Experiments in Robust Optimization

Daniel Bienstock

Columbia University

10-26-06
Robust Optimization

- Optimization under parameter (data) uncertainty
- Ben-Tal and Nemirovsky, El Ghaoui et al
- Bertsimas et al

Uncertainty is modeled by assuming that data is not known precisely, and will instead lie in known sets.

Example: a coefficient $a_i$ is uncertain. We allow $a_i \in [l_i, u_i]$.

Typically, a minimization problem becomes a min-max problem.
Robust Optimization

- Optimization under parameter (data) uncertainty
- Ben-Tal and Nemirovsky, El Ghaoui et al
- Bertsimas et al
- Uncertainty is modeled by assuming that data is not known precisely, and will instead lie in known sets.
- Example: a coefficient $a_i$ is uncertain. We allow $a_i \in [l_i, u_i]$.
- Typically, a minimization problem becomes a min-max problem.
Robust Optimization

- Optimization under parameter (data) uncertainty
- Ben-Tal and Nemirovsky, El Ghaoui et al
- Bertsimas et al

Uncertainty is modeled by assuming that data is not known precisely, and will instead lie in known sets.

- Example: a coefficient $a_i$ is uncertain. We allow $a_i \in [l_i, u_i]$.
- Typically, a minimization problem becomes a min-max problem.
Robust Optimization

- Optimization under parameter (data) uncertainty
- Ben-Tal and Nemirovsky, El Ghaoui et al
- Bertsimas et al

Uncertainty is modeled by assuming that data is not known precisely, and will instead lie in known sets.

- Example: a coefficient $a_i$ is uncertain. We allow $a_i \in [l_i, u_i]$.
- Typically, a minimization problem becomes a min-max problem.
Robust Optimization

- Optimization under parameter (data) uncertainty
- Ben-Tal and Nemirovsky, El Ghaoui et al
- Bertsimas et al

Uncertainty is modeled by assuming that data is not known precisely, and will instead lie in known sets.

Example: a coefficient $a_i$ is uncertain. We allow $a_i \in [l_i, u_i]$.

Typically, a minimization problem becomes a min-max problem.
Robust Optimization

- Optimization under parameter (data) uncertainty
- Ben-Tal and Nemirovsky, El Ghaoui et al
- Bertsimas et al

Uncertainty is modeled by assuming that data is not known precisely, and will instead lie in known sets.

Example: a coefficient \( a_i \) is uncertain. We allow \( a_i \in [l_i, u_i] \).

Typically, a **minimization** problem becomes a **min-max** problem.
Example: Linear Programs with Row-Wise uncertainty
Ben-Tal and Nemirovsky, 1999

\[
\begin{align*}
\text{min } & c^t x \\
\text{Subject to: } & Ax \geq b \quad \text{for all } A \in \mathcal{U} \\
& x \in X
\end{align*}
\]

\[\mathcal{U} = \text{uncertainty set}\]
\[\rightarrow \text{the } i^{th} \text{ row of } A \text{ belongs to an ellipsoidal set } \mathcal{E}_i\]
e.g. \[\sum_j \alpha_{ij}^2 (a_{ij} - \bar{a}_{ij})^2 \leq 1\]

\[\rightarrow \text{can be solved using SOCP techniques}\]
Other forms of optimization under uncertainty

- Stochastic programming
- Adversarial queueing, online optimization
- “Risk-aware” optimization
- Optimization of utility functions as a substitute for handling infeasibilities
Other forms of optimization under uncertainty

- Stochastic programming
- Adversarial queueing, online optimization
- “Risk-aware” optimization
- Optimization of utility functions as a substitute for handling infeasibilities
Other forms of optimization under uncertainty

- Stochastic programming
- Adversarial queueing, online optimization
- “Risk-aware” optimization
- Optimization of utility functions as a substitute for handling infeasibilities
Other forms of optimization under uncertainty

- Stochastic programming
- Adversarial queueing, online optimization
- “Risk-aware” optimization
- Optimization of utility functions as a substitute for handling infeasibilities
**Scenario I: Stability**

Data is fairly accurate, though possibly noisy – small errors are possible

→ Idiosyncratic decisions and small changes in data could have major impact
Scenario I: Stability
Data is fairly accurate, though possibly noisy – small errors are possible

→ Idiosyncratic decisions and small changes in data could have major impact
Scenario II: Hedging

Significant, but within order-of-magnitude, data uncertainty

Example:
A certain parameter, $\alpha$, is volatile. Its long-term average is 1.5 but it we could expect changes of the order of .3.

- Possibly more than just noise
- Could use deviations to our advantage, especially if there are several uncertain parameters that act “correlated”
- Are we guarding against risk or are we hedging?
Scenario III: Insurance
Real world data can exhibit undesirable and unexpected behavior

- Classical goal: how can we protect without becoming too risk averse
- Need to clearly spell out desired tradeoff between risk and performance
- Magnitude and geometry of risk are not the same
Scenario III: Insurance
Real world data can exhibit undesirable and unexpected behavior

- Classical goal: how can we protect without becoming too risk averse
- Need to clearly spell out desired tradeoff between risk and performance
- **Magnitude** and **geometry** of risk are not the same
Application: Portfolio Optimization

\[
\min \lambda x^T Q x - \mu^T x
\]

Subject to:

\[
Ax \geq b
\]

- \( \mu = \) vector of “returns”, \( Q = \) “covariance” matrix
- \( x = \) vector of “asset weights”
- \( Ax \geq b \): general linear constraints
- \( \lambda \geq 0 = \) “risk-aversion” multiplier
Application: Portfolio Optimization

\[ \min \lambda x^T Q x - \mu^T x \]

Subject to:

\[ Ax \geq b \]

- \( \mu \) = vector of “returns”, \( Q \) = “covariance” matrix
- \( x \) = vector of “asset weights”
- \( Ax \geq b \): general linear constraints
- \( \lambda \geq 0 \) = “risk-aversion” multiplier
Application: Portfolio Optimization

\[ \min \lambda x^T Q x - \mu^T x \]
Subject to:

\[ Ax \geq b \]

- \( \mu \) = vector of “returns”, \( Q \) = “covariance” matrix
- \( x \) = vector of “asset weights”
- \( Ax \geq b \): general linear constraints
- \( \lambda \geq 0 \) = “risk-aversion” multiplier
Application: Portfolio Optimization

\[ \min \lambda x^T Q x - \mu^T x \]

Subject to:

\[ Ax \geq b \]

- \( \mu = \) vector of “returns”, \( Q = \) “covariance” matrix
- \( x = \) vector of “asset weights”
- \( Ax \geq b \): general linear constraints
- \( \lambda \geq 0 = \) “risk-aversion” multiplier
Application: Portfolio Optimization

\[ \min \lambda x^T Q x - \mu^T x \]

Subject to:

\[ Ax \geq b \]

- \( \mu \) = vector of “returns”, \( Q \) = “covariance” matrix
- \( x \) = vector of “asset weights”
- \( Ax \geq b \): general linear constraints
- \( \lambda \geq 0 \) = “risk-aversion” multiplier
Robust Portfolio Optimization
Goldfarb and Iyengar, 2001

→ $Q$ and $\mu$ are uncertain

Robust Problem

$$\min_x \left\{ \max_{Q \in Q} \lambda x^T Q x - \min_{\mu \in \mathcal{E}} \mu^T x \right\}$$

Subject to:

$$\sum_j x_j = 1, \quad x \geq 0$$

→ When $Q$ is an ellipsoid and $\mathcal{E}$ is a product of intervals the robust problem can be solved as an SOCP
Linear Programs with Row-Wise uncertainty

Bertsimas and Sim, 2002

\[
\begin{align*}
\text{max } & \quad c^t x \\
\text{Subject to: } & \quad Ax \leq b \quad \text{for all } A \in \mathcal{U} \\
x & \geq 0
\end{align*}
\]

→ in every row i at most $\Gamma_i$ coefficients can change:

\[
\hat{a}_{ij} - \delta_{ij} \leq a_{ij} \leq \hat{a}_{ij} + \delta_{ij}
\]

→ for all other coefficients: $a_{ij} = \hat{a}_{ij}$.

