional value-at-risk of x under the choices w, π , is given by

$$\text{CVaR}^{w,\pi}(x) \doteq \frac{1}{\theta} \sum_{i=q}^K \pi_i \left(\nu - \sum_j \bar{\mu}_j (1 - \delta_i w_j) x_j \right)^+$$

Note: in the definition we could alternatively have the sum starting from $q + 1$; however the difference is small.

As for the AVaR case, given x , the adversary wants to choose w and π so as to maximize $\text{CVaR}^{w,\pi}$ (yielding a worst-case value $\text{CVaR}^{\max}(x)$), while the implementor wants to choose x so as to minimize $\text{CVaR}^{\max}(x)$.

3.1 The robust optimization problem

Informally, in the context of the AVaR (resp., CVaR) case, the implementor wants to choose a vector x with small worst case value-at-risk, i.e. small $\text{AVaR}^{\max}(x)$ (resp., $\text{CVaR}^{\max}(x)$). However, focusing solely on this objective might produce an overly conservative solution. Instead, we would like to rely on the folklore notion (widely found in the robust optimization literature) that typically one should be able to find solutions to a problem that are simultaneously near-optimal (e.g., compared to the optimal solution to the nominal problem) *and* robust. To that effect, we restate the nominal portfolio optimization problem,

$$\begin{aligned} \Psi^* = \min \quad & \kappa x^T Q x - \bar{\mu}^T x \\ \text{s.t.} \quad & A x \geq b. \end{aligned}$$

Let $\epsilon > 0$ be a positive tolerance parameter chosen by the modeler. The robust optimization problem that we consider is:

$$\min \quad L \tag{65}$$

$$\text{s.t.} \quad A x \geq b, \tag{66}$$

$$\kappa x^T Q x - \bar{\mu}^T x \leq \Psi^* + \epsilon, \tag{67}$$

$$\text{in the AVaR case,} \quad L \geq \text{AVaR}^{\max}(x) \tag{68}$$

$$\text{in the CVaR case,} \quad L \geq \text{CVaR}^{\max}(x) \tag{69}$$

Here, constraint (67) captures the notion of near-optimality. This is a (convex) quadratic constraint. Constraints (68) and (69), on the other hand, are non-convex. In order to handle the above problem we will use our implementor-adversary algorithmic template, suitably adapted.

For theoretical completeness, we state the following result:

Lemma 3.6 *In the AVaR case, problem (65)-(68) can be reduced to at most K SOCPs.*

The proof of this Lemma, which we omit, is similar to that given in the Appendix. The key behind the result is that we can enumerate all possible values of the parameter q as in the proof of Lemma 3.3. For each value of q , one can argue that the optimal distribution π is uniquely determined, and independent of x . Using linear programming duality, one obtains a “compact” SOCP formulation of the robust optimization problem (again, for the given value of q). It seems likely that the result can be improved to $O(\log K)$ SOCPs – however, our algorithm, empirically, requires a far smaller number of SOCPs to prove termination.

3.1.1 The implementor problem in the AVaR case

Remark 3.7 *Let w, π satisfy (R.4)-(R.5). Let q be highest such that $\pi_q + \pi_{q+1} + \dots + \pi_K \geq \theta$. Then*

$$L \geq \nu - \sum_j \bar{\mu}_j (1 - \delta_q w_j) x_j \quad (70)$$

is valid for (65)-(68).

Proof. Let x be any asset vector. To show that (70) is valid it suffices to show that

$$\text{AVaR}^{w,\pi}(x) \geq \nu - \sum_j \bar{\mu}_j (1 - \delta_q w_j) x_j, \quad (71)$$

which follows from Lemma 3.3. ■

Based on this Lemma, we can now state the implementor problem. Suppose we are running iteration r of the basic implementor-adversary algorithm. Thus, at each iteration $1 \leq t < r$, the adversary has produced a vector $w^{(t)}$ and a probability distribution $\pi^{(t)}$; and let $q(t)$ be largest such that $\pi_{q(t)} + \pi_{q(t)+1} + \dots + \pi_K \geq \theta$. Then the implementor problem at iteration r is:

$$\min \quad L \quad (72)$$

$$\text{s.t.} \quad Ax \geq b, \quad (73)$$

$$\kappa x^T Qx - \bar{\mu}^T x \leq \Psi^* + \epsilon, \quad (74)$$

$$L \geq \nu - \sum_j \bar{\mu}_j \left(1 - \delta_{q(t)} w_j^{(t)}\right) x_j, \quad 1 \leq t \leq r-1. \quad (75)$$

This is a convex, quadratically constrained linear program, solvable using SOCP techniques. Below (Section 3.1.5) we will see that inequality (75) can be considerably tightened.

3.1.2 Formulations for the adversarial problem in the AVaR case

We formulate the adversarial problem for the random model as a nonlinear, 0-1 mixed-integer program. Given an asset vector x , let M be a sufficiently large value, and consider the problem (with variables D, π, w, z):

$$\tilde{D}(x) = \min \sum_{i=1}^K D_i \quad (76)$$

Subject to:

$$\text{for } 1 \leq i \leq K, \quad \pi_i^l \leq \pi_i \leq \pi_i^u, \quad (77)$$

$$\sum_i \pi_i = 1, \quad (78)$$

$$\text{for } 1 \leq i \leq K, \quad z_i - \theta^{-1} (\pi_i + \pi_{i+1} + \dots + \pi_K) \leq 0, \quad (79)$$

$$\sum_{i=1}^K z_i = 1, \quad (80)$$

$$\text{for } 1 \leq i \leq K, \quad z_i = 0 \text{ or } 1, \quad (81)$$

$$\text{for } 1 \leq i \leq K, \quad D_i + Mz_i \geq 0, \quad (82)$$

$$\text{for } 1 \leq i \leq K, \quad D_i - Mz_i + \delta_i \sum_j \bar{\mu}_j x_j w_j \geq -M + \sum_j \bar{\mu}_j x_j \quad (83)$$

$$\text{for all } j, \quad 0 \leq w_j \leq u_j, \quad (84)$$

$$\left(\sum_{i=1}^K \delta_i \pi_i \right) \left(\sum_{j \in T_h} \bar{\mu}_j w_j \right) \leq \Gamma_h, \quad \text{for each tier } h \quad (85)$$

Lemma 3.8 Given x , $AVaR^{\max}(x) = \nu - \tilde{D}(x)$.

Proof. (83) can be rewritten as

$$D_i \geq M(z_i - 1) + \sum_j \bar{\mu}_j (1 - \delta_i w_j) x_j. \quad (86)$$

Consider an optimal solution $(\tilde{\pi}, \tilde{w}, \tilde{D}, \tilde{z})$ to (76 - 85). If $1 \leq i \leq K$ is such that $\tilde{z}_i = 0$ then, assuming we have chosen M large enough, $\tilde{D}_i = 0$ by (82). On the other hand, if $1 \leq p \leq K$ is the unique index such that $\tilde{z}_p = 1$, then (86) dominates (82), and thus $\tilde{D}(x) = \tilde{D}_p = \sum_j \bar{\mu}_j (1 - \delta_p w_j) x_j$.

On the other hand, let q be largest such that $\sum_{i=q}^K \tilde{\pi}_i \geq \theta$. Then by (79), $p \leq q$, and therefore, $\delta_p \leq \delta_q$, and since \tilde{w}, μ and x are all nonnegative,

$$\sum_j \bar{\mu}_j (1 - \delta_p \tilde{w}_j) x_j \geq \sum_j \bar{\mu}_j (1 - \delta_q \tilde{w}_j) x_j.$$

Hence, without loss of generality, $p = q$, and the solution $(\tilde{\pi}, \tilde{w}, \tilde{D})$ achieves value-at-risk equal to $\nu - \tilde{D}_q$. The converse is similar. ■

Note that the proof explicitly used nonnegativity of w . Another issue is the constant M . Setting $M = \sum_j \bar{\mu}_j \max\{1, u_j\} x_j$ will suffice.

Constraint (85) is nonlinear. We next show how to approximate this constraint using a linear system. Let $N > 1$ be an integer and for any h consider the following mixed-integer (linear) system:

$$\sum_{j \in T_h} \bar{\mu}_j w_j \leq \Gamma_h \sum_{g=1}^N \frac{N}{g} \rho_{hg}, \quad (87)$$

$$\sum_{g=1}^N \frac{g-1}{N} \rho_{hg} \leq \sum_{i=1}^K \delta_i \pi_i \leq \sum_{g=1}^N \frac{g}{N} \rho_{hg}, \quad (88)$$

$$\sum_{g=1}^N \rho_{hg} = 1, \quad (89)$$

$$\rho_{hg} = 0 \text{ or } 1, \text{ for } 1 \leq g \leq N. \quad (90)$$

This system is a discretization of (85), and it relies on the fact that $0 \leq \sum_{i=1}^K \delta_i \pi_i \leq 1$ for every distribution π (which is due to $0 \leq \delta_i \leq 1$ for all i). Intuitively, since the system underestimates $\sum_{i=1}^K \delta_i \pi_i$ by at most an additive error of $1/N$, the approximation error should be quite small.

