Robust optimization with the Xpress Optimizer

CO@Work, Berlin, 6 October 2015

Pietro Belotti
Xpress Optimization, FICO
Birmingham UK

About the Xpress suite

- Started by Dash Optimization (Bob **D**aniel, Robert **Ash**ford) in 1988
- Acquired by FICO (Fair Isaac) in 2008

Comes with

- A modeling language, Mosel
- A solver for most optimization problems
- API to create and solve your own optimization problems
- A modeler running on-site and on the FICO Analytic Cloud

Free academic licenses with the Academic Partnership Program[1]

[1] `http://subscribe.fico.com/Academic-Partner-Program`

FICO

About the Xpress suite (cont'd)

Latest version: 7.9, released earlier this year.

Solves:

- LP, QP, QCQP
- MILP, MIQP, MIQCQP, MISOCP
- NLP, MINLP (Nonconvex (MI)NLPs solved to local optimality)
- CP

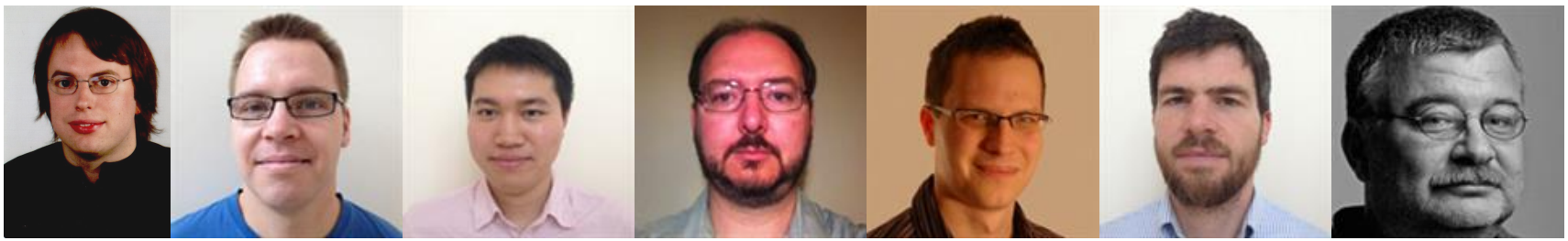
About the Xpress suite (cont'd)

Latest version: 7.9, released earlier this year.

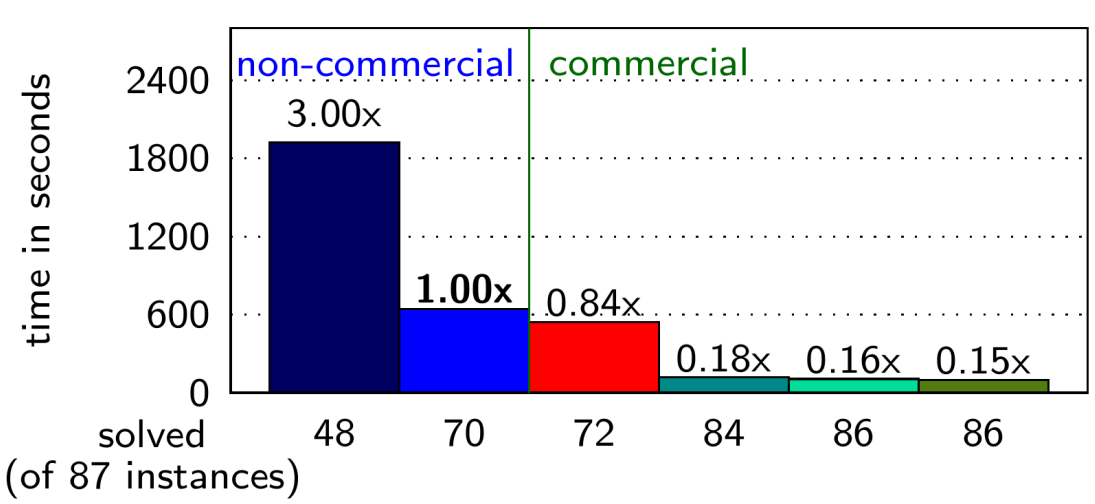
Solves:

- LP, QP, QCQP
- MILP, MIQP, MIQCQP, MISOCP
- NLP, MINLP (Nonconvex (MI)NLPs solved to local optimality)
- CP

Development group based in Berlin, Birmingham, Budapest



MILP comparison (courtesy of the SCIP web page)