Robust problem can be formulated as a (larger) linear program.
Linear Programs with Row-Wise uncertainty

Bertsimas and Sim, 2002

\[
\begin{align*}
\text{max} & \quad c^T x \\
\text{Subject to:} & \quad Ax \leq b \quad \text{for all } A \in \mathcal{U} \\
x & \geq 0
\end{align*}
\]

\[
\begin{align*}
\rightarrow \text{ in every row } i \text{ at most } \Gamma_i \text{ coefficients can change:} & \\
\hat{a}_{ij} - \delta_{ij} & \leq a_{ij} \leq \hat{a}_{ij} + \delta_{ij}
\end{align*}
\]

\[
\begin{align*}
\rightarrow \text{ for all other coefficients: } & \\
a_{ij} = \hat{a}_{ij}.
\end{align*}
\]

Robust problem can be formulated as a (larger) linear program.
Linear Programs with Row-Wise uncertainty
Bertsimas and Sim, 2002

\[ \max \ c^T x \]

Subject to:
\[ A x \leq b \quad \text{for all } A \in U \]
\[ x \geq 0 \]

→ in every row \( i \) at most \( \Gamma_i \) coefficients can change:
\[ \hat{a}_{ij} - \delta_{ij} \leq a_{ij} \leq \hat{a}_{ij} + \delta_{ij} \]

→ for all other coefficients: \( a_{ij} = \hat{a}_{ij} \).

Robust problem can be formulated as a (larger) linear program.
Linear Programs with Row-Wise uncertainty
Bertsimas and Sim, 2002

\[ \text{max } c^t x \]
\[ \text{Subject to:} \]
\[ Ax \leq b \quad \text{for all } A \in \mathcal{U} \]
\[ x \geq 0 \]

→ in every row i at most \( \Gamma_i \) coefficients can change:
\[ \hat{a}_{ij} - \delta_{ij} \leq a_{ij} \leq \hat{a}_{ij} + \delta_{ij} \]

→ for all other coefficients: \( a_{ij} = \hat{a}_{ij} \).

Robust problem can be formulated as a (larger) linear program.
Equivalent formulation

\[
\text{max } c^t x \\
\text{Subject to: } A x \leq b \quad \text{for all } A \in \mathcal{U} \\
x \geq 0
\]

→ in every row i exactly \( \Gamma_i \) coefficients change: \( a_{ij} = \hat{a}_{ij} + \delta_{ij} \)

→ for all other coefficients: \( a_{ij} = \hat{a}_{ij} \).
Robust Portfolio Optimization

A different uncertainty model

→ Want to model that deviations of the returns $\mu_j$ from their nominal values are rare but could be significant

A simple example

- Parameters: $0 \leq \gamma \leq 1$, integer $N \geq 0$, for each asset $j$:
  $\bar{\mu}_j =$ expected return, $0 \leq \delta_j$ small (possibly zero)

- Well-behaved asset $j$: $\bar{\mu}_j - \delta_j \leq \mu_j \leq \bar{\mu}_j + \delta_j$

- Misbehaving asset $j$: $(1 - \gamma)\bar{\mu}_j \leq \mu_j \leq \bar{\mu}_j$

- At most $N$ assets misbehave
Robust Portfolio Optimization

A different uncertainty model

→ Want to model that deviations of the returns $\mu_j$ from their nominal values are rare but could be significant

A simple example

- Parameters: $0 \leq \gamma \leq 1$, integer $N \geq 0$, for each asset $j$:
  $\mu_j = \text{expected return}$, $0 \leq \delta_j$ small (possibly zero)

- Well-behaved asset $j$: $\mu_j - \delta_j \leq \mu_j \leq \mu_j + \delta_j$

- Misbehaving asset $j$: $(1 - \gamma)\mu_j \leq \mu_j \leq \mu_j$

- At most $N$ assets misbehave
Robust Portfolio Optimization

A different uncertainty model

→ Want to model that deviations of the returns $\mu_j$ from their nominal values are rare but could be significant

A simple example

- Parameters: $0 \leq \gamma \leq 1$, integer $N \geq 0$, for each asset $j$:
  
  $\bar{\mu}_j =$ expected return, $0 \leq \delta_j$ small (possibly zero)

- Well-behaved asset $j$: $\bar{\mu}_j - \delta_j \leq \mu_j \leq \bar{\mu}_j + \delta_j$

- Misbehaving asset $j$: $(1 - \gamma)\bar{\mu}_j \leq \mu_j \leq \bar{\mu}_j$

- At most $N$ assets misbehave
Robust Portfolio Optimization

A different uncertainty model

→ Want to model that deviations of the returns \( \mu_j \) from their nominal values are rare but could be significant

A simple example

- Parameters: \( 0 \leq \gamma \leq 1 \), integer \( N \geq 0 \), for each asset \( j \):
  \( \bar{\mu}_j = \text{expected return}, \ 0 \leq \delta_j \text{ small (possibly zero)} \)

- Well-behaved asset \( j \): \( \bar{\mu}_j - \delta_j \leq \mu_j \leq \bar{\mu}_j + \delta_j \)

- Misbehaving asset \( j \): \( (1 - \gamma)\bar{\mu}_j \leq \mu_j \leq \bar{\mu}_j \)

- At most \( N \) assets misbehave
A more comprehensive setting

- Parameters: $0 \leq \gamma_1 \leq \gamma_2 \leq \ldots \leq \gamma_K \leq 1$, integers $0 \leq n_i \leq N_i, \ 1 \leq i \leq K$

  for each asset $j$: $\bar{\mu}_j = \text{expected return}$

- Between $n_i$ and $N_i$ assets $j$ satisfy:

$$(1 - \gamma_i)\bar{\mu}_j \leq \mu_j \leq (1 - \gamma_{i-1})\bar{\mu}_j, \text{ for each } i \geq 1 \quad (\gamma_0 = 0)$$
A more comprehensive setting

- Parameters: \( 0 \leq \gamma_1 \leq \gamma_2 \leq \ldots \leq \gamma_K \leq 1 \), integers \( 0 \leq n_i \leq N_i, \quad 1 \leq i \leq K \)

for each asset \( j \): \( \bar{\mu}_j = \) expected return

- between \( n_i \) and \( N_i \) assets \( j \) satisfy:

\[
(1 - \gamma_i)\bar{\mu}_j \leq \mu_j \leq (1 - \gamma_{i-1})\bar{\mu}_j, \quad \text{for each} \quad i \geq 1 \quad (\gamma_0 = 0)
\]
A more comprehensive setting

- Parameters: \(0 \leq \gamma_1 \leq \gamma_2 \leq \ldots \leq \gamma_K \leq 1\), integers \(0 \leq n_i \leq N_i, \ 1 \leq i \leq K\)
  for each asset \(j\): \(\bar{\mu}_j = \text{expected return}\)

- between \(n_i\) and \(N_i\) assets \(j\) satisfy:
  \[(1 - \gamma_i)\bar{\mu}_j \leq \mu_j \leq (1 - \gamma_{i-1})\bar{\mu}_j\]

- \(\sum_j \mu_j \geq \Gamma \sum_j \bar{\mu}_j; \ \Gamma > 0\) a parameter

- (R. Tütüncü) For \(1 \leq h \leq H\),
  - a set (“tier”) \(T_h\) of assets, and a parameter \(\Gamma_h > 0\)
  for each \(h\), \(\sum_{j \in T_h} \mu_j \geq \Gamma_h \sum_{j \in S_h} \bar{\mu}_j\)

Note: only downwards changes are modeled
A more comprehensive setting

- Parameters: \(0 \leq \gamma_1 \leq \gamma_2 \leq \ldots \leq \gamma_K \leq 1\), integers \(0 \leq n_i \leq N_i, \ 1 \leq i \leq K\)
  for each asset \(j\): \(\bar{\mu}_j = \text{expected return}\)

- between \(n_i\) and \(N_i\) assets \(j\) satisfy:
  
  \[
  (1 - \gamma_i)\bar{\mu}_j \leq \mu_j \leq (1 - \gamma_{i-1})\bar{\mu}_j
  \]

- \(\sum_j \mu_j \geq \Gamma \sum_j \bar{\mu}_j; \ \Gamma > 0\) a parameter

- (R. Tütüncü) For \(1 \leq h \leq H\),
  
  a set (“tier”) \(T_h\) of assets, and a parameter \(\Gamma_h > 0\)

  for each \(h\), \(\sum_{j \in T_h} \mu_j \geq \Gamma_h \sum_{j \in S_h} \bar{\mu}_j\)

Note: only downwards changes are modeled
A more comprehensive setting

Parameters: \(0 \leq \gamma_1 \leq \gamma_2 \leq \ldots \leq \gamma_K \leq 1\), integers \(0 \leq n_i \leq N_i\), \(1 \leq i \leq K\) for each asset \(j\): \(\bar{\mu}_j\) = expected return

between \(n_i\) and \(N_i\) assets \(j\) satisfy:
\[(1 - \gamma_i)\bar{\mu}_j \leq \mu_j \leq (1 - \gamma_{i-1})\bar{\mu}_j\]

\[\sum_j \mu_j \geq \Gamma \sum_j \bar{\mu}_j; \quad \Gamma > 0\] a parameter

(R. Tütüncü) For \(1 \leq h \leq H\),

- a set (“tier”) \(T_h\) of assets, and a parameter \(\Gamma_h > 0\)

for each \(h\), \(\sum_{j \in T_h} \mu_j \geq \Gamma_h \sum_{j \in S_h} \bar{\mu}_j\)