From a pragmatic standpoint (and keeping in mind the context – we cannot expect to know extremely sharp information about the probability distribution) we could take the model obtained by replacing (85) with (87 - 90) as our actual uncertainty model, but we comment on the error nevertheless.

In our implementations we have used $\delta_i = i/K$, $0 \leq i \leq K$, with $K \geq 100$, and here it can be argued that the error should be small when $N \geq K$. In fact, from an adversarial standpoint, the optimal value of $\sum_i \pi_i \delta_i$ *should* be “large” – the events with larger δ_i should have non-negligible probability in order to achieve high value-at-risk. Thus, since the smallest positive δ_i is $\delta_1 = 1/K$, we have that $\sum_i \pi_i \delta_i$ should be larger than $1/K$, unless our uncertainty model forces the adversary (through the bounds π_j^u , or the tier constraints) to concentrate most of the probability mass exactly at zero. From our point of view, this would essentially amount to mismodeling. As an illustrative example, suppose the adversary wants to achieve a “heavy tail” which nevertheless concentrates probability near zero, say

$$\pi_i = \frac{1}{i H_K}, \quad 1 \leq i \leq K,$$

with $\pi_0 = 0$, where $H^K = \sum_{i=1}^K 1/i \approx \ln K$. Then $\sum_i \pi_i \delta_i \approx 1/\ln K$. For $K = 100$ this is more than forty times larger than $1/K$. The following result formalizes the above:

Lemma 3.9 *Let x be a given asset vector, and let $(\tilde{\pi}, \tilde{w}, \tilde{D})$ be an optimal solution to (76 - 85), let q be largest such that $\sum_{i=q}^K \tilde{\pi}_i \geq \theta$, and let $1 \leq g \leq N$ be such that $\frac{q-1}{N} \leq \sum_{i=1}^K \delta_i \tilde{\pi}_i \leq \frac{q}{N}$. Finally, let $\hat{D}(x)$ be the value of the mixed-integer program obtained by replacing each constraint (85) with the corresponding system (87 - 90). Then $\tilde{D}(x) \leq \hat{D}(x) \leq \tilde{D}(x) + \frac{\delta_q}{g} \sum_j \bar{\mu}_j x_j \tilde{w}_j < (1 + \frac{1}{g}) \tilde{c}D(x)$.*

Proof sketch. Any solution (π, w, D) to the mixed-integer program is feasible for (76 - 85). This proves the first inequality. For the second inequality, note that $(\tilde{\pi}, \frac{q-1}{g} \tilde{w})$ is feasible for the mixed-integer program. ■

As a comment on this Lemma, consider our canonical choice of $\delta_i = \frac{i}{K}$, $0 \leq i \leq K$. Assuming that $\sum_{i=1}^K \delta_i \tilde{\pi}_i \geq 1/K$, we have that $g \geq \frac{N}{K}$. Of course $\delta_q \leq 1$, but typically we would expect δ_q to be rather smaller. Finally, we would expect that the benchmark ν , and the nominal return $\sum_j \bar{\mu}_j x_j$, would be of same the order of magnitude. Thus, if the w_j are bounded by small constants (for example, if $u_j = 1$ for all j) then the error term in the Lemma is very small provided N is large enough. As a final, related point regarding the construction, note that the Lemma shows how to approximate $\tilde{D}(x)$, and not $\text{AVaR}^{\max}(x)$. When $\nu \approx \tilde{D}(x)$ then $\text{AVaR}^{\max}(x) \approx 0$ and the approximation could be poor *on a percentage* basis. However, when $\text{AVaR}^{\max}(x) \approx 0$ then the knowledge that $\text{AVaR}^{\max}(x) \approx 0$ by itself is likely more useful than its precise value.

Note: there is an alternative, based on a “powers of two” approach, to the construction (87)-(90). In fact, we will use this alternative below in our algorithm for the CVaR problem. However, in general, we prefer the above method, in that the use of constraint (89) makes the formulation tighter.

3.1.3 The implementor problem in the CVaR case

Our approach for the CVaR case mirrors that in Section 3.1.1. Suppose that at each iteration $1 \leq t < r$, the adversary has produced a vector $w^{(t)}$ and a probability distribution $\pi^{(t)}$; and let $q^{(t)}$ be smallest such that $\pi_1 + \pi_2 + \dots + \pi_{q^{(t)}} \geq 1 - \theta$. Then the implementor problem at iteration r is:

$$\min \quad C \tag{91}$$

$$\text{s.t.} \quad Ax \geq b, \tag{92}$$

$$\kappa x^T Qx - \bar{\mu}^T x \leq \Psi^* + \epsilon, \tag{93}$$

$$C \geq \frac{1}{\theta} \sum_{i=q(t)}^K \pi_i^{(t)} \left(\nu - \sum_j \bar{\mu}_j (1 - \delta_i w_j^{(t)}) x_j \right), \quad 1 \leq t \leq r-1. \tag{94}$$

3.1.4 The adversarial problem in the CVaR case

Let x be a given asset vector. Consider the problem

$$C^*(x) \doteq \max_{L, \pi, w, z} \sum_{i=1}^K L_i \tag{95}$$

Subject to:

$$\text{for } 1 \leq i \leq K, \quad \pi_i^l \leq \pi_i \leq \pi_i^u, \tag{96}$$

$$\sum_i \pi_i = 1, \tag{97}$$

$$\text{for } 1 \leq i \leq K, \quad z_i - (1 - \theta)^{-1} (\pi_1 + \pi_2 + \dots + \pi_i) \leq 0, \tag{98}$$

$$\sum_{i=1}^K z_i = 1, \tag{99}$$

$$\text{for } 1 \leq i \leq K, \quad z_i = 0 \text{ or } 1, \tag{100}$$

$$\text{for } 1 \leq i \leq K, \quad L_i \leq \frac{\pi_i}{\theta} \left(\nu - \sum_j \bar{\mu}_j (1 - \delta_i w_j) x_j \right)^+, \tag{101}$$

$$L_i \leq \frac{\pi_i^u}{\theta} \left(\nu - \sum_j \bar{\mu}_j (1 - \delta_i u_j) x_j \right) \sum_{k=1}^i z_k, \tag{102}$$

$$\text{for all } j, \quad 0 \leq w_j \leq u_j, \tag{103}$$

$$\left(\sum_{i=1}^K \delta_i \pi_i \right) \left(\sum_{j \in T_h} \bar{\mu}_j w_j \right) \leq \Gamma_h, \quad \text{for each tier } h. \tag{104}$$

Lemma 3.10 *Let x be an asset vector. (a) If (96)-(104) is feasible, then $C^*(x) = \text{CVaR}^{\max}(x)$. (b) If $\text{CVaR}^{\max}(x) > 0$, then (96)-(104) is feasible. (c) If (96)-(104) is infeasible, then $0 = \text{CVaR}^{\max}(x)$ and x is an optimal solution to the robust optimization problem.*

Proof. Suppose $\text{CVaR}^{\max}(x) > 0$. Let w, π be an optimal solution to the adversarial problem, and let q be smallest such that (1) $\sum_{i=1}^q \pi_i \geq 1 - \theta$, and (2) $\nu - \sum_j \bar{\mu}_j (1 - \delta_q w_j) x_j \geq 0$. We set $z_q = 1$ and $z_i = 0$ for all other i ; we set $L_i = 0$ for all $i < q$ and $\nu - \sum_j \bar{\mu}_j (1 - \delta_i w_j) x_j$ for all $i \geq q$. Since the δ_i are monotonely increasing, we obtain a feasible solution to (96)-(104), of value $\text{CVaR}^{\max}(x)$. This proves (a) and (b) and the first statement in (c). Since CVaR is nonnegative, the proof is complete. ■

In contrast to the AVaR case, we have two nonlinearities: (101) and (104). We handle (104) as in the AVaR case, by replacing it with the subsystem (87)-(90), which entails adding the H N