- CBC 2.9.4
- **SCIP 3.2.0 – SoPlex 2.2.0**
- SCIP 3.2.0 – CPLEX 12.6.2
- Gurobi 6.0.0
- CPLEX 12.6.2
- Xpress 7.9.0

data: Hans Mittelmann
graphics: ZIB

FICO

Today: Robust Optimization

We'll put the solver to work for solving an interesting meta-class of problems: optimization under uncertainty

Examples, exercises, and slides can be downloaded at

`zib.de/berthold/rob-opt.zip`

Outline

- Robust Optimization at a glance
- The opponent's viewpoint
- A ridiculously fast intro to Mosel
- Exercise session I

Outline

- Robust Optimization at a glance
- The opponent's viewpoint
- A ridiculously fast intro to Mosel
- Exercise session I
- Theory and pitfalls of RO
- Types of uncertainty sets
- Exercise session II

The name of the game

- Robust optimization is a paradigm for modeling optimization problems under uncertainty

i.e. One or more of the problem's **parameters** are unknown

- We only have limited information on their value
- We assume these parameters vary in a **well-defined** uncertainty set

Robust optimization

- Real-life problems have uncertainty in their data
- Some might be missing or affected by inaccuracy, measurement errors, etc.
- However, sometimes there's a limit on the uncertainty

$\Rightarrow$ We have an uncertainty set

Robust optimization

- Real-life problems have uncertainty in their data
- Some might be missing or affected by inaccuracy, measurement errors, etc.
- However, sometimes there's a limit on the uncertainty

$\Rightarrow$ We have an uncertainty set

- There are a few ties with stochastic programming, but RO is an entirely different approach.

Example

$$
\begin{array}{ll}
\min & 2x_1 + 3x_2 \\
\text{s.t.} & u x_1 + 2x_2 \geq 5 \\
& 2x_1 + x_2 \geq v \\
& x_1, x_2 \geq 0
\end{array}
$$

Uncertain parameters: u and v. The only information we have on them is $u \in [1, 2]$, $v \in [5, 6]$.

Main assumptions

- Once the uncertainty set is defined, the uncertain parameters can take any value in it

$\Rightarrow$ RO does **not** use any probabilistic information on the uncertainty set

Main assumptions

- Once the uncertainty set is defined, the uncertain parameters can take any value in it

$\Rightarrow$ RO does **not** use any probabilistic information on the uncertainty set

The problem is easier

Main assumptions

- Once the uncertainty set is defined, the uncertain parameters can take any value in it

$\Rightarrow$ RO does **not** use any probabilistic information on the uncertainty set

☺ The problem is easier

☹ Throwing away lots of useful info

RO’s challenge

Find a solution that is feasible for **all** possible realizations of the uncertain parameters.

Example

$$\begin{array}{ll} \min & 2x_1 + 3x_2 \\ \text{s.t.} & u_1 x_1 + u_2 x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{array}$$

Uncertainty set: $U = \{(2,1),(1,2)\}$.

- $(x_1, x_2) = (3,0)$ is infeasible because it violates the robust constraint $x_1 + 2x_2 \geq 6$

Example

$$\begin{array}{ll} \min & 2x_1 + 3x_2 \\ \text{s.t.} & u_1 x_1 + u_2 x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{array}$$

Uncertainty set: $U = \{(2,1),(1,2)\}$.

- $(x_1, x_2) = (3,0)$ is infeasible because it violates the robust constraint $x_1 + 2x_2 \geq 6$
- $(x_1, x_2) = (0,3)$ is infeasible because it violates the robust constraint $2x_1 + x_2 \geq 6$

Example

$$\begin{array}{ll} \min & 2x_1 + 3x_2 \\ \text{s.t.} & u_1 x_1 + u_2 x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{array}$$

Uncertainty set: $U = \{(2,1),(1,2)\}$.

- $(x_1, x_2) = (3,0)$ is infeasible because it violates the robust constraint $x_1 + 2x_2 \geq 6$
- $(x_1, x_2) = (0,3)$ is infeasible because it violates the robust constraint $2x_1 + x_2 \geq 6$
- $(x_1, x_2) = (2,2)$ is feasible and optimal

Example

$$
\begin{array}{ll}
\min & 2x_1 + 3x_2 \\
\text{s.t.} & x_1 + 2x_2 \geq 6 \\
& 2x_1 + x_2 \geq 6 \\
& x_1, x_2 \geq 0
\end{array}
$$

The trick was simply to impose the constraint for all elements of the uncertainty set.