Note: only downwards changes are modeled
A more comprehensive setting

- Parameters: \( 0 \leq \gamma_1 \leq \gamma_2 \leq \ldots \leq \gamma_K \leq 1 \), integers \( 0 \leq n_i \leq N_i, \ 1 \leq i \leq K \)
  for each asset \( j \): \( \bar{\mu}_j = \) expected return

- between \( n_i \) and \( N_i \) assets \( j \) satisfy:
  \[ (1 - \gamma_i)\bar{\mu}_j \leq \mu_j \leq (1 - \gamma_{i-1})\bar{\mu}_j \]

- \( \sum_j \mu_j \geq \Gamma \sum_j \bar{\mu}_j; \quad \Gamma > 0 \) a parameter

- (R. Tütüncü) For \( 1 \leq h \leq H \),
  - a set (“tier”) \( T_h \) of assets, and a parameter \( \Gamma_h > 0 \)

  \[ \sum_{j \in T_h} \mu_j \geq \Gamma_h \sum_{j \in S_h} \bar{\mu}_j \]

Note: only downwards changes are modeled
A more comprehensive setting

- Parameters: \(0 \leq \gamma_1 \leq \gamma_2 \leq \cdots \leq \gamma_K \leq 1\), integers \(0 \leq n_i \leq N_i, 1 \leq i \leq K\)
  for each asset \(j\): \(\bar{\mu}_j = \) expected return

- between \(n_i\) and \(N_i\) assets \(j\) satisfy:
  \[(1 - \gamma_i) \bar{\mu}_j \leq \mu_j \leq (1 - \gamma_{i-1}) \bar{\mu}_j\]

- \(\sum_j \mu_j \geq \Gamma \sum_j \bar{\mu}_j; \ \Gamma > 0\) a parameter

- (R. Tütüncü) For \(1 \leq h \leq H\),

  - a set (“tier”) \(T_h\) of assets, and a parameter \(\Gamma_h > 0\)

  for each \(h\), \(\sum_{j \in T_h} \mu_j \geq \Gamma_h \sum_{j \in S_h} \bar{\mu}_j\)

Note: only downwards changes are modeled
General methodology:
Benders’ decomposition (= cutting-plane algorithm)

Generic problem: \( \min_{x \in X} \max_{d \in D} f(x, d) \)

→ Maintain a finite subset \( \tilde{D} \) of \( D \) (a “model”)

GAME

1. Implementor: solve \( \min_{x \in X} \max_{d \in \tilde{D}} f(x, d) \), with solution \( x^* \)

2. Adversary: solve \( \max_{d \in D} f(x^*, d) \), with solution \( \tilde{d} \)

3. Add \( \tilde{d} \) to \( \tilde{D} \), and go to 1.
**General methodology:**

Benders’ decomposition (= cutting-plane algorithm)

Generic problem:  \( \min_{x \in X} \max_{d \in D} f(x, d) \)

\[\rightarrow \text{Maintain a finite subset } \tilde{D} \text{ of } D \text{ (a “model”) }\]

**GAME**

1. Implementor: solve \( \min_{x \in X} \max_{d \in \tilde{D}} f(x, d) \), with solution \( x^* \)

2. Adversary: solve \( \max_{d \in D} f(x^*, d) \), with solution \( \tilde{d} \)

3. Add \( \tilde{d} \) to \( \tilde{D} \), and go to 1.
**General methodology:**

Benders’ decomposition (= cutting-plane algorithm)

Generic problem: \[ \min_{x \in X} \max_{d \in D} f(x, d) \]

→ Maintain a **finite subset** \( \tilde{D} \) of \( D \) (a “model”)

**GAME**

1. **Implementor:** solve \[ \min_{x \in X} \max_{d \in \tilde{D}} f(x, d) \]
   with solution \( x^* \)

2. **Adversary:** solve \[ \max_{d \in \tilde{D}} f(x^*, d) \]
   with solution \( \tilde{d} \)

3. **Add** \( \tilde{d} \) to \( \tilde{D} \), and go to 1.
**General methodology:**

Benders’ decomposition (= cutting-plane algorithm)

Generic problem: \[ \min_{x \in X} \max_{d \in D} f(x, d) \]

→ Maintain a **finite subset** \( \tilde{D} \) of \( D \) (a “model”)

**GAME**

1. **Implementor:** solve \[ \min_{x \in X} \max_{d \in \tilde{D}} f(x, d) \], with solution \( x^* \)

2. **Adversary:** solve \[ \max_{d \in D} f(x^*, d) \], with solution \( \tilde{d} \)

3. Add \( \tilde{d} \) to \( \tilde{D} \), and go to 1.
General methodology:
Benders’ decomposition (= cutting-plane algorithm)

Generic problem: \( \min_{x \in X} \max_{d \in D} f(x, d) \)

→ Maintain a finite subset \( \tilde{D} \) of \( D \) (a “model”)

GAME

1. **Implementor**: solve \( \min_{x \in X} \max_{d \in \tilde{D}} f(x, d) \), with solution \( x^* \)

2. **Adversary**: solve \( \max_{d \in D} f(x^*, d) \), with solution \( \tilde{d} \)

3. Add \( \tilde{d} \) to \( \tilde{D} \), and go to 1.
Why this approach

- Decoupling of implementor and adversary yields considerably simpler, and smaller, problems.

- Decoupling allows us to use more sophisticated uncertainty models.

- If number of iterations is small, implementor’s problem is a small “convex” problem.

- Most progress will be achieved in initial iterations – permits “soft” termination criteria.
Why this approach

- Decoupling of implementor and adversary yields considerably simpler, and smaller, problems

- Decoupling allows us to use more sophisticated uncertainty models

- If number of iterations is small, implementor’s problem is a small “convex” problem

- Most progress will be achieved in initial iterations – permits “soft” termination criteria
Why this approach

- Decoupling of implementor and adversary yields considerably simpler, and smaller, problems

- Decoupling allows us to use more sophisticated uncertainty models

  - If number of iterations is small, implementor’s problem is a small “convex” problem

- Most progress will be achieved in initial iterations – permits “soft” termination criteria
Why this approach

- Decoupling of implementor and adversary yields considerably simpler, and smaller, problems.

- Decoupling allows us to use more sophisticated uncertainty models.

- If number of iterations is small, implementor’s problem is a small “convex” problem.

- Most progress will be achieved in initial iterations – permits “soft” termination criteria.
Why this approach

- Decoupling of implementor and adversary yields considerably simpler, and smaller, problems

- Decoupling allows us to use more sophisticated uncertainty models

- If number of iterations is small, implementor’s problem is a small “convex” problem

- Most progress will be achieved in initial iterations – permits “soft” termination criteria
Implementor’s problem
A convex quadratic program

At iteration $m$, solve

$$\min \lambda x^T Q x - r$$

Subject to:

$$Ax \geq b$$

$$r \leq \mu_{(i)}^T x, \quad i = 1, \ldots, m$$

Here, $\mu(1), \ldots, \mu(m)$ are given return vectors
Adversarial problem: A mixed-integer program

\( x^* = \text{given asset weights} \)

\[
\begin{align*}
\min & \quad \sum_j x_j^* \mu_j \\
\text{Subject to:} & \\
\bar{\mu}_j (1 - \sum_i \gamma_{i-1} y_{ij}) & \leq \mu_j \leq \bar{\mu}_j (1 - \sum_i \gamma_i y_{ij}) & \forall i \geq 1 \\
\sum_i y_{ij} & \leq 1, \quad \forall j \quad \text{(each asset in at most one segment)} \\
n_i & \leq \sum_j y_{ij} \leq N_i, \quad 1 \leq i \leq K \quad \text{(segment cardinalities)} \\
\sum_{j \in \mathcal{T}_h} \mu_j & \geq \Gamma_h \sum_{j \in \mathcal{T}_h} \bar{\mu}_j, \quad 1 \leq h \leq H \quad \text{(tier ineqs.)} \\
\mu_j & \text{ free, } y_{ij} = 0 \text{ or } 1, \quad \forall i, j
\end{align*}
\]
Example: 2464 assets, 152-factor model. CPU time: 500 seconds

10 segments (a: “heavy tail”)
6 tiers: the top five deciles lose at most 10% each, total loss \( \leq 5\% \)
Same run

2464 assets, 152 factors;
10 segments, 6 tiers
## Summary of average problems with 3-4 segments, 2-3 tiers