0/1-variables ρ_{hg} . We could use a similar approach to handle (101), but this would result in KN 0/1 variables, which, for $K = 100$ and $N = 10000$ (say) would be probably be excessive. Instead, we rely on an approach due to Glover [G75]. Ignoring the “+” superscript, the right-hand side of (101) can be rewritten as

$$\frac{\pi_i}{\theta} \left(\nu - \sum_j \bar{\mu}_j x_j \right) + \frac{\delta_i}{\theta} \pi_i \sum_j \bar{\mu}_j w_j x_j,$$

with the second term nonlinear. Let W be an upper bound on $\sum_j \bar{\mu}_j w_j x_j$ (we discuss below how to set W). Let $R \geq 1$ be an integer. Then we can approximate

$$\sum_j \bar{\mu}_j w_j x_j \approx W \sum_{r=1}^R 2^{-r} y_r, \quad (105)$$

$$y_r = 0 \text{ or } 1, \quad 1 \leq r \leq R. \quad (106)$$

The error entailed in (105) is less than $2^{-R}W$. Using this idea, we can approximate (101) with the system

$$L_i \leq \frac{\pi_i}{\theta} \left(\nu - \sum_j \bar{\mu}_j x_j \right) + \frac{\delta_i}{\theta} \sum_{r=1}^R \eta_{i,r} + \left(\frac{\pi_i^u}{\theta} \sum_j \bar{\mu}_j x_j \right) \sum_{k=i+1}^K z_k \quad (107)$$

$$\text{for } 1 \leq r \leq R, \quad 0 \leq \eta_{i,r} \leq 2^{-r} W \pi_i^U y_r, \quad (108)$$

$$\eta_{ir} - 2^{-r} W \pi_i \leq 0, \quad (109)$$

$$y_r = 0 \text{ or } 1. \quad (110)$$

Here, the η_{ir} and y_r are added variables. Each variable η_{ir} approximates the product $\pi_i \sum_j \bar{\mu}_j w_j x_j$. Our approach is to replace each inequality (101) with the corresponding system (107)-(110), and to add the constraint

$$W \sum_{r=1}^R 2^{-r} y_r - \sum_j \bar{\mu}_j w_j x_j \leq W 2^{-R}. \quad (111)$$

The last term in (107) is needed – together with (102) this implies that if (say) $z_q = 1$ and $i < q$, then $L_i = 0$. To see that this approach is correct, note that we have formulated the adversarial problem as a maximization problem. At an optimal solution, the L_i , and therefore, the η_{ir} , will be set as large as possible. Thus, if we set $y_r = 1$, we will also set $\eta_{ir} = 2^{-r} W \pi_i$, and so we will have $\sum_{r=1}^R \eta_{i,r} = \pi_i W \sum_{r=1}^R 2^{-r} y_r$, as desired.

This process introduces $R(K + 1)$ new variables. We will be using $R = 20$, and thus the growth in problem size is modest. As for the quantity W , we simply compute an upper bound on $\sum_j \bar{\mu}_j u_j x_j$, for all feasible x , which can be done by solving a linear program over the constraints $Ax \geq b$. If, as frequently occurs, we have the constraint $\sum_j x_j = 1$, then we can use $W = \max_j \bar{\mu}_j u_j$.

Clearly, a similar “powers of two” approach could have been used to handle (104). However, we prefer the approach chosen above because (among other reasons) constraint (89) yields a tighter (faster to solve) formulation.

3.1.5 Strengthening the implementor formulations

In our implementation of the implementor-adversary algorithm for the random problem we incorporate a modification which is motivated by the fact that the implementor problem is an SOCP, and as a result comparatively slow (and sometimes quite slow) to solve.

As a result, it may prove desirable to strengthen the formulation of the implementor problem, even at the cost of having a larger formulation, if the result is that the number of iterations of the overall algorithm is significantly reduced.

We first show how to do so for the AVaR case. Recall that at iteration t of the implementor-adversary algorithm we add cut (75), repeated here for convenience:

$$L \geq \nu - \sum_j \bar{\mu}_j \left(1 - \delta_{q(t)} w_j^{(t)}\right) x_j, \quad (112)$$

where $(\pi^{(t)}, w^{(t)})$ is optimal solution to the adversarial problem, and $q(t)$ is appropriately defined. This cut can be strengthened so that we obtain an inequality (system) that implies

$$L \geq \nu - \sum_j \bar{\mu}_j \left(1 - \delta_{q(t)} \check{w}_j\right) x_j, \quad 1 \leq t \leq r-1. \quad (113)$$

for *every* vector \check{w} such that $(\pi^{(t)}, \check{w})$ is feasible for the adversarial problem. As before, this amounts to an application of strong LP duality. To do so, note that for any asset vector x ,

$$\text{AVaR}^{\max}(x) \geq \nu - \bar{\mu}^T x + \Delta_{\pi^{(t)}}(x) \quad (114)$$

where $\Delta_{\pi^{(t)}}(x)$ is defined as the value of the linear program

$$\begin{aligned} \max_w \quad & \delta_{q(t)} \sum_j \bar{\mu}_j x_j w_j \\ \text{Subject to:} \quad & \end{aligned} \quad (115)$$

$$\sum_{j \in T_h} \bar{\mu}_j w_j \leq \frac{\Gamma_h}{E(\delta)}, \quad \text{for each tier } h \quad (116)$$

$$0 \leq w_j \leq u_j, \quad \text{for all } j \quad (117)$$

where $E(\delta) = \sum_{i=1}^K \delta_i \pi_i$. Using strong linear programming duality, there exist values α_h (for each tier h) and β_j (for each asset j), such that

$$\text{AVaR}^{\max}(x) + \bar{\mu}^T x - \sum_h \frac{\Gamma_h}{E(\delta)} \alpha_h - \sum_j u_j \beta_j \geq \nu \quad (118)$$

$$\bar{\mu}_j \sum_{h: j \in T_h} \alpha_h + \beta_j - \delta_{q(t)} \bar{\mu}_j x_j \geq 0, \quad \text{for all } j \quad (119)$$

$$\alpha, \beta \geq 0 \quad (120)$$

Note that here both α and β depend on $\pi^{(t)}$. Based on these observations, we modify the implementor-adversary template, by adding, at each iteration t , two new sets of nonnegative variables, $\alpha^{(t)}$, and $\beta^{(t)}$, and the corresponding system (118)-(120). Thus, we do not only add constraints to the implementor problem – we add $n + H$ new variables as well, in each iteration. Nevertheless this cost is well-worth the price in that the strengthening we obtain significantly reduces the combinatorial complexity of the robust optimization problem.

In the CVaR case, the approach is similar, but instead of (118) and (120) we add

$$\text{CVaR}^{\max}(x) + \frac{\sum_{i=q}^K \pi^{(t)}}{\theta} \bar{\mu}^T x - \sum_h \frac{\Gamma_h}{E(\delta)} \alpha_h - \sum_j u_j \beta_j \geq \frac{\sum_{i=q}^K \pi^{(t)}}{\theta} \nu, \quad \text{and} \quad (121)$$

$$\bar{\mu}_j \sum_{h: j \in T_h} \alpha_h + \beta_j - \frac{\sum_{i=q}^K \pi^{(t)} \delta_i}{\theta} \bar{\mu}_j x_j \geq 0, \quad \text{for all } j, \quad (122)$$

respectively.

4 Computational tests with the random model

In this section we present our results with the AVaR and CVaR models. In our implementation we rely in the solvers included in [CP10], in particular the SOCP and MIP solvers. A significant experimental fact (as of this writing) is that although both solvers are, overall, highly effective and stable, they nevertheless can experience numerical difficulties and may fail to converge to their default tolerances. When the optimal min-max AVaR or CVaR is close to zero, the solvers may have difficulty converging to a truly “optimal” solution. Admittedly, such a situation is not so interesting in that, from a practitioner’s point of view, the fact that AVaR (resp., CVaR) is close to zero would by itself already be useful. In any case, a heuristic fix to this problem is to scale all data so that the optimal portfolio returns are at least of the order of 1.

First we describe our results with the AVaR computations. To control termination of our strengthened algorithm, we choose a tolerance $0 < \tau < 1$, and terminate as soon as either

- $\text{AVaR}^U - \text{AVaR}^L < \tau \text{AVaR}^U$, or
- $\text{AVaR}^U - \text{AVaR}^L < \tau \nu$,

where AVaR^U and AVaR^L are the upper and lower bounds proved on the min-max value-at-risk. We used $\tau = 1.0\text{e-}04$, and $N = 10000$.

Our first set of tests is summarized in Table 8. The parameter ϵ in the definition of the robust optimization problem (see eq. (67)) was set as $\epsilon = 0.02|\Psi^*|$. To construct the benchmark parameter ν in the definition of AVaR (see eq. (3.2)) we used the formula $\nu = 0.5r^*$, where r^* is the return value attained by the optimal solution to the nominal quadratic optimization problem (6)-(7). The probability threshold θ was set at 0.05.