Not-yet-robust optimization

Forget uncertainty for a second. Your optimization problem has

- parameters: known before solving the problem.
- variables: unknown, but we find them by solving a problem

Variables express quantities we make **decisions** on.

- We have control over them, we decide their value
- We want to set them so as to minimize some objective

Robust optimization

In RO, an optimization problem has

- parameters: known before solving the problem.
- variables: unknown, decided by us
- uncertains: unknown, decided by somebody else

Robust optimization

In RO, an optimization problem has

- parameters: known before solving the problem.
- variables: unknown, decided by us
- uncertains: unknown, decided by somebody else

Unlike variables, we have no control over uncertainties.

Robust optimization

In RO, an optimization problem has

- parameters: known before solving the problem.
- variables: unknown, decided by us
- uncertains: unknown, decided by somebody else

Unlike variables, we have no control over uncertainties.

- In fact, they may be known **after** we made our decision.
- Remember: we have to make our decisions so that they are feasible for any uncertain realization

The opponent's viewpoint

Uncertains in RO are decisions made by an opponent.

The opponent's viewpoint

Uncertains in RO are decisions made by an opponent.

- Think about your favorite supervillain (Voldemort, the Joker, Skynet, Gargamel, Thanos, . . .)

The opponent's viewpoint

Uncertains in RO are decisions made by an opponent.

- Think about your favorite supervillain (Voldemort, the Joker, Skynet, Gargamel, Thanos, . . .)
- **After** we're done optimizing, it's their turn:
  - They see our solution
  - They know the uncertainty set
  - They pick the uncertains that will do us **maximum** harm

  i.e. Make any of our constraints violated

$\Rightarrow$ They are also solving an optimization problem: the uncertains are their variables

- This is akin to a *leader-follower* game

The opponent is quite real

- Nature
- Competitors
- Market
- Customers
- Suppliers
- All of the above, aka Murphy’s law

Example

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & u_1 x_1 + u_2 x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Uncertainty set: $U = \{(2,1),(1,2)\}$.

Example

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & u_1 x_1 + u_2 x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Uncertainty set: $U = \{(2,1),(1,2)\}$.

If we picked $(x_1, x_2) = (1.1, 2.5)$ as a solution, the opponent would search $(u_1, u_2) \in U$ such that

$$1.1u_1 + 2.5u_2 < 6$$

So $(u_1, u_2) = (2,1)$ would be the opponent's solution, and ours would be proven infeasible.

Example

$$\begin{array}{ll} \min & 2x_1 + 3x_2 \\ \text{s.t.} & u_1 x_1 + u_2 x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{array}$$

Uncertainty set:

$$U = \{(u_1, u_2) \in \mathbb{R}^2_+ : u_1 + u_2 \geq 3, u_1 + 2u_2 \geq 4\}.$$

Suppose we pick again $(x_1, x_2) = (1.1, 2.5)$ as a solution.

Example (cont'd)

The opponent would then solve

$$\begin{array}{ll} \min & 1.1u_1 + 2.5u_2 \\ \text{s.t.} & u_1 + u_2 \geq 3 \\ & u_1 + 2u_2 \geq 4 \end{array}$$

A simple LP that gives $(u_1, u_2) = (4, 0)$, with an objective of $4.4 < 6$.

- The opponent managed to break our constraint,

i.e. Our solution is not robust

Solving RO problems

In order to solve a RO problem, we must

- Assume that the opponent will try to invalidate solution

Solving RO problems

In order to solve a RO problem, we must

- Assume that the opponent will try to invalidate solution
- Anticipate the opponent's move

Solving RO problems

In order to solve a RO problem, we must

- Assume that the opponent will try to invalidate solution
- Anticipate the opponent's move

$\Rightarrow$ Create a robust counterpart of our optimization problem

The RC is just another optimization problem (if one exists). It just embeds the opponent's optimization problem.

Example

$$
\begin{array}{ll}
\min & 2x_1 + 3x_2 \\
\text{s.t.} & u x_1 + 2x_2 \geq 5 \\
& 2x_1 + x_2 \geq v \\
& x_1, x_2 \geq 0
\end{array}
$$

Uncertainty set: $u \in [1, 2]$, $v \in [5, 6]$.

Example

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & ux_1 + 2x_2 \geq 5 \\ & 2x_1 + x_2 \geq v \\ & x_1, x_2 \geq 0 \end{aligned}$$

Uncertainty set: $u \in [1, 2]$, $v \in [5, 6]$.