<table>
<thead>
<tr>
<th>columns</th>
<th>rows</th>
<th>iterations</th>
<th>time (sec.)</th>
<th>imp. time</th>
<th>adv. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>500</td>
<td>20</td>
<td>47</td>
<td>1.85</td>
<td>1.34</td>
</tr>
<tr>
<td>2</td>
<td>500</td>
<td>20</td>
<td>3</td>
<td>0.09</td>
<td>0.01</td>
</tr>
<tr>
<td>3</td>
<td>703</td>
<td>108</td>
<td>1</td>
<td>0.29</td>
<td>0.13</td>
</tr>
<tr>
<td>4</td>
<td>499</td>
<td>140</td>
<td>3</td>
<td>3.12</td>
<td>2.65</td>
</tr>
<tr>
<td>5</td>
<td>499</td>
<td>20</td>
<td>19</td>
<td>0.42</td>
<td>0.21</td>
</tr>
<tr>
<td>6</td>
<td>1338</td>
<td>81</td>
<td>7</td>
<td>0.45</td>
<td>0.17</td>
</tr>
<tr>
<td>7</td>
<td>2019</td>
<td>140</td>
<td>8</td>
<td>41.53</td>
<td>39.6</td>
</tr>
<tr>
<td>8</td>
<td>2443</td>
<td>153</td>
<td>2</td>
<td>12.32</td>
<td>9.91</td>
</tr>
<tr>
<td>9</td>
<td>2464</td>
<td>153</td>
<td>111</td>
<td>100.81</td>
<td>60.93</td>
</tr>
</tbody>
</table>
Why the adversarial problem is “easy”

\[ (x^* = \text{given asset weights}) \]

\[
\begin{align*}
\min & \quad \sum_j x_j^* \mu_j \\
\text{Subject to:} & \\
\bar{\mu}_j (1 - \sum_i \gamma_{i-1} y_{ij}) & \leq \mu_j \leq \bar{\mu}_j (1 - \sum_i \gamma_i y_{ij}) \\
\sum_i y_{ij} & \leq 1, \forall j \quad \text{(each asset in at most one segment)} \\
n_i & \leq \sum_j y_{ij} \leq N_i, \forall i \quad \text{(segment cardinalities)} \\
\sum_{j \in T_h} \mu_j & \geq \Gamma_h (\sum_{j \in T_h} \bar{\mu}_j), \forall h \quad \text{(tier inequalities)} \\
\mu_j & \text{ free, } y_{ij} = 0 \text{ or } 1, \forall i, j
\end{align*}
\]
Why the adversarial problem is “easy”

\( K = \text{no. of segments}, \quad H = \text{no. of tiers} \)

**Theorem.** For every fixed \( K \) and \( H \), and for every \( \epsilon > 0 \), there is an algorithm that finds a solution to the adversarial problem with optimality relative error \( \leq \epsilon \), in time polynomial in \( \epsilon^{-1} \) and \( n \) (= no. of assets).
The simplest case

\[ \max \sum_{j} x^*_j \delta_j \]

Subject to:

\[ \sum_{j} \delta_j \leq \Gamma \]

\[ 0 \leq \delta_j \leq u_j y_j, \ y_j = 0 \text{ or } 1, \ \text{all } j \]

\[ \sum_{j} y_j \leq N \]

\( \cdots \) a cardinality constrained knapsack problem

What is the impact of the uncertainty model

All runs on the same data set with 1338 columns and 81 rows

- 1 segment: (200, 0.5)
  robust random return = 4.57, 157 assets
- 2 segments: (200, 0.25), (100, 0.5)
  robust random return = 4.57, 186 assets
- 2 segments: (200, 0.2), (100, 0.6)
  robust random return = 3.25, 213 assets
- 2 segments: (200, 0.1), (100, 0.8)
  robust random return = 1.50, 256 assets
- 1 segment: (100, 1.0)
  robust random return = 1.24, 281 assets
Ambiguous chance-constrained models

1. The implementor chooses a vector $\mathbf{x}^*$ of assets

2. The adversary chooses a probability distribution $P$ for the returns vector

3. A random returns vector $\mathbf{\mu}$ is drawn from $P$

→ Implementor wants to choose $\mathbf{x}^*$ so as to minimize value-at-risk (conditional value at risk, etc.)


→ We want to model correlated errors in the returns
Ambiguous chance-constrained models

1. The implementor chooses a vector $x^*$ of assets

2. The adversary chooses a *probability distribution* $P$ for the returns vector

3. A random returns vector $\mu$ is drawn from $P$

→ Implementor wants to choose $x^*$ so as to minimize value-at-risk (conditional value at risk, etc.)


→ We want to model *correlated* errors in the returns
Ambiguous chance-constrained models

1. The implementor chooses a vector $x^*$ of assets

2. The adversary chooses a *probability distribution* $P$ for the returns vector

3. A random returns vector $\mu$ is drawn from $P$

→ Implementor wants to choose $x^*$ so as to minimize value-at-risk (conditional value at risk, etc.)


→ We want to model *correlated* errors in the returns
Ambiguous chance-constrained models

1. The implementor chooses a vector \( x^\ast \) of assets
2. The adversary chooses a probability distribution \( P \) for the returns vector
3. A random returns vector \( \mu \) is drawn from \( P \)

Implementor wants to choose \( x^\ast \) so as to minimize value-at-risk (conditional value at risk, etc.)


We want to model correlated errors in the returns
Ambiguous chance-constrained models

1. The implementor chooses a vector $x^*$ of assets

2. The adversary chooses a *probability distribution* $P$ for the returns vector

3. A random returns vector $\mu$ is drawn from $P$

→ Implementor wants to choose $x^*$ so as to minimize *value-at-risk* (conditional value at risk, etc.)


→ We want to model *correlated* errors in the returns
Ambiguous chance-constrained models

1. The implementor chooses a vector \( x^* \) of assets

2. The adversary chooses a *probability distribution* \( P \) for the returns vector

3. A random returns vector \( \mu \) is drawn from \( P \)

→ Implementor wants to choose \( x^* \) so as to minimize *value-at-risk* (conditional value at risk, etc.)


→ We want to model *correlated* errors in the returns
**Uncertainty set**

Given a vector $x^*$ of assets, the adversary

1. Chooses a vector $w \in \mathbb{R}^n$ ($n =$ no. of assets) with $0 \leq w_j \leq 1$ for all $j$.

2. Chooses a random variable $0 \leq \delta \leq 1$

→ Random return: $\mu_j = \bar{\mu}_j (1 - \delta w_j)$ ($\bar{\mu} =$ nominal returns).

**Definition:** Given reals $\nu$ and $0 \leq \theta \leq 1$ the value-at-risk of $x^*$ is the real $\rho \geq 0$ such that

$$\text{Prob}(\nu - \mu^T x^* \geq \rho) \geq \theta$$

→ The adversary wants to maximize VAR
### Uncertainty set

Given a vector $x^*$ of assets, the adversary

1. Chooses a vector $w \in R^n$ ($n =$ no. of assets) with $0 \leq w_j \leq 1$ for all $j$.

2. Chooses a random variable $0 \leq \delta \leq 1$

→ Random return: $\mu_j = \bar{\mu}_j(1 - \delta w_j)$ ($\bar{\mu} =$ nominal returns).

**Definition:** Given reals $\nu$ and $0 \leq \theta \leq 1$ the value-at-risk of $x^*$ is the real $\rho \geq 0$ such that

$$\text{Prob}(\nu - \mu^T x^* \geq \rho) \geq \theta$$

→ The adversary wants to maximize VAR
Uncertainty set

Given a vector \( x^* \) of assets, the adversary

1. Chooses a vector \( w \in \mathbb{R}^n \) (\( n = \) no. of assets) with \( 0 \leq w_j \leq 1 \) for all \( j \).

2. Chooses a random variable \( 0 \leq \delta \leq 1 \)

\[ \rightarrow \text{Random return: } \mu_j = \bar{\mu}_j (1 - \delta w_j) \quad (\bar{\mu} = \text{nominal returns}). \]

**Definition:** Given reals \( \nu \) and \( 0 \leq \theta \leq 1 \) the value-at-risk of \( x^* \) is the real \( \rho \geq 0 \) such that

\[ \text{Prob}(\nu - \mu^T x^* \geq \rho) \geq \theta \]

\[ \rightarrow \text{The adversary wants to maximize } \text{VAR} \]
Uncertainty set

Given a vector \( x^* \) of assets, the adversary

1. Chooses a vector \( w \in \mathbb{R}^n \) \((n = \text{no. of assets})\) with \( 0 \leq w_j \leq 1 \) for all \( j \).