We used a distribution on $K = 100$ points, with $\delta_i = 1/K$, $0 \leq i \leq K - 1$. Figure 5 shows the upper and lower bounds π^u and π^l used in the experiment. To construct these values, we first computed the probability mass function with

$$P(j) = \frac{1/j}{\sum_{i=1}^{99} 1/i}, \quad 1 \leq j \leq 99,$$

and set $P(0) = 1 - \sum_{j=1}^{99} P(j)$. We then randomly perturbed $P(j)$ up and down, to obtain π_j^u and π_j^l . The perturbations were constructed so that $\pi_j^u \approx 2\pi_j^l$.

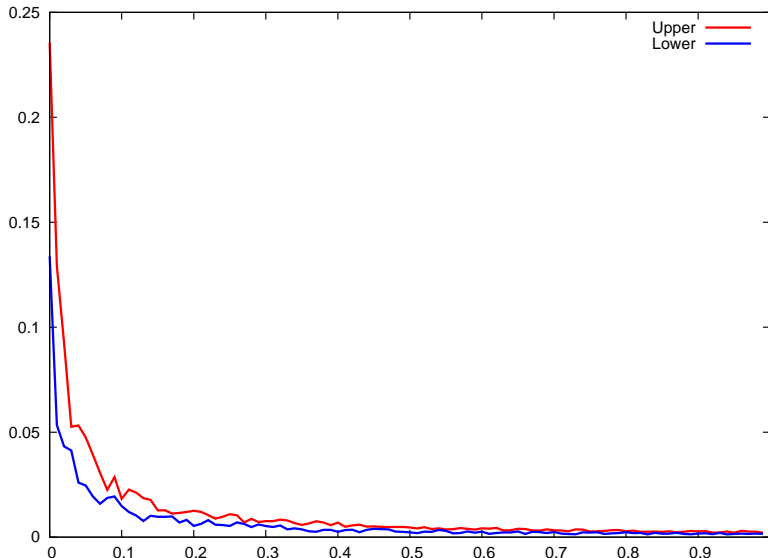


Figure 5: Upper and lower bounds on probability distribution (for Table 8)

Figure 6 shows the values $\sum_{i=0}^j \pi_i^u$ and $\sum_{i=0}^j \pi_i^l$, and the smallest value of $q \approx 0.26$ is the smallest value for which the 95% confidence level can be achieved. That is to say, for any $q \geq 0.26$ there is

a probability distribution π that the adversary could choose, with $\sum_{i=q}^{99} \geq 0.05$ but $\sum_{i=q+1}^{99} < 0.05$. Clearly, the data in this case afford the adversary great latitude.

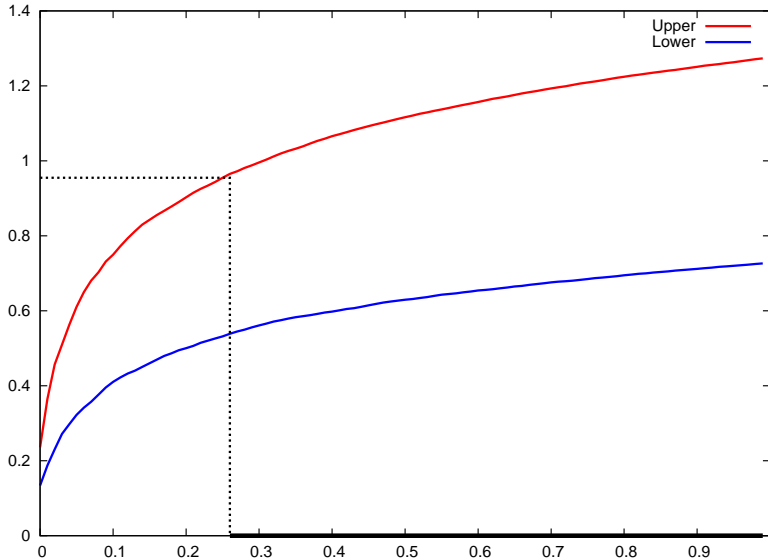


Figure 6: Upper and lower bounds on cumulative mass function (for Table 8)

For $1 \leq j \leq n$ set $u_j = 1$. Finally, we used three tiers, consisting of the top three return deciles, allowing each tier to lose up to 10% of the sum of returns. In Table 8 we display the results of the tests. Here “time” is the overall time in seconds (this includes all implementor and adversarial problems, plus solving the nominal quadratic program), “iters” is the number of iterations of the (strengthened) implementor-adversary algorithm, “impt” and “advnt” is the cumulative time spent solving implementor and adversarial problems. $\text{adj}\tau$ is the correction to the *relative* error that our algorithm incurs, as per Lemma 3.9 – using the notation of Lemma, we keep track of the minimum integer g that we encounter in any iteration of the algorithm.

	A	B	C	D	E	F	G	H	I
time	1.98	1.12	2.68	5.02	2.47	2.03	26.51	36.88	38.32
iters	2	2	2	2	2	2	2	2	2
impt	0.25	0.26	0.3	2.25	0.54	1.07	14.09	26.99	19.90
advnt	1.26	0.45	1.92	1.14	1.32	0.24	2.17	0.95	1.47
adj τ	2.8e-04	2.3e-04	2.5e-04	2.4e-04	3.0e-04	2.5e-04	4.7e-05	1.5e-04	2.1e-04

Table 8: Algorithm performance – AVaR computations

We note the overall fast computational time, the very small number of iterations, and the dominance of the time spent solving SOCPs over the time spent solving MIPs. These results are typical.

Next we consider the impact of the probability threshold θ on AVaR. Figure 7 shows the result of tests on data set A, using the same parameters as in Table 8, but with ν equal to the return at the optimal solution to the nominal QP, and using different values of θ (as the abscissa). Thus, AVaR seems a fairly smooth function of θ .

Finally we consider the impact of the near-optimality parameter ϵ (c.f. (67)) on AVaR. Figure 7 shows the result of tests on data set A, using the same parameters as in Table 8, but for different values of ϵ . Figure 8 shows the outcome of the tests.

Next we describe results using CVaR models. In our implementation we used $R = 20$ (see eq. (105)), and a termination criterion similar to that for the AVaR case. In Table 9, we present results using the same data as for Table 8, using $\tau = 1.0e-03$, and $N = 500$ (cf. (87)-(90)).