The RC is trivial: the constraints will be hardest if $u = 1$ and $v = 6$, so the RC is

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 \geq 5 \\ & 2x_1 + x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Questions?

If you haven't any, here are two:

- What if there are uncertains in the objective?

Questions?

If you haven't any, here are two:

- What if there are uncertains in the objective?
- What if two or more constraints are affected by uncertains?

We'll answer these later.

A quick intro to Xpress-Mosel

Mosel is a modeling language for optimization problems.

- It can do a lot of other things
- But we'll deal with optimization only

Typical

```
model "hello world"
declarations
  y, x: mpvar              ! This is a comment
end-declarations
y + 2*x <= 10
y + x <= 5
maximize(2*x + 3*y)
writeln("x: ", getsol(x), "; Obj: ", getobjval)
end-model
```

Declaration section

This is where all variables, parameters, sets, and uncertains are declared

```
model "my model"
uses "mmxprs", "mmrobust"
declarations
  x, y: mpvar                    ! variables
  a: array (1..4) of real        ! vector of parameters
  b: real                        ! single parameters
  R: range                       ! set i..j, def'd later
  u: uncertain                   ! uncertain parameter
end-declarations
[...]
```

Constraints

Not too far from how they are formulated in other languages...

```
model "my model"
declarations
  P: range
  a: array(P) of real
  b: real
  x: array(P) of mpvar
  [...]
end-declarations
c1 := sum (i in range) a(i) * x(i) <= b
[...]
```

The "`c1 :=`" is not necessary

Objective

```
maximize (sum (i in P) a(i)* x(i))
writeln ("Obj: ", getobjval)
```

Miscellaneous

Loops:

```
forall (i in P) do
  writeln ("x_", i, ": ", getsol (x(i)))
end-do

forall (i in P: i <= 2) do
  writeln ("x_", i, ": ", getsol (x(i)))
end-do
```

Long comments:

```
(!
  Write whatever you want here.
  I'll just ignore it
!)
```

Variable types and bounds

To be done outside of the declaration section

```
forall (i in P) do
  x(i) is_binary
end-do
y is_integer
z >= 4
z <= z_upper ! z_upper is a parameter
```

Reading data from files

```
initializations from 'datafile.txt'
   a n
end-initializations
```

The file `datafile.txt` must have a specific format

Modules

We have to tell Mosel the modules we want to use. We'll use two:

```
model "my model"
uses "mmxprs", "mmrobust"
[...]
end-model
```

Example: knapsack

$$\begin{array}{ll} \max & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{s.t.} & \boldsymbol{a}^\top \boldsymbol{x} \leq b \\ & \boldsymbol{x} \in \{0,1\}^n \end{array}$$

Writing the model is easy. We'll have to set up a data file with n, $\boldsymbol{a}$ and b.

Example: knapsack (cont'd)

```
model "knapsack"
uses "mmxprs"
declarations
  P: range
  c: array(P) of real
  a: array(P) of real
  b: real
  x: array(P) of mpvar
end-declarations
initializations from 'knapsack.dat'
  c a b
end-initializations
forall (i in P) x(i) is_binary
sum (i in P) a(i) * x(i) <= b
maximize (sum (i in P) c(i) * x(i))
end-model
```


FICO

The `knapsack.dat` file

```
n: 6
c: [(1) 34 (2) 37 (3) 41 (4) 49 (5) 52 (6) 55]
a: [(1) 56 (2) 38 (3) 32 (4) 24 (5) 22 (6) 20]
b: 100
```

Example: **robust** knapsack

$$\begin{array}{ll} \max & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{s.t.} & (\boldsymbol{a}+\boldsymbol{u})^\top \boldsymbol{x} \leq b \\ & \boldsymbol{x} \in \{0,1\}^n \end{array}$$

Uncertainty set:

$$U = \{\boldsymbol{u} \in \mathbb{R}^n_+ : \boldsymbol{d}^\top \boldsymbol{u} \leq f\},$$

where $\boldsymbol{d} \in \mathbb{Q}^n_+$ and $f > 0$ is a scalar.