2. Chooses a random variable \( 0 \leq \delta \leq 1 \)

\[ \rightarrow \text{Random return: } \mu_j = \bar{\mu}_j (1 - \delta w_j) \quad (\bar{\mu} = \text{nominal returns}) \]

Definition: Given reals \( \nu \) and \( 0 \leq \theta \leq 1 \) the value-at-risk of \( x^* \) is the real \( \rho \geq 0 \) such that

\[ \text{Prob}(\nu - \mu^T x^* \geq \rho) \geq \theta \]

\[ \rightarrow \text{The adversary wants to maximize } \text{VAR} \]
Uncertainty set

Given a vector $x^*$ of assets, the adversary

1. Chooses a vector $w \in \mathbb{R}^n$ (n = no. of assets) with $0 \leq w_j \leq 1$ for all j.

2. Chooses a random variable $0 \leq \delta \leq 1$

→ Random return: $\mu_j = \bar{\mu}_j (1 - \delta w_j)$ ($\bar{\mu}$ = nominal returns).

Definition: Given reals $\nu$ and $0 \leq \theta \leq 1$ the value-at-risk of $x^*$ is the real $\rho \geq 0$ such that

$$\text{Prob}(\nu - \mu^T x^* \geq \rho) \geq \theta$$

→ The adversary wants to maximize VAR
Uncertainty set

Given a vector $\mathbf{x}^*$ of assets, the adversary

1. Chooses a vector $\mathbf{w} \in \mathbb{R}^n$ (n = no. of assets) with $0 \leq w_j \leq 1$ for all $j$.

2. Chooses a random variable $0 \leq \delta \leq 1$

→ Random return: $\mu_j = \bar{\mu}_j (1 - \delta w_j)$ ($\bar{\mu}$ = nominal returns).

**Definition:** Given reals $\nu$ and $0 \leq \theta \leq 1$ the value-at-risk of $\mathbf{x}^*$ is the real $\rho \geq 0$ such that

$$\text{Prob}(\nu - \mu^T \mathbf{x}^* \geq \rho) \geq \theta$$

→ The adversary wants to maximize VAR
Given a vector $x^*$ of assets, the adversary

1. Chooses a vector $w \in \mathbb{R}^n$ (n = no. of assets) with $0 \leq w_j \leq W$ for all $j$.

2. Chooses a random variable $0 \leq \delta \leq 1$

→ Random return: $\mu_j = \bar{\mu}_j - \delta w_j$ ($\bar{\mu}$ = nominal returns).

**Definition:** Given reals $\nu$ and $0 \leq \theta \leq 1$ the value-at-risk of $x^*$ is the real $\rho \geq 0$ such that

$$\text{Prob}(\nu - \mu^T x^* \geq \rho) \geq \theta$$

→ The adversary wants to maximize VAR.
Given a vector $\mathbf{x}^*$ of assets, the adversary

1. Chooses a vector $\mathbf{w} \in \mathbb{R}^n$ ($n =$ no. of assets) with $0 \leq w_j \leq W$ for all $j$.

2. Chooses a random variable $0 \leq \delta \leq 1$

→ Random return: $\mu_j = \bar{\mu}_j - \delta w_j$ ($\bar{\mu} =$ nominal returns).

**Definition:** Given reals $\nu$ and $0 \leq \theta \leq 1$ the value-at-risk of $\mathbf{x}^*$ is the real $\rho \geq 0$ such that

$$\text{Prob}(\nu - \mu^T \mathbf{x}^* \geq \rho) \geq \theta$$

→ The adversary wants to maximize VAR
The classical factor model for returns

\[ \mu = \bar{\mu} + V^T f + \epsilon \]

where

- \( \bar{\mu} \) = expected return,
- \( V \) = “factor exposure matrix”,
- \( f \) = a bounded random variable,
- \( \epsilon \) = residual errors

\( V \) is \( r \times n \) with \( r << n \).
Adversarial problem

\[
\text{Random return}_j = \bar{\mu}_j (1 - \delta w_j) \quad \text{where} \quad 0 \leq w_j \leq 1 \quad \forall j, \quad \text{and} \quad 0 \leq \delta \leq 1 \quad \text{is a random variable.}
\]

A discrete distribution:

- We are given \textbf{fixed} values \(0 = \delta_0 \leq \delta_2 \leq \ldots \leq \delta_K = 1\)
  
  \begin{align*}
  \text{example:} & \quad \delta_i = \frac{i}{K} \\
  \text{Adversary chooses} & \quad \pi_i = \text{Prob}(\delta = \delta_i), \quad 0 \leq i \leq K \\
  \text{The } \pi_i \text{ are constrained:} & \quad \text{we have fixed bounds, } \pi^l_i \leq \pi_i \leq \pi^u_i \\
  \text{(and possibly other constraints)} \end{align*}

- Tier constraints: for sets ("tiers") \(T_h\) of assets, \(1 \leq h \leq H\), we require:
  
  \begin{align*}
  E(\delta \sum_{j \in T_h} w_j) & \leq \Gamma_h \quad \text{(given)} \\
  \text{or,} & \quad (\sum_i \delta_i \pi_i) \sum_{j \in T_h} w_j \leq \Gamma_h
  \end{align*}

- Cardinality constraint: \(w_j > 0\) for at most \(N\) indices \(j\)
Random return \( j \) = \( \bar{\mu}_j(1 - \delta w_j) \) where \( 0 \leq w_j \leq 1 \) \( \forall j \), and \( 0 \leq \delta \leq 1 \) is a random variable.

A discrete distribution:

- We are given fixed values \( 0 = \delta_0 \leq \delta_2 \leq \ldots \leq \delta_K = 1 \)
  - example: \( \delta_i = \frac{i}{K} \)
- Adversary chooses \( \pi_i = \text{Prob}(\delta = \delta_i) \), \( 0 \leq i \leq K \)
- The \( \pi_i \) are constrained: we have fixed bounds, \( \pi_i^l \leq \pi_i \leq \pi_i^u \)
  (and possibly other constraints)
- Tier constraints: for sets ("tiers") \( T_h \) of assets, \( 1 \leq h \leq H \), we require:
  \[ E(\delta \sum_{j \in T_h} w_j) \leq \Gamma_h \] (given)

  or, \( (\sum_i \delta_i \pi_i) \sum_{j \in T_h} w_j \leq \Gamma_h \)

- Cardinality constraint: \( w_j > 0 \) for at most \( N \) indices \( j \)
Random return$_j = \bar{\mu}_j(1 - \delta w_j)$ where $0 \leq w_j \leq 1 \ \forall \ j$, and $0 \leq \delta \leq 1$ is a random variable.

A discrete distribution:

- We are given fixed values $0 = \delta_0 \leq \delta_2 \leq \ldots \leq \delta_K = 1$
  
  example: $\delta_i = \frac{i}{K}$

- Adversary chooses $\pi_i = \text{Prob}(\delta = \delta_i)$, $0 \leq i \leq K$

- The $\pi_i$ are constrained: we have fixed bounds, $\pi_i^l \leq \pi_i \leq \pi_i^u$ (and possibly other constraints)

- Tier constraints: for sets (“tiers”) $T_h$ of assets, $1 \leq h \leq H$, we require:
  
  $E(\delta \sum_{j \in T_h} w_j) \leq \Gamma_h$ (given)

  or, $(\sum_i \delta_i \pi_i) \sum_{j \in T_h} w_j \leq \Gamma_h$

- Cardinality constraint: $w_j > 0$ for at most $N$ indices $j$
Random return \( j \) = \( \bar{\mu}_j (1 - \delta w_j) \) where \( 0 \leq w_j \leq 1 \ \forall \ j \), and \( 0 \leq \delta \leq 1 \) is a random variable.

A discrete distribution:

- We are given fixed values \( 0 = \delta_0 \leq \delta_2 \leq \ldots \leq \delta_K = 1 \)
  - example: \( \delta_i = \frac{i}{K} \)
- Adversary chooses \( \pi_i = \text{Prob}(\delta = \delta_i), 0 \leq i \leq K \)
- The \( \pi_i \) are constrained: we have fixed bounds, \( \pi^l_i \leq \pi_i \leq \pi^u_i \)
  (and possibly other constraints)
- Tier constraints: for sets ("tiers") \( T_h \) of assets, \( 1 \leq h \leq H \), we require:
  \[
  E(\delta \sum_{j \in T_h} w_j) \leq \Gamma_h \quad \text{(given)}
  \]
  or, \( (\sum_i \delta_i \pi_i) \sum_{j \in T_h} w_j \leq \Gamma_h \)
- Cardinality constraint: \( w_j > 0 \) for at most \( N \) indices \( j \)
→ Random return \( j \) = \( \bar{\mu}_j (1 - \delta w_j) \) where \( 0 \leq w_j \leq 1 \) \( \forall j \), and \( 0 \leq \delta \leq 1 \) is a random variable.