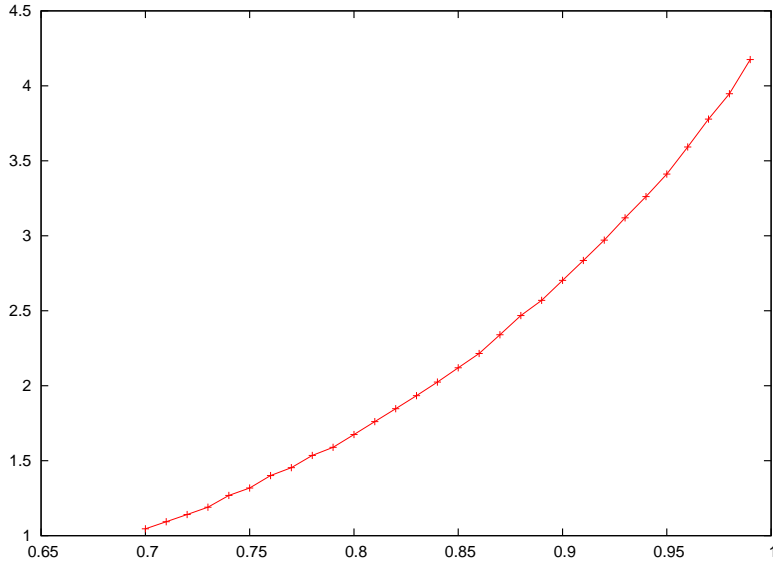


Figure 7: AVaR as a function of confidence level

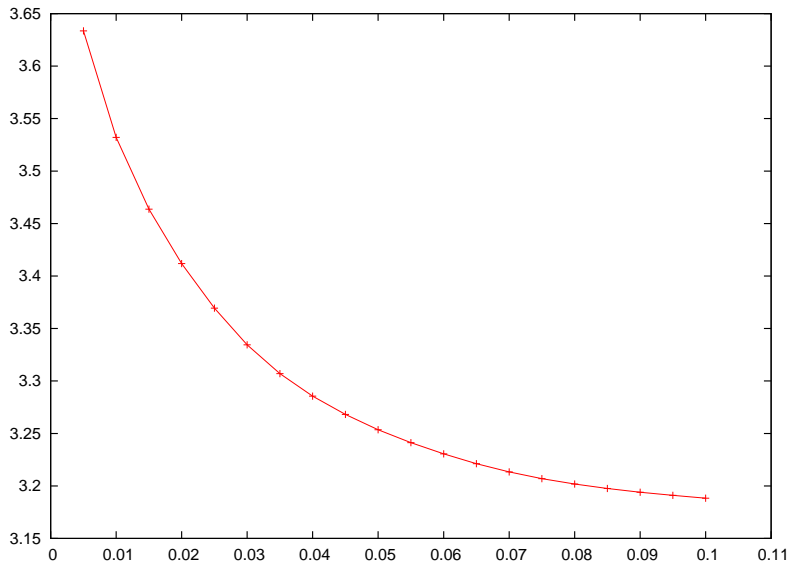


Figure 8: AVaR as a function of nominal suboptimality

In Table 9, “time”, “iters”, “impt” and “advrt” describe, respectively, the CPU time (in seconds), number of iterations, cumulative implementor time and adversarial time required by the algorithm. The row labeled “gap” describes the relative gap between the lower and upper bounds on CVaR obtained by the algorithm, while “apperr” is the relative gap between the true CVaR attained by the solution to the adversarial problem computed in the last iteration, and its approximation (via eq. (105)) that the algorithm actually computes. The results in this table are typical: our algorithm appears quite effective, requiring few iterations to achieve numerical convergence.

To construct a more elaborate test to stress the algorithm, we consider the case where the random variable δ is sampled *conditional* to being in a certain range of interest. To do so, we set $K = 201$, use the deviations

$$\delta_i = 0.05 + (i - 1)/1000, \quad 1 \leq i \leq K,$$

(thus the smallest deviation is 0.05 and the largest, 0.25), construct the values

$$p_i = \frac{1/i^2}{\sum_{j=1}^K 1/j^2}, \quad 1 \leq i \leq K,$$

	A	B	C	D	E	F	G	H	I
time	7.10	7.58	12.10	14.11	6.23	11.45	33.13	47.83	88.43
iters	2	2	3	2	2	2	2	2	3
impt	0.16	0.22	0.65	1.72	1.18	0.66	9.56	32.56	52.13
adv	6.72	7.20	11.25	10.67	4.74	10.33	12.2	6.1	23.85
gap	9.8e-04	8.5e-04	9.2e-04	2.2e-05	7.3e-05	5.1e-05	3.2e-05	1.2e-04	1.3e-04
apperr	2.3e-04	3.0e-05	5.1e-04	2.2e-05	2.4e-04	1.6e-05	1.0e-04	4.0e-04	2.2e-04

Table 9: Algorithm performance – CVaR computations

and set

$$\pi_i^l = .9p_i, \pi_i^u = 1.1p_i, 1 \leq i \leq K.$$

In this case we do not have a heavy-tailed distribution, but the highest probability points (near zero) have been cut-off. For the tests using this distribution, we used $\nu = 0.99r^*$, where r^* is the return value attained by the optimal solution to the nominal quadratic program, $\theta = 0.05$ and $\epsilon = 0.02$. We set $N = 1000, R = 20$ and $\tau = 1.0e-03$. Table 10 shows the results.

	A	B	C	D	E	F	G	H	I
time	10.41	9.85	25.0	11.78	11.26	15.29	45.04	46.92	33.82
iters	2	2	2	2	2	2	2	2	2
impt	0.15	0.16	0.15	1.68	0.9	0.87	19.74	12.75	11.7
adv	9.96	9.39	24.54	8.22	9.7	13.54	14.66	8.77	7.35
gap	4.8e-07	3.5e-06	1.3e-04	4.2e-06	7.0e-06	2.9e-05	4.2e-05	1.5e-05	7.9e-06
apperr	9.8e-06	9.5e-06	5.0e-06	6.0e-05	1.3e-05	1.3e-05	6.4e-05	1.1e-04	4.0e-04

Table 10: CVaR algorithm performance, conditional distribution

We can see that the running times are significantly faster and the accuracy greater than with a heavy-tailed distribution, even though we are dealing with a larger sample space (i.e. $K = 201$ while in Table 9 we used $K = 100$).

In the next set of tests we measure the impact on AVaR and CVaR of variations in the parameters π_i^l, π_i^u . We constructed data sets where, for $0 \leq i \leq 99$:

$$\pi_i^l = \max \left\{ \frac{1/(i+1)}{\sum_{j=1}^{100} 1/j} - \Delta, 0 \right\}, \quad \pi_i^u = \frac{1/(i+1)}{\sum_{j=1}^{100} 1/j} + \Delta. \quad (123)$$

Here Δ is a control parameter, which we set at 0.0005, 0.001, 0.00015, \dots , 0.08. In Figure 9, we show the outcome of runs on data set A, where the robust model (other than the π^l, π^u) is as above. For these runs we used $\tau = 1.0e-04$. Both curves show non-convexities; it may be tempting to attribute these to roundoff errors. However all of these runs were performed by setting $N = 10^4$ in the approximation (87)-(90), and in the AVaR case we additionally verified all values by running our algorithm both with and without the strengthening given in Section 3.1.5. A better explanation, in the CVaR case, concerns its definition (3.5): the critical parameter q can change abruptly. For example, for $\Delta = 0.00015$ at termination we have $q = 81$, whereas for $\Delta = 0.002$ we have $q = 86$. Overall, the CVaR curve appears much smoother, which is not surprising. In both cases, the rate of growth (as a function of Δ) appears sublinear.

In Figure 10 we further explore the run for $\Delta = 0.005$ from Figure 9. To construct this figure we took the the optimal adversarial π_i , at the last iteration of our algorithm, and computed the values $\sum_{i \geq k} \pi_i$, for $90 \leq k \leq 99$. In other words, for each value $.9 \leq \delta^* < 1$ we computed $P(\delta \geq \delta^*)$; both for the AVaR and CVaR case. We can see that the distributions achieve the significance level of 0.05 at $\delta = 0.92$, but after that the CVaR distribution has more mass (which would not help the adversary in the AVaR case).

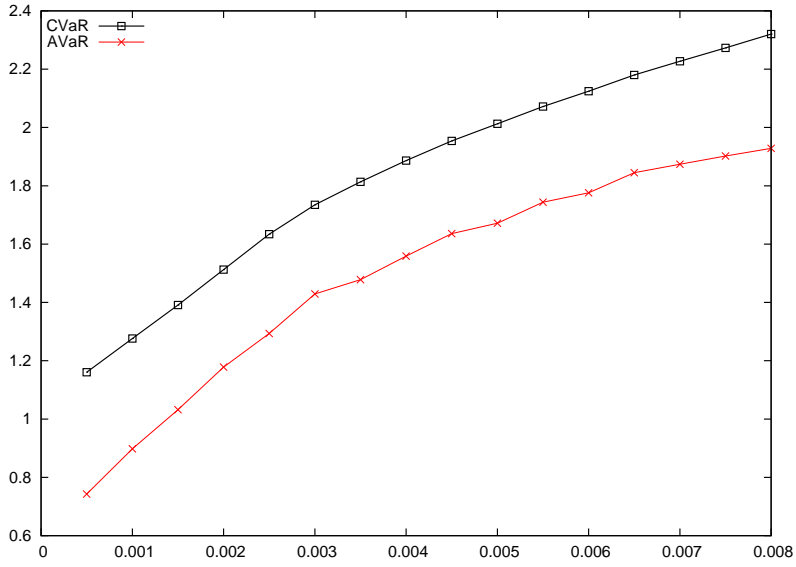


Figure 9: AVaR and CVaR as a function of data perturbations

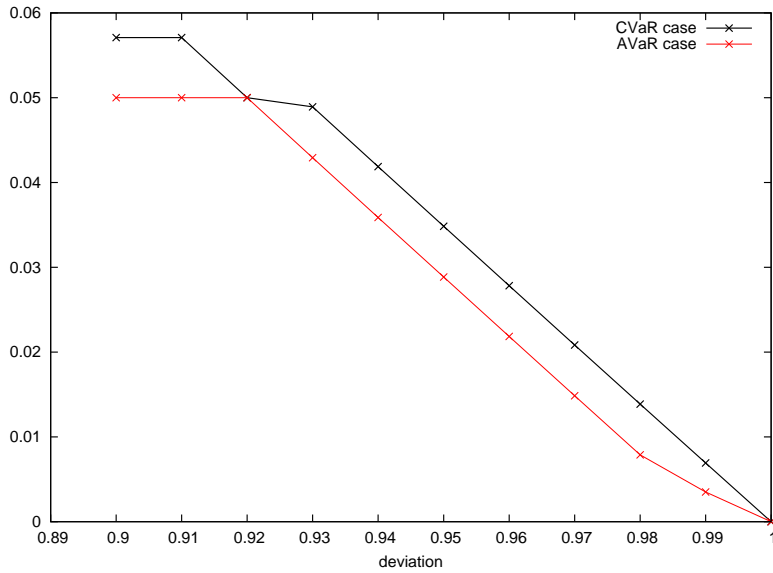


Figure 10: $P(\delta \geq \text{deviation})$ for optimal adversarial distribution

The two distributions essentially agree for $0 \leq i \leq 92$, i.e. for $0 \leq \delta_i \leq .92$. In fact, both distributions remain significant at the 95% level until $\delta_i = 0.92$. In other words, for a given asset vector x , and adversarial vector w , both distributions would produce the same value-at-risk. However, the CVaR distribution shifts some probability mass to values of $0.8 < \delta < 0.92$. In a sense, the CVaR distribution reflects a more sophisticated adversary.