Example: **robust** knapsack (cont'd)

```
model "knapsack"
declarations
  [...]
  x: array(P) of mpvar
  d: array(P) of real          ! data for unc. set
  f: real
  u: array(P) of uncertain     ! unc parameters
end-declarations
initializations from 'knapsack_robust.dat'
  c a b d f
end-initializations
forall (i in P) x(i) is_binary
sum (i in P) (a(i) + u(i)) * x(i) <= b
sum (i in P) d(i) * u(i) <= f
forall (i in P) u(i) >= 0
[...]
```

The knapsack_robust.dat file

```
n: 6
c: [(1) 34 (2) 37 (3) 41 (4) 49 (5) 52 (6) 55]
a: [(1) 56 (2) 38 (3) 32 (4) 24 (5) 22 (6) 20]
d: [(1) 11 (2) 12 (3)  8 (4) 13 (5) 11 (6) 13]
f: 1
b: 100
```

Unanswered questions in RO

- What if there are uncertains in the objective?
- What if two or more constraints are affected by uncertains?

Remember: we are looking for a solution that is feasible **regardless** of the uncertain parameters.

What if there are uncertains in the **objective**?

Suppose the uncertain parameters are only in the objective.

- Feasibility of any solution is independent of the uncertains
- In principle, optimality would depend on the uncertains

$\Rightarrow$ We would have to deal with robust optimality, an entirely different issue

- We'll use a trick to reduce this to a feasibility problem.

Example: knapsack

$$\begin{array}{ll} \max & (\boldsymbol{c}+\boldsymbol{u})^\top \boldsymbol{x} \\ \text{s.t.} & \boldsymbol{a}^\top \boldsymbol{x} \leq b \\ & \boldsymbol{x} \in \{0,1\}^n \end{array}$$

Uncertainty set: $U = \{\boldsymbol{u} \in \mathbb{R}^n_+ : \boldsymbol{d}^\top \boldsymbol{u} \leq f\}$.

Reformulate

$$
\begin{array}{ll}
\max & z \\
\text{s.t.} & z - (\boldsymbol{c} + \boldsymbol{u})^\top \boldsymbol{x} \leq 0 \\
& \boldsymbol{a}^\top \boldsymbol{x} \leq b \\
& \boldsymbol{x} \in \{0,1\}^n
\end{array}
$$

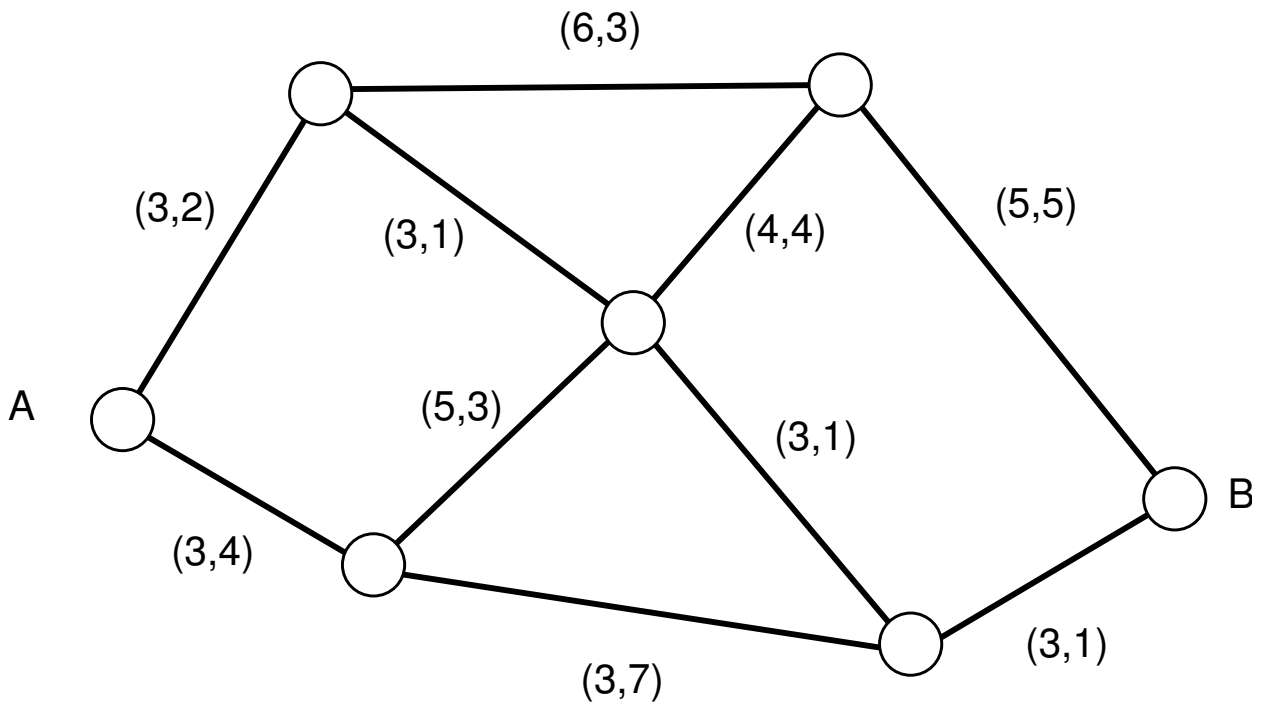
Now we have a robust problem in the classical sense.