A discrete distribution:

- We are given fixed values \( 0 = \delta_0 \leq \delta_2 \leq \ldots \leq \delta_K = 1 \)
  example: \( \delta_i = \frac{i}{K} \)
- Adversary chooses \( \pi_i = \text{Prob}(\delta = \delta_i) \), \( 0 \leq i \leq K \)
- The \( \pi_i \) are constrained: we have fixed bounds, \( \pi_i^l \leq \pi_i \leq \pi_i^u \) (and possibly other constraints)
- Tier constraints: for sets (“tiers”) \( T_h \) of assets, \( 1 \leq h \leq H \), we require:
  \( E(\delta \sum_{j \in T_h} w_j) \leq \Gamma_h \) (given)
  or, \( (\sum_i \delta_i \pi_i) \sum_{j \in T_h} w_j \leq \Gamma_h \)
- Cardinality constraint: \( w_j > 0 \) for at most \( N \) indices \( j \)
→ Random return \( j = \bar{\mu}_j(1 - \delta w_j) \) where \( 0 \leq w_j \leq 1 \ \forall \ j \), and \( 0 \leq \delta \leq 1 \) is a random variable.

A discrete distribution:

- We are given fixed values \( 0 = \delta_0 \leq \delta_2 \leq \ldots \leq \delta_K = 1 \)
- example: \( \delta_i = \frac{i}{K} \)
- Adversary chooses \( \pi_i = \text{Prob}(\delta = \delta_i) \), \( 0 \leq i \leq K \)
- The \( \pi_i \) are constrained: we have fixed bounds, \( \pi^l_i \leq \pi_i \leq \pi^u_i \)
  (and possibly other constraints)
- Tier constraints: for sets ("tiers") \( T_h \) of assets, \( 1 \leq h \leq H \), we require:
  \( E(\delta \sum_{j \in T_h} w_j) \leq \Gamma_h \) (given)

  or, \( \left( \sum_i \delta_i \pi_i \right) \sum_{j \in T_h} w_j \leq \Gamma_h \)

- Cardinality constraint: \( w_j > 0 \) for at most \( N \) indices \( j \)
→ Random return \( j \) = \( \bar{\mu}_j (1 - \delta w_j) \) where \( 0 \leq w_j \leq 1 \ \forall \ j \), and \( 0 \leq \delta \leq 1 \) is a random variable.

A discrete distribution:

- We are given fixed values \( 0 = \delta_0 \leq \delta_2 \leq \ldots \leq \delta_K = 1 \)
  example: \( \delta_i = \frac{i}{K} \)
- Adversary chooses \( \pi_i = \text{Prob}(\delta = \delta_i) \), \( 0 \leq i \leq K \)
- The \( \pi_i \) are constrained: we have fixed bounds, \( \pi^l_i \leq \pi_i \leq \pi^u_i \)
  (and possibly other constraints)
- Tier constraints: for sets (“tiers”) \( T_h \) of assets, \( 1 \leq h \leq H \), we require:
  \[ E(\delta \sum_{j \in T_h} w_j) \leq \Gamma_h \quad \text{(given)} \]

or, \( (\sum_i \delta_i \pi_i) \sum_{j \in T_h} w_j \leq \Gamma_h \)

- Cardinality constraint: \( w_j > 0 \) for at most \( N \) indices \( j \)
Adversarial problem

Random return \( j \) = \( \bar{\mu}_j (1 - \delta w_j) \) where \( 0 \leq w_j \leq 1 \) \( \forall j \), and \( 0 \leq \delta \leq 1 \) is a random variable.

A discrete distribution:

- We are given fixed values \( 0 = \delta_0 \leq \delta_2 \leq \ldots \leq \delta_K = 1 \)
  
  example: \( \delta_i = \frac{i}{K} \)

- Adversary chooses \( \pi_i = \text{Prob}(\delta = \delta_i), 0 \leq i \leq K \)

- The \( \pi_i \) are constrained: we have fixed bounds, \( \pi_i^l \leq \pi_i \leq \pi_i^u \)
  (and possibly other constraints)

- Tier constraints: for sets (“tiers”) \( T_h \) of assets, \( 1 \leq h \leq H \), we require:
  \[
  E(\delta \sum_{j \in T_h} w_j) \leq \Gamma_h \quad (\text{given})
  \]
  
  or,
  \[
  (\sum_i \delta_i \pi_i) \sum_{j \in T_h} w_j \leq \Gamma_h
  \]

- Cardinality constraint: \( w_j > 0 \) for at most \( N \) indices \( j \)
The adversarial problem is “easy”

\[ K = \text{no. of points in discrete distribution}, \quad H = \text{no. of tiers} \]

**Theorem**

- Without the cardinality constraint, for each fixed \( K \) and \( H \) the adversarial problem can be solved as a polynomial number of linear programs.

- With the cardinality constraint, for each fixed \( K \) and \( H \) the adversarial problem can be solved as a polynomial number of knapsack problems.
The adversarial problem is “easy”

\[ K = \text{no. of points in discrete distribution}, \quad H = \text{no. of tiers} \]

**Theorem**

- Without the cardinality constraint, for each fixed \( K \) and \( H \) the adversarial problem can be solved as a polynomial number of linear programs.

- With the cardinality constraint, for each fixed \( K \) and \( H \) the adversarial problem can be solved as a polynomial number of knapsack problems.
The adversarial problem is “easy”

\( K = \) no. of points in discrete distribution, \( H = \) no. of tiers

**Theorem**

- Without the cardinality constraint, for each fixed \( K \) and \( H \) the adversarial problem can be solved as a polynomial number of linear programs.

- With the cardinality constraint, for each fixed \( K \) and \( H \) the adversarial problem can be solved as a polynomial number of knapsack problems.
**Adversarial problem as an MIP**

Recall: random return $\mu_j = \bar{\mu}_j (1 - \delta w_j)$

where $\delta = \delta_i$ (given) with probability $\pi_i$ (chosen by adversary),

$$0 \leq \delta_0 \leq \delta_1 \leq \ldots \leq \delta_K = 1$$

and $0 \leq w$

$$\min_{\pi,w,V} \min_{1 \leq i \leq k} V_i$$

Subject to

$$0 \leq w_j \leq 1, \text{ all } j, \pi^l_i \leq \pi_i \leq \pi^u_i, \text{ all } i,$$

$$\sum_i \pi_i = 1,$$

$$V_i = \sum_j \bar{\mu}_j (1 - \delta_i w_j) x_j^*, \text{ if } \pi_i + \pi_{i+1} + \ldots + \pi_K \geq 1 - \theta$$

$$V_i = M \text{ (large), otherwise}$$

$$(\sum_i \delta_i \pi_i) \sum_{j \in T_h} w_j \leq \Gamma_h, \text{ for each tier } h$$
Adversarial problem as an MIP

Recall: random return $j$, $\mu_j = \bar{\mu}_j(1 - \delta w_j)$
where $\delta = \delta_i$ (given) with probability $\pi_i$ (chosen by adversary),
$0 \leq \delta_0 \leq \delta_1 \leq \ldots \leq \delta_K = 1$ and $0 \leq w$

$$\min_{\pi,w,v} \min_{1 \leq i \leq k} V_i$$

Subject to

$$0 \leq w_j \leq 1, \text{ all } j, \pi^l_i \leq \pi_i \leq \pi^u_i, \text{ all } i,$$
$$\sum_i \pi_i = 1,$$

$$V_i = \sum_j \bar{\mu}_j(1 - \delta_i w_j) x_j^*, \text{ if } \pi_i + \pi_{i+1} + \ldots + \pi_K \geq 1 - \theta$$
$$V_i = M \text{ (large)}, \text{ otherwise}$$

$$\sum_i \delta_i \pi_i \sum_{j \in T_h} w_j \leq \Gamma_h, \text{ for each tier } h$$
Adversarial problem as an MIP

Recall: random return $j \quad \mu_j = \bar{\mu}_j(1 - \delta w_j)$
where $\delta = \delta_i$ (given) with probability $\pi_i$ (chosen by adversary),
$0 \leq \delta_0 \leq \delta_1 \leq \ldots \leq \delta_K = 1$ and $0 \leq w$

$$\min_{\pi,w,V} \min_{1 \leq i \leq k} V_i$$

Subject to

- $0 \leq w_j \leq 1$, all $j$, $\pi_i^l \leq \pi_i \leq \pi_i^u$, all $i$,
- $\sum_i \pi_i = 1$,
- $V_i = \sum_j \bar{\mu}_j(1 - \delta_i w_j)x^*_j$, if $\pi_i + \pi_{i+1} + \ldots + \pi_K \geq 1 - \theta$
- $V_i = M$ (large), otherwise

$$(\sum_i \delta_i \pi_i) \sum_{j \in T_h} w_j \leq \Gamma_h$$, for each tier $h$
Adversarial problem as an MIP