For our final set of experiments, we incorporate a suggestion of B.R. Barmish [B06]. We repeat the CVaR runs described in Figure 9, with $0.005 \leq \Delta \leq 0.05$, except that we impose on the adversary the additional constraint

$$\pi_{i+1} \leq \pi_i, \quad 1 \leq i < K, \quad (124)$$

in order to enforce monotonicity. Figure 11 shows the results. Clearly, non-monotonicity can result in far more conservative models.

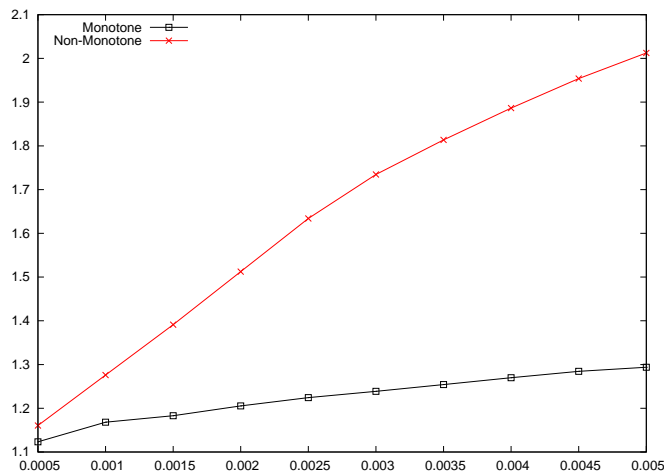


Figure 11: Impact of monotonicity on CVaR

5 Conclusion

The cutting-plane methods discussed in this paper seem well-capable of handling realistic uncertainty models with explicit non-convexities. We expect that such models will prove useful in many contexts besides portfolio optimization.

An important point regarding our implementations is that we did not resort to any low-level methodologies for speeding up the individual iterations (such as early termination of the adversarial problems or implementor problems) or smoothing techniques (see [BI06], [Ne05], [N04]) designed to reduce the number of iterations. The application of such techniques is likely to drastically reduce running times.

In future work we will take up both issues.

Acknowledgement. We thank B.R. Barmish, A. Ben-Tal and R. Tütüncü for helpful comments.

References

- [AT02] C. Acerbi and D. Tasche, On the coherence of expected shortfall, *Journal of Banking and Finance* **26** (2002), 1487–1503.
- [AT04] C. Acerbi, Coherent representations of subjective risk aversion, in *Risk Measures for the 21st century* (G. Szegö, Ed.), Wiley, New York (2004)
- [AZ05] A. Atamtürk and M. Zhang, Two-Stage Robust Network Flow and Design under Demand Uncertainty, Research Report BCOL.04.03, Dept. of IEOR, University of California at Berkeley (2004).
- [B06] B.R. Barmish, personal communication.
- [B62] J.F. Benders, Partitioning procedures for solving mixed variables programming problems, *Numerische Mathematik* **4** (1962) 238-252.
- [BBN06] A. Ben-Tal, S. Boyd and A. Nemirovski, Extending Scope of Robust Optimization: Comprehensive Robust Counterparts of Uncertain Problems, *Math. Program.* **107**, (2006) 63 – 89.
- [BGGN04] A Ben-Tal, A. Goryashko, E Guslitzer and A. Nemirovski, Adjusting robust solutions of uncertain linear programs, *Mathematical Programming* **99**(2) (2004), 351 – 376.
- [BGNV05] A. Ben-Tal, B. Golany, A. Nemirovski and J.-P. Vial, Retailer-Supplier Flexible Commitments Contracts: A Robust Optimization Approach. *MSOM* **7** (2005), 248 – 271.

- [BN98] A. Ben-Tal and A. Nemirovski, Robust convex optimization, *Mathematics of Operations Research* **23** (1998), 769 – 805.
- [BN99] A. Ben-Tal and A. Nemirovski, Robust solutions of uncertain linear programs, *Operations Research Letters* **25** (1999), 1-13.
- [BN00] A. Ben-Tal and A. Nemirovski, Robust solutions of linear programming problems contaminated with uncertain data, *Mathematical Programming Series A* **88** (2000), 411 – 424.
- [BPS03] D. Bertsimas, D. Pachamanova, and M. Sim. Robust Linear Optimization under General Norms, *Operations Research Letters* **32** (2003) 510–516.
- [BS03] D. Bertsimas and M. Sim, The price of robustness, *Operations Research* **52** (2003), 35 – 53.
- [BT06] D. Bertsimas and A. Thiele, A robust optimization approach to inventory theory, *Oper. Research* **54** (2006), 150 – 168.
- [B96] D. Bienstock, Computational study of a family of mixed-integer quadratic programming problems, *Math. Programming* **74** (1996), 121–140.
- [BI06] D. Bienstock and I. Iyengar, Faster approximation algorithms for covering and packing problems, *SIAM J. Computing* **35** (2006) 825–854.
- [BKRSW96] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan and D. P. Williamson, Adversarial queueing theory, *Proceedings of the 28th Annual ACM Symposium on Theory of Computing* (1996), 376 – 385.
- [CC05] G. Calafiore and M. C. Campi, Uncertain convex programs: Randomized solutions and confidence levels, *Math. Programming* (2005), 25 – 46.
- [CS06] S. Ceria and R. Stubbs, Incorporating estimation errors into portfolio selection: Robust portfolio construction, *Journal of Asset Management*, Volume 7 **2** (2006) 109 – 127.
- [CP10] Ilog Cplex v. 10.
- [DFN02] I.R. de Farias JR. and G.L. Nemhauser, Polyhedral Study of the Cardinality Constrained Knapsack Problem, *Mathematical Programming* **96** (2003) 439-467.
- [EGL97] L. El Ghaoui and H. Lebret, Robust solutions to least-squares problems with uncertain data. *SIAM J. Matrix Anal. Appl.* **18** (1997), 1035-1064.
- [EGOL98] L. El Ghaoui, F. Oustry and H. Lebret, Robust solutions to uncertain semidefinite programs, *SIAM J. Optim.* **9** (1998) 33–52.
- [EI04] E. Erdogan and G. Iyengar, Ambiguous chance constrained problems and robust optimization, CORC Report TR-2004-10, Columbia University (2004).
- [GI04] D. Goldfarb and G. Iyengar, Robust portfolio selection problems, *Math. of Oper. Res.* **28** (2003), 1 – 38.
- [G75] F. Glover, Improved Linear Integer Programming Formulations of Nonlinear Integer Problems, *Mgt. Science* **22** (1975), 455 – 460.
- [GF04] C. Goncalves and F. Ferreira, Tail Risk and p_K -tail risk, Working Paper, ISCTE Business School, Lisbon (2004)
- [GLS93] M. Grötschel, L. Lovász and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag. 1993.
- [GM93] G. Gallego and I. Moon, The distribution free newsboy problem: Review and extensions, *Journal of Operations Research Society* **44** (1993), 825 – 834.