Example: robust shortest path

Find the shortest route from A to B on the city's road network.

- It takes c_e minutes to drive on road e
- Unless there's construction work, and then it's $c_e + d_e$
- We don't know where the construction work is
- **But** we know it is on at most k roads

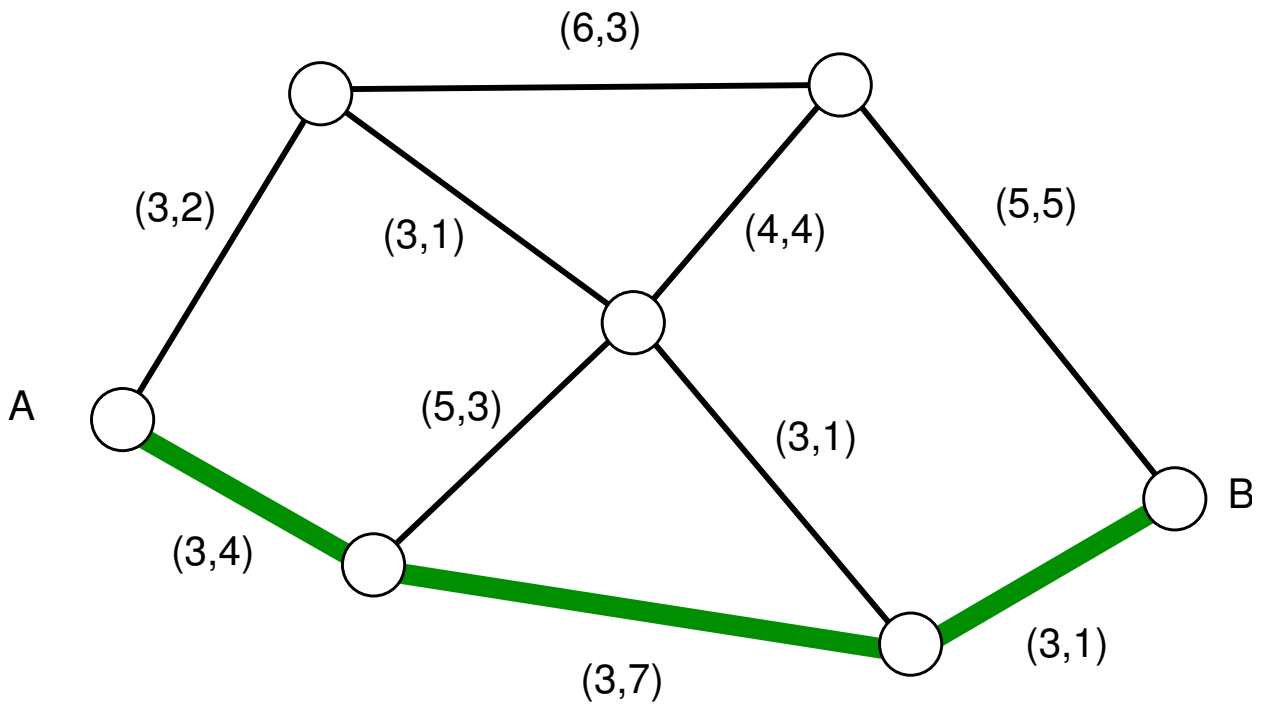
Each link e has (c_e, d_e). $k = 2$ construction zones.



How long to get from A to B?