Recall: random return

\[ \mu_j = \bar{\mu}_j(1 - \delta w_j) \]

where \( \delta = \delta_i \) (given) with probability \( \pi_i \) (chosen by adversary),
\[ 0 \leq \delta_0 \leq \delta_1 \leq \ldots \leq \delta_K = 1 \] and \( 0 \leq w \)

\[
\min_{\pi, w, V} \min_{1 \leq i \leq k} V_i
\]

Subject to

\[ 0 \leq w_j \leq 1, \text{ all } j, \pi_i^l \leq \pi_i \leq \pi_i^u, \text{ all } i, \]
\[ \sum_i \pi_i = 1, \]
\[ V_i = \sum_j \bar{\mu}_j(1 - \delta_i w_j)x_j^*, \text{ if } \pi_i + \pi_{i+1} + \ldots + \pi_K \geq 1 - \theta \]
\[ V_i = M \text{ (large), otherwise} \]

\[(\sum_i \delta_i \pi_i) \sum_{j \in T_h} w_j \leq \Gamma_h, \text{ for each tier } h\]
Approximation

\[
(\sum_i \delta_i \pi_i) \sum_{j \in T_h} w_j \leq \Gamma_h, \quad \text{for each tier } h \quad (\star)
\]

Let \( N > 0 \) be an integer. For \( 1 \leq k \leq N \), write

\[
\frac{k}{N} \sum_{j \in T_h} w_j \leq \Gamma_h + M \left( 1 - z_{hk} \right), \quad \text{where}
\]

\[
z_{hk} = 1 \quad \text{if} \quad \frac{k-1}{N} < \sum_i \delta_i \pi_i \leq \frac{k}{N}
\]

\[
z_{hk} = 0 \quad \text{otherwise}
\]

\[
\sum_k z_{hk} = 1
\]

and \( M \) is large

**Lemma.** Under reasonable conditions, replacing \((\star)\) with this system changes the value of the problem by at most a factor of \((1 + \frac{1}{N})\)
Approximation

\[(\sum_i \delta_i \pi_i) \sum_{j \in T_h} w_j \leq \Gamma_h, \quad \text{for each tier } h \quad (\star)\]

Let \( N > 0 \) be an integer. For \( 1 \leq k \leq N \), write

\[\frac{k}{N} \sum_{j \in T_h} w_j \leq \Gamma_h + M \left(1 - z_{hk}\right), \quad \text{where}\]

\[z_{hk} = 1 \text{ if } \frac{k-1}{N} < \sum_i \delta_i \pi_i \leq \frac{k}{N}\]

\[z_{hk} = 0 \text{ otherwise}\]

\[\sum_k z_{hk} = 1\]

and \( M \) is large

**Lemma.** Under reasonable conditions, replacing \((\star)\) with this system changes the value of the problem by at most a factor of \((1 + \frac{1}{N})\)
Approximation

\[(\sum_i \delta_i \pi_i) \sum_{j \in T_h} w_j \leq \Gamma_h, \quad \text{for each tier } h \quad (*)\]

Let \( N > 0 \) be an integer. For \( 1 \leq k \leq N \), write

\[\frac{k}{N} \sum_{j \in T_h} w_j \leq \Gamma_h + M (1 - z_{hk}), \quad \text{where}\]

\[z_{hk} = 1 \text{ if } \frac{k-1}{N} < \sum_i \delta_i \pi_i \leq \frac{k}{N}\]

\[z_{hk} = 0 \text{ otherwise}\]

\[\sum_k z_{hk} = 1\]

and \( M \) is large

**Lemma.** Under reasonable conditions, replacing \((*)\) with this system changes the value of the problem by at most a factor of \( 1 + \frac{1}{N} \)
Implementor’s problem

Find a near-optimal solution with minimum value-at-risk

Nominal problem:

\[ v^* = \min_x \lambda x^T Q x - \mu^T x \]

Subject to:

\[ Ax \geq b \]
Implementor’s problem

Find a near-optimal solution with minimum value-at-risk

Nominal problem:

\[ v^* = \min_x \lambda x^T Q x - \mu^T x \]

Subject to:

\[ Ax \geq b \]
Implementor’s problem

Find a near-optimal solution with minimum value-at-risk

Given asset weights $\mathbf{x}$, we have:

- value-at-risk $\geq \rho$, if the adversary can produce a return vector $\mathbf{\mu}$ with

$$\text{Prob}(\mathbf{\nu} - \mathbf{\mu}^T \mathbf{x} \geq \rho) \geq \theta$$

where $\mathbf{\nu}$ is a fixed reference value.
Implementor’s problem

Find a near-optimal solution with minimum value-at-risk

Implementor’s problem at iteration \( r \):

\[
\begin{align*}
\text{min} & \quad V \\
\text{Subject to:} & \\
\lambda x^T Q x - \mu^T x & \leq (1 + \epsilon) v^* \\
Ax & \geq b \\
V & \geq \nu - \sum_j \bar{\mu}_j \left( 1 - \delta_i(t) w_j^{(t)} \right) x_j, \quad t = 1, 2, \ldots, r - 1
\end{align*}
\]

Here, \( \delta_i(t) \) and \( w^{(t)} \) are the adversary’s output at iteration \( t < r \).
Implementor’s problem

Find a near-optimal solution with minimum value-at-risk

Implementor’s problem at iteration $r$:

$$\text{min } V$$

Subject to:

$$\lambda x^T Q x - \mu^T x \leq (1 + \epsilon) \nu^*$$

$$Ax \geq b$$

$$V \geq \nu - \sum_j \bar{\mu}_j \left( 1 - \delta_{i(t)} w_j^{(t)} \right) x_j, \quad t = 1, 2, \ldots, r - 1$$

Here, $\delta_{i(t)}$ and $w^{(t)}$ are the adversary’s output at iteration $t < r$. 
First set of experiments
1338 assets, 41 factors, 81 rows

- problem: find a “near optimal” solution with minimum value-at-risk, for a given threshold probability $\theta$
- experiment: investigate different values of $\theta$
- “near optimal”: want solutions that are at most 1% more expensive than optimal
- random variable $\delta$:
First set of experiments

1338 assets, 41 factors, 81 rows

- problem: find a “near optimal” solution with minimum value-at-risk, for a given threshold probability $\theta$
- experiment: investigate different values of $\theta$
- “near optimal”: want solutions that are at most 1% more expensive than optimal
- random variable $\delta$: 

![CDF graph]

Daniel Bienstock (Columbia University)
Experiments in Robust Optimization
10-26-06
### 1338 assets, 41 factors, 81 rows, $\leq 1\%$ suboptimality

<table>
<thead>
<tr>
<th>$\theta$</th>
<th>time (sec.)</th>
<th>iters.</th>
<th>VAR</th>
<th>VAR as %</th>
</tr>
</thead>
<tbody>
<tr>
<td>.89</td>
<td>1.18</td>
<td>2</td>
<td>3.43131</td>
<td>71.50</td>
</tr>
<tr>
<td>.90</td>
<td>1.42</td>
<td>3</td>
<td>3.74498</td>
<td>78.04</td>
</tr>
<tr>
<td>.91</td>
<td>1.42</td>
<td>3</td>
<td>3.74498</td>
<td>78.04</td>
</tr>
<tr>
<td>.92</td>
<td>3.47</td>
<td>11</td>
<td>4.05669</td>
<td>84.53</td>
</tr>
<tr>
<td>.93</td>
<td>6.35</td>
<td>29</td>
<td>4.05721</td>
<td>84.54</td>
</tr>
<tr>
<td>.94</td>
<td>6.37</td>
<td>20</td>
<td>4.05721</td>
<td>84.54</td>
</tr>
<tr>
<td>.95</td>
<td>26.59</td>
<td>51</td>
<td>4.35481</td>
<td>90.74</td>
</tr>
<tr>
<td>.96</td>
<td>26.25</td>
<td>51</td>
<td>4.35481</td>
<td>90.74</td>
</tr>
<tr>
<td>.97</td>
<td>26.20</td>
<td>51</td>
<td>4.35481</td>
<td>90.74</td>
</tr>
<tr>
<td>.98</td>
<td>33.07</td>
<td>58</td>
<td>4.63938</td>
<td>96.67</td>
</tr>
<tr>
<td>.99</td>
<td>33.11</td>
<td>58</td>
<td>4.63938</td>
<td>96.67</td>
</tr>
</tbody>
</table>
Second set of experiments
Fix $\theta = 0.90$ but vary suboptimality criterion
Typical convergence behavior
- Heavy tail, proportional error (100 points):