- [I05] G. Iyengar, Robust Dynamic Programming, *Math. of OR* **30** (2005), 257 – 280.
- [KPU02] P. Krokmal, J. Palmquist and S. Uryasev, Portfolio Optimization with Conditional Value-At-Risk Objective and Constraints, *The Journal of Risk* **4** (2002), 11 – 27.
- [LVBL85] M. Lobo, L. Vandenberghe, S. Boyd and H. Lebret, Applications of Second-order Cone Programming', *Linear Algebra and its Application* **284** (1985), 193 - 228.
- [M52] H.M. Markowitz, Portfolio selection, *J. Finance* **7** (1952) 77-91.
- [M59] H.M. Markowitz, *Portfolio selection*. Wiley, New York (1959).
- [MG94] I. Moon and G. Gallego, Distribution free procedures for some inventory models, *Journal of Operations Research Society* **45** (1994), 651 – 658.
- [N04] A. Nemirovski, Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems, *SIAM J. on Optimization* **15** (2004) 229–251.
- [NW88] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York (1988).
- [Ne05] Y. Nesterov, Smooth minimization of non-smooth functions, *Math. Programming* **B** (2005) 127–152.
- [RU00] R.T. Rockafellar and S. Uryasev, Optimization of Conditional Value-at-Risk, *Journal of Risk* **2** (2000), 21 - 41.
- [S58] H. Scarf, A min-max solution of an inventory problem, *Studies in the Mathematical Theory of Inventory and Production* (Chapter 12) (1958), Stanford University Press, Stanford, CA.
- [T05] A.Thiele, Robust dynamic optimization: a distribution-free approach. Manuscript (2005).
- [T06] R. Tütüncü, personal communication.
- [TK04] R. Tütüncü and M. Koenig, Robust Asset Allocation, *Annals of Operations Research* **132** (2004), 157 – 187.

A Appendix

A.1 The adversarial problem for the histogram model

In this Section we consider the adversarial problem (17-26) in Section 2.2.2, given a vector x of asset weights. We assume $\Omega = 0..n$ The adversarial problem can be stated as:

$$V^* = \max \quad \sum_j x_j \delta_j \quad (125)$$

Subject to:

$$\bar{\mu}_j \left(\sum_{i=1}^K \gamma_{i-1} y_{ij} \right) \leq \delta_j \leq \bar{\mu}_j \left(\sum_{i=1}^K \gamma_i y_{ij} \right) \quad (126)$$

$$\sum_{i=1}^K y_{ij} \leq 1, \quad \forall j \quad (127)$$

$$n_i \leq \sum_j y_{ij} \leq N_i, \quad 1 \leq i \leq K \quad (128)$$

$$\sum_{j \in T_h} \delta_j \leq (1 - \Lambda_h) \sum_{j \in T_h} \bar{\mu}_j, \quad 1 \leq h \leq H \quad (129)$$

$$y_{ij} = 0 \text{ or } 1, \text{ all } i, j \quad (130)$$

For convenience, we restate the result to be proved here.

Theorem. Suppose $x \geq 0$ and $\bar{\mu} \geq 0$. For every fixed K and H , and for every $\epsilon > 0$, there is an algorithm that finds a solution to problem (125-130) with optimality relative error $\leq \epsilon$, in time polynomial in ϵ^{-1} and n .

Our approach is a simple adaptation of the classical technique for approximating a knapsack problem.

Definition A.1 A class is an inclusion-maximal set $S \subseteq \{1, 2, \dots, n\}$, such that for all $j \in S$ and $j' \in S$,

$$\{1 \leq h \leq H : j \in T_h\} = \{1 \leq h \leq H : j' \in T_h\}.$$

Note: there are at most $2^{|H|}$ classes.

Definition A.2 Let $(\hat{\delta}, \hat{y})$ be a feasible solution to (125-130). We let

$$L(\hat{\delta}, \hat{y}) = \left\{ 1 \leq j \leq n : \bar{\mu}_j \left(\sum_{i=1}^K \gamma_{i-1} \hat{y}_{ij} \right) < \hat{\delta}_j < \bar{\mu}_j \left(\sum_{i=1}^K \gamma_i \hat{y}_{ij} \right), \text{ for some } 1 \leq i \leq K \right\}.$$

Note that if $j \notin L(\hat{\delta}, \hat{y})$ then either $\hat{y}_{ij} = 0$ for all i , in which case $\hat{\delta}_j = 0$ (by (126)), or, without loss of generality, for some $1 \leq i \leq K$, $\hat{y}_{ij} = 1$ and $\hat{\delta}_j = \bar{\mu}_j \gamma_i$.

Lemma A.3 Let $(\hat{\delta}, \hat{y})$ be an optimal solution to (125-130). Then without loss of generality $|L(\hat{\mu}, \hat{y})| \leq H$.

Proof. Routine, since there are H constraints (129). ■

In what follows we assume that we are given $x \geq 0$ and $0 < \epsilon < 1$. By adding if necessary a new tier h with $\Lambda_h = 0$ we can assume that $\cup_h T_h = \{1, 2, \dots, n\}$. Let $\theta > 0$ be such that $2^{-\theta} n = \epsilon$.

We can now describe our algorithm. This algorithm enumerates a certain number of cases (later shown to be polynomial in ϵ^{-1} and n). Each case corresponds to searching for a solution to (125-130) with a certain structure. In each case the algorithm either constructs such a solution, or it shows that no solution with that structure exists. In the last step of the algorithm, the best enumerated solution is output. We will first state the algorithm, then comment upon it, then analyze its complexity, and finally prove that it solves the approximation problem to desired accuracy.

Step 0. The cases we enumerate are as follows. First, we choose a subset $L \subseteq \{1, 2, \dots, n\}$ with $|L| \leq H$. Next, we choose $\tilde{j} \in \{1, 2, \dots, n\} - L$ and $1 \leq \tilde{i} \leq K$. Finally, we choose a mapping f that assigns to each $j \in L$ an integer $1 \leq f(j) \leq K$. For each such combination of choices $(L, \tilde{j}, \tilde{i},$ and $f)$ we run the following steps:

Step 1. For each $j \notin L$ and each $1 \leq i \leq K$ write

$$w_{ij} = \left[2^\theta \frac{x_j \bar{\mu}_j \gamma_i}{x_{\tilde{j}} \bar{\mu}_{\tilde{j}} \gamma_i} \right].$$

Step 2. For each class S , each vector of integers $Q = (q_1, q_2, \dots, q_K)$ with $0 \leq q_i \leq N_i$ for each i , and each integer W with $0 \leq W \leq 2^\theta |S - L|$, solve the problem

$$P^{(S, Q, W)} : \min \sum_{j \in S - L} \sum_{i=1}^K \bar{\mu}_j \gamma_i y_{ij}$$

Subject to:

$$\sum_{i=1}^K y_{ij} \leq 1, \quad \forall j \in S - L$$

$$\sum_{j \in S-L} \sum_{i=1}^K y_{ij} = q_i, \quad \forall 1 \leq i \leq K,$$

$$\sum_{j \in S-L} \sum_{i=1}^K w_{ij} y_{ij} = W$$

$$y_{ij} = 0 \text{ or } 1, \quad \forall j \in S - L, \text{ and } \forall i,$$

$$y_{i\tilde{j}} = 1, \text{ if } \tilde{j} \in S. \quad (131)$$

$$y_{ij} = 0, \quad \forall i \text{ with } x_j \bar{\mu}_j \gamma_i > x_{\tilde{j}} \bar{\mu}_{\tilde{j}} \gamma_i. \quad (132)$$

[**Note:** A simple dynamic programming procedure for this step will be described below].

Assuming this problem is feasible, let $y^{(S,Q,W)}$ be an optimal solution, and construct a vector $\delta^{(S,Q,W)}$ with an entry for each $j \in S - L$ by setting

$$\delta_j^{(S,Q,W)} = \bar{\mu}_j \gamma_i y_{ij}^{(S,Q,W)} \quad \forall j \in S - L, \text{ and } \forall 1 \leq i \leq K.$$

Step 3. Let \mathcal{C} denote the number of classes, and denote by S^s the s^{th} -class. For every combination of a \mathcal{C} -tuple of the form $(Q^1, W^1), (Q^2, W^2), \dots, (Q^{\mathcal{C}}, W^{\mathcal{C}})$, where for $1 \leq s \leq \mathcal{C}$, $P(S^s, Q^s, W^s)$ is feasible, we solve the following linear program (where f is the mapping chosen in Step 0):

$$\hat{v}(L, f) = \max \quad \sum_{j \in L} x_j \delta_j \quad (133)$$

Subject to:

$$\bar{\mu}_j \gamma_{f(j)-1} \leq \delta_j \leq \bar{\mu}_j \gamma_{f(j)} \quad (134)$$

$$\sum_{j \in L \cap T_h} \delta_j \leq (1 - \Lambda_h) \left(\sum_{j \in T_h} \bar{\mu}_j \right) - \sum_{s=1}^{\mathcal{C}} \sum_{j \in S^s \cap T_h} \delta_j^{(S^s, Q^s, W^s)}, \quad 1 \leq h \leq H \quad (135)$$

Step 3b. Suppose this LP is feasible, with optimal solution $\hat{\delta}$. For each $j \in L$, write $\hat{y}_{f(j)j} = 1$ and $\hat{y}_{ij} = 0$ for every $i \neq f(j)$. If the vector (y, δ) obtained by combining all the vectors $(y^{(S^s, Q^s, W^s)}, \delta^{(S^s, Q^s, W^s)})$, and the vector $(\hat{y}, \hat{\delta})$, is feasible for problem (125-130), then we record the value

$$\sum_{s=1}^{\mathcal{C}} \sum_{j \in S^s - L} x_j \delta_j^{(S^s, Q^s, W^s)} + \hat{v}(L, f).$$

Step 4. Output the largest value computed in Step 4b.