FICO

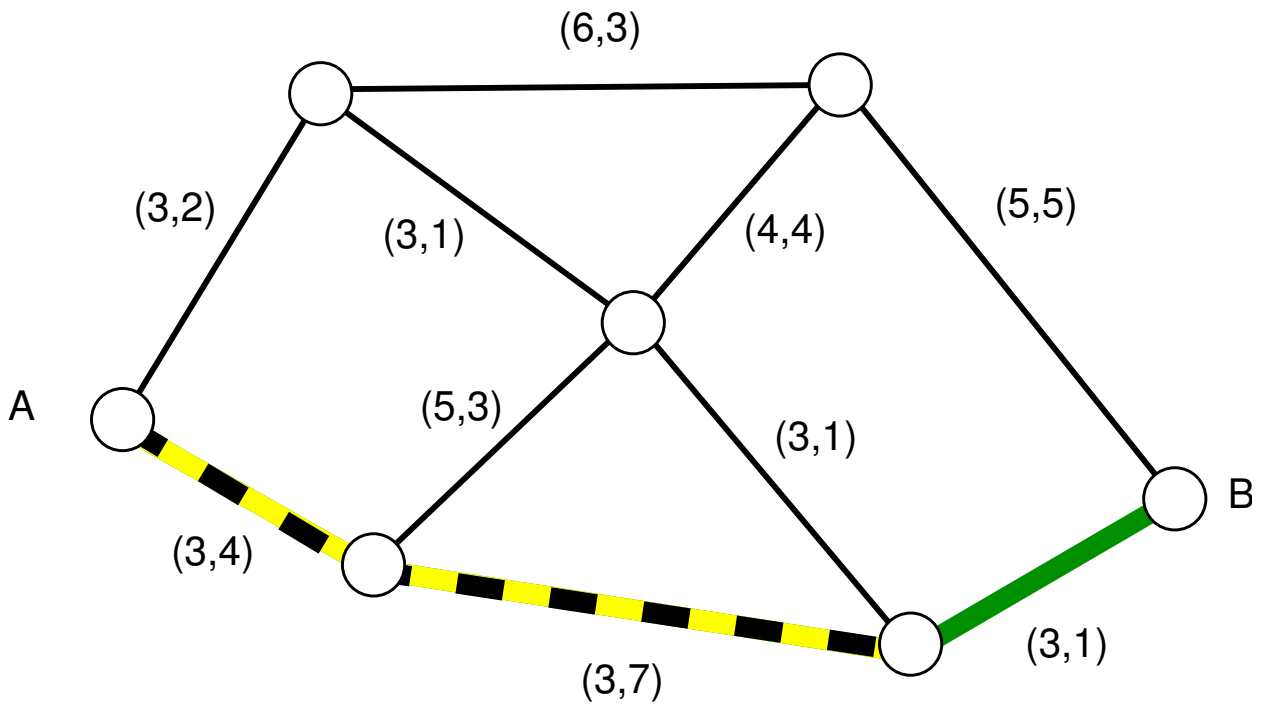
Each link e has (c_e, d_e). $k = 2$ construction zones.



How long to get from A to B? In theory, 9 minutes

FICO

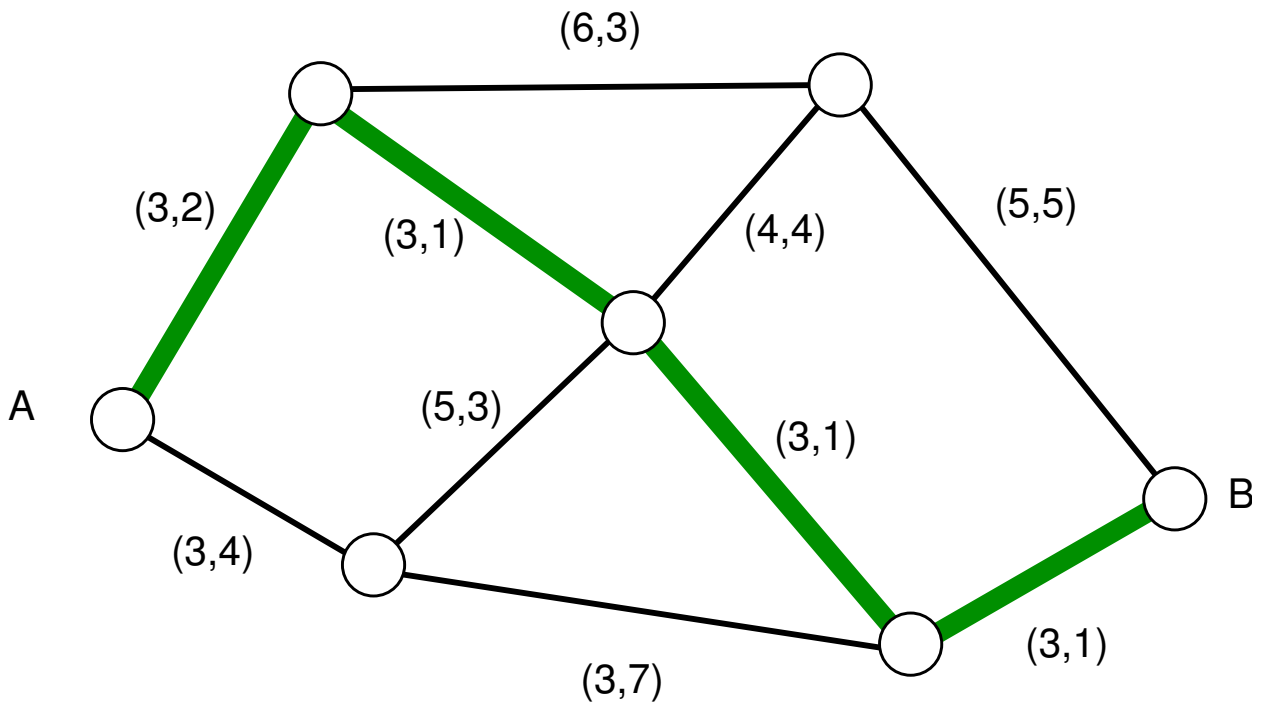
Each link e has (c_e, d_e). $k = 2$ construction zones.