- Heavy tail, constant error (100 points):
- Heavy tail, proportional error (100 points):

- Heavy tail, constant error (100 points):
<table>
<thead>
<tr>
<th>$\theta$</th>
<th>Proportional</th>
<th></th>
<th>Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Its</td>
<td>Time</td>
<td>Its</td>
</tr>
<tr>
<td>.85</td>
<td>4.79</td>
<td>8</td>
<td>11.84</td>
</tr>
<tr>
<td>.86</td>
<td>2.32</td>
<td>3</td>
<td>8.27</td>
</tr>
<tr>
<td>.87</td>
<td>6.40</td>
<td>10</td>
<td>9.55</td>
</tr>
<tr>
<td>.88</td>
<td>4.34</td>
<td>4</td>
<td>18.10</td>
</tr>
<tr>
<td>.89</td>
<td>8.00</td>
<td>14</td>
<td>5.85</td>
</tr>
<tr>
<td>.90</td>
<td>2.58</td>
<td>4</td>
<td>13.54</td>
</tr>
<tr>
<td>.91</td>
<td>4.79</td>
<td>9</td>
<td>16.31</td>
</tr>
<tr>
<td>.92</td>
<td>7.99</td>
<td>15</td>
<td>13.13</td>
</tr>
<tr>
<td>.93</td>
<td>13.43</td>
<td>27</td>
<td>22.47</td>
</tr>
<tr>
<td>.94</td>
<td>10.04</td>
<td>15</td>
<td>21.99</td>
</tr>
<tr>
<td>.95</td>
<td>9.59</td>
<td>16</td>
<td>11.90</td>
</tr>
<tr>
<td>.96</td>
<td>6.63</td>
<td>17</td>
<td>29.89</td>
</tr>
<tr>
<td>.97</td>
<td>48.43</td>
<td>110</td>
<td>16.45</td>
</tr>
<tr>
<td>.98</td>
<td>20.25</td>
<td>53</td>
<td>20.25</td>
</tr>
<tr>
<td>.99</td>
<td>22.02</td>
<td>52</td>
<td>21.89</td>
</tr>
</tbody>
</table>
A difficult case

- 2464 columns, 152 factors, 3 tiers
- time = 6191 seconds
- 258 iterations
- implementor time = 6123 seconds, adversarial time = 20 seconds
A difficult case

- 2464 columns, 152 factors, 3 tiers
- time = 6191 seconds
- 258 iterations
- implementor time = 6123 seconds, adversarial time = 20 seconds
A difficult case

- 2464 columns, 152 factors, 3 tiers
- time = 6191 seconds
- 258 iterations
- implementor time = 6123 seconds, adversarial time = 20 seconds
A difficult case

- 2464 columns, 152 factors, 3 tiers
- time = 6191 seconds
- 258 iterations

- implementor time = 6123 seconds,
  adversarial time = 20 seconds
A difficult case

- 2464 columns, 152 factors, 3 tiers
- time = 6191 seconds
- 258 iterations
- implementor time = 6123 seconds, adversarial time = 20 seconds
Implementor runtime

![Graph showing QCP time over iterations]
Implementor’s problem at iteration $r$

$$\begin{align*}
\min & \quad V \\
\text{Subject to:} & \\
\lambda x^T Q x - \mu^T x & \leq (1 + \epsilon) v^* \\
Ax & \geq b \\
V & \geq \nu - \sum_j \bar{\mu}_j \left(1 - \delta_{i(t)} w^{(t)}_j\right) x_j, \quad t = 1, 2, \ldots, r - 1
\end{align*}$$

Here, $\delta_{i(t)}$ and $w^{(t)}$ are the adversary’s output at iteration $t < r$. 
Implementor’s problem at iteration r

Approximate version

\[
\begin{align*}
\min & \quad V \\
\text{Subject to:} & \quad 2\lambda x_{(k)}^T Q x - \lambda x_{(k)}^T Q x_{(k)} - \mu^T x \leq (1 + \epsilon) V^*, \quad \forall k < r \\
& \quad Ax \geq b \\
& \quad V \geq \nu - \sum_j \bar{\mu}_j \left(1 - \delta_{i(k)} w_j^{(k)}\right) x_j, \quad \forall k < r
\end{align*}
\]

Here, \(\delta_{i(k)}\) and \(w^{(k)}\) are the adversary’s output at iteration \(k < r\), and \(x_{(k)}\) is the implementor’s output at iteration \(k\).
Does it work?

- Before: 258 iterations, 6191 seconds
- Linearized: 1776 iterations, 3969 seconds
Does it work?

- Before: 258 iterations, **6191** seconds
- Linearized: 1776 iterations, **3969** seconds
Averaging

- $X(k)$ is the implementor’s output at iteration $k$.

- Define $y(1) = x(1)$

- For $k > 1$, $y(k) = \lambda x(k) + (1 - \lambda) y(k-1)$, $0 \leq \lambda \leq 1$

- Input $y(k)$ to the adversary

- Old ideas, also Nesterov, Nemirovsky (2003)
Averaging

- \( x_k \) is the implementor’s output at iteration \( k \).
- Define \( y(1) = x(1) \)
- For \( k > 1 \), \( y(k) = \lambda x(k) + (1 - \lambda) y(k-1) \), \( 0 \leq \lambda \leq 1 \)
- Input \( y(k) \) to the adversary
- Old ideas, also Nesterov, Nemirovsky (2003)
Averaging

- $x(k)$ is the implementor’s output at iteration $k$.

- Define $y(1) = x(1)$

- For $k > 1$, $y(k) = \lambda x(k) + (1 - \lambda) y(k-1)$, $0 \leq \lambda \leq 1$

- Input $y(k)$ to the adversary

- Old ideas, also Nesterov, Nemirovsky (2003)
Averaging

- \( x(k) \) is the implementor’s output at iteration \( k \).
- Define \( y(1) = x(1) \)
- For \( k > 1 \), \( y(k) = \lambda x(k) + (1 - \lambda) y(k - 1), \quad 0 \leq \lambda \leq 1 \)
- Input \( y(k) \) to the adversary

Old ideas, also Nesterov, Nemirovsky (2003)
Averaging

- \( x(k) \) is the implementor’s output at iteration \( k \).
- Define \( y(1) = x(1) \)
- For \( k > 1 \), \( y(k) = \lambda x(k) + (1 - \lambda) y(k-1) \), \( 0 \leq \lambda \leq 1 \)
- Input \( y(k) \) to the adversary
- Old ideas, also Nesterov, Nemirovsky (2003)
Does it work?

- Default: 258 iterations, **6191** seconds
- Linearized: 1776 iterations, **3969** seconds
- Averaging plus Linearized: 860 iterations, **530** seconds
Does it work?

- Default: 258 iterations, 6191 seconds
- Linearized: 1776 iterations, 3969 seconds
- Averaging plus Linearized: 860 iterations, 530 seconds
Does it work?

- Default: 258 iterations, 6191 seconds
- Linearized: 1776 iterations, 3969 seconds
- Averaging plus Linearized: 860 iterations, 530 seconds
Other robust models

- Min-max expected loss with orthogonal missing factor

Random return = $\bar{\mu} \cdot (1 - \delta w)$ where $-1 \leq w_j \leq 1 \ \forall j$, and $0 \leq \delta \leq 1$ is a random variable.

normalization constraints, e.g. $\sum_j w_j = 0$

- errors in covariance matrix $Q$

robust problem: $\rightarrow \min_x \max_{Q \in \mathcal{Q}} \lambda x^T Q x - \mu^T x$
Other robust models

- Min-max expected loss with orthogonal missing factor

  Random return = $\bar{\mu} \bullet (1 - \delta w)$ where $-1 \leq w_j \leq 1 \ \forall \ j$, and $0 \leq \delta \leq 1$ is a random variable.

  normalization constraints, e.g. $\sum_j w_j = 0$

- errors in covariance matrix $Q$

  robust problem: $\rightarrow \min_x \ \max_{Q \in Q} \ \lambda x^T Q x - \mu^T x$
Other robust models

- Min-max expected loss with orthogonal missing factor
  
  Random return = $\bar{\mu} \cdot (1 - \delta w)$ where $-1 \leq w_j \leq 1$ $\forall j$, and $0 \leq \delta \leq 1$ is a random variable.

  normalization constraints, e.g. $\sum_j w_j = 0$

- errors in covariance matrix $Q$

  robust problem: $\rightarrow \min_x \max_{Q \in \mathcal{Q}} \lambda x^T Q x - \mu^T x$
On-going work: a provably good version of Benders’ algorithm