Next we comment on the algorithm. In Step 0 we choose a set $L \subseteq \{1, 2, \dots, n\}$ with $|L| \leq H - 1$ this hinges on Lemma A.3. Thus, in Step 2 we enforce that each δ_j for $j \notin L$ is either zero or equal to $\bar{\mu}_j \gamma_i$ for some i . Likewise, in Step 0 we enumerate a pair of indices \tilde{j} and \tilde{i} - we fix $\delta_{\tilde{j}} = \bar{\mu}_{\tilde{j}} \gamma_{\tilde{i}}$ (constraint (131)) and furthermore, we enforce the rule that $x_{\tilde{j}} \delta_{\tilde{j}} = \max_{j \notin L} \{x_j \delta_j\}$ (constraint (132)).

Having chosen the set L and the indices \tilde{j} and \tilde{i} , the algorithm searches for a solution with the properties just stated (call such a solution, a $(L, \tilde{j}, \tilde{i})$ -solution) that is near optimal. In Step 2 the algorithm computes the minimum value $\sum_{j \in S-L} \delta_j$ achieved by a $(L, \tilde{j}, \tilde{i})$ -solution whose sum of w -weights over $S-L$ is exactly W . Note that the w_{ij} are, approximately, scaled values $x_j \bar{\mu}_j \gamma_i$. So, approximately, in Step 2 we are computing the minimum value $\sum_{j \in S-L} \delta_j$ that can be achieved while insisting that the objective function $\sum_{j \in S-L} x_j \delta_j$ equals a prescribed value. Finally, in Step 3 we enumerate all possibilities for the variables in L .

Now we turn to the complexity of the algorithm. Assume that we are considering a fixed choice of L, \tilde{j} , and \tilde{i} .

For a given class S , we can solve all problems of the form $P^{(S,Q,W)}$ (i.e., for all Q and W) with a single pass of a dynamic programming procedure. This procedure solves problems similar $P^{(S,Q,W)}$, but *restricted* to the first t variables of $S-L$, for $t = 1, 2, \dots, |S-L|$. The complexity of this procedure is at most $O(|S-L| n^K 2^\theta n) = O(n^{H+1} 2^\theta n)$. Thus, the total workload incurred in Step 2 during the entire run of the algorithm is

$$O(n^{K+H+2} 2^\theta).$$

Concerning Step 3, note that the total number of \mathcal{C} -tuples that are enumerated is at most $(n^K 2^\theta n)^{\mathcal{C}} < (n^{K+1} 2^\theta)^{2^H}$, and that the total number of mappings f is at most H^K . Thus the total number of LPs solved in Step 3 is at most

$$(n^{K+1} 2^\theta)^{2^H} K^H.$$

In summary, the complexity of the algorithm is

$$O((n^{K+2} 2^\theta)^{2^H} K^H \mathcal{L}(H)) = O((n^{K+3} \epsilon^{-1})^{2^H} K^H \mathcal{L}(H)),$$

where $\mathcal{L}(H)$ is the complexity of solving a linear program with H variables and $2H$ constraints.

This upper bound can be improved, although it should remain exponential in K and H . The upper bound can also be improved in special cases. For example, if the tiers are noncrossing (e.g. for all h and h' , either T_h and $T_{h'}$ are disjoint or one contains the other) then the 2^H dependence can be improved to just H .

Finally, we consider the validity of the algorithm. Consider an optimal solution $(\hat{\delta}, \hat{y})$ to problem (125-130). Writing $L = L(\hat{\delta}, \hat{y})$ by Lemma A.3 we may assume $|L| \leq H$. Without loss of generality, we may assume that for each $j \notin L$, either $\hat{\delta}_j = 0$ or $\hat{\delta}_j = \bar{\mu}_j \gamma_i$ for some i (and in the latter case $\hat{y}_{ij} = 1$).

Let $1 \leq \tilde{i} \leq K$, $\tilde{j} \notin L$ be such that $x_{\tilde{j}} \hat{\delta}_{\tilde{j}} = \max_{j \notin L} \{x_j \hat{\delta}_j\}$. Consider a given class S , and write $W^S = \sum_{j \in S-L} \sum_i w_{ij} \hat{y}_{ij}$. Let Q^S be the vector with entries $(q_1^S, q_2^S, \dots, q_K^S)$ where for $1 \leq i \leq K$, $q_i^S = \sum_{j \in S-L} \hat{y}_{ij}$. Clearly, problem $P^{(S,Q,W)}$ is feasible, and by construction in Step 2,

$$\sum_{j \in S-L} \delta_j^{(S,Q,W)} \leq \sum_{j \in S-L} \sum_{i=1}^K \bar{\mu}_j \gamma_i \hat{y}_{ij} = \sum_{j \in S-L} \hat{\delta}_j. \quad (136)$$

Furthermore,

$$\frac{\sum_{j \in S-L} x_j \delta_j^{(S,Q,W)}}{x_{\tilde{j}} \bar{\mu}_{\tilde{j}} \gamma_{\tilde{i}}} = \frac{\sum_{j \in S-L} \sum_i x_j \bar{\mu}_j \gamma_i y_{ij}^{(S,Q,W)}}{x_{\tilde{j}} \bar{\mu}_{\tilde{j}} \gamma_{\tilde{i}}} \quad (137)$$

$$= 2^{-\theta} \sum_{j \in S-L} \sum_i \frac{2^\theta x_j \bar{\mu}_j \gamma_i}{x_{\tilde{j}} \bar{\mu}_{\tilde{j}} \gamma_{\tilde{i}}} y_{ij}^{(S,Q,W)} \quad (138)$$

$$\geq 2^{-\theta} \sum_{j \in S-L} \sum_i (w_{ij} - 1) y_{ij}^{(S,Q,W)} \quad (139)$$

$$\geq 2^{-\theta} \left(\sum_{j \in S-L} \sum_i w_{ij} y_{ij}^{(S,Q,W)} - |S-L| \right) \quad (140)$$

$$= 2^{-\theta} W^S - 2^{-\theta} |S-L| \quad (141)$$

$$= 2^{-\theta} \sum_{j \in S-L} \sum_i w_{ij} \hat{y}_{ij} - 2^{-\theta} |S-L| \quad (142)$$

$$\geq \sum_{j \in S-L} \sum_i \frac{x_j \bar{\mu}_j \gamma_i}{x_j \bar{\mu}_j \gamma_i} \hat{y}_{ij} - 2^{-\theta} |S-L| \quad (143)$$

$$= \frac{\sum_{j \in S-L} x_j \hat{\delta}_j}{x_j \bar{\mu}_j \gamma_i} - 2^{-\theta} |S-L|. \quad (144)$$

In other words,

$$\sum_{j \in S-L} x_j \hat{\delta}_j - \sum_{j \in S-L} x_j \delta_j^{(S,Q,W)} \leq 2^{-\theta} |S-L| (x_j \bar{\mu}_j \gamma_i). \quad (145)$$

Adding (145) over all classes S_s , we obtain

$$\sum_{s=1}^{\mathcal{C}} \sum_{j \in S^s-L} x_j \hat{\delta}_j - \sum_{s=1}^{\mathcal{C}} \sum_{j \in S^s-L} x_j \delta_j^{(S^s, Q^s, W^s)} \leq 2^{-\theta} (n-H) (x_j \bar{\mu}_j \gamma_i) \leq \quad (146)$$

$$\leq 2^{-\theta} (n-H) \sum_j x_j \hat{\delta}_j \leq \epsilon \sum_j x_j \hat{\delta}_j \quad (\text{by choice of } \theta). \quad (147)$$

Finally, for each $j \in L$ let $f(j)$ be that index i such that

$$\bar{\mu}_j \left(\sum_{i=1}^K \gamma_{i-1} \hat{y}_{ij} \right) < \hat{\delta}_j < \bar{\mu}_j \left(\sum_{i=1}^K \gamma_i \hat{y}_{ij} \right).$$

By construction in Step 3, and (136) we have that

$$\sum_{j \in L} x_j \hat{\delta}_j \leq \hat{v}(L, f). \quad (148)$$

Now (147) and (148) yield the desired result, thus proving the Theorem. ■