How long to get from A to B? In practice, 20 min.

FICO

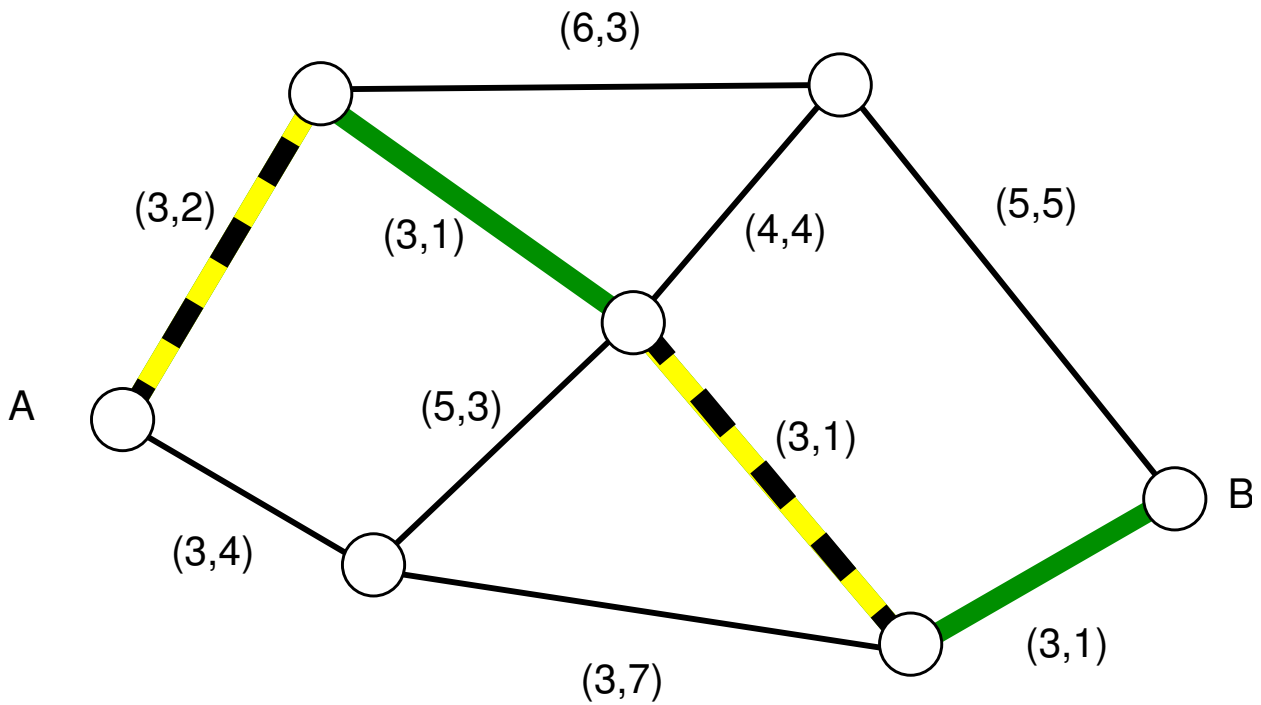
Each link e has (c_e, d_e). $k = 2$ construction zones.



How long to get from A to B? Nominal 12 min. (worse than 9)

FICO

Each link e has (c_e, d_e). $k = 2$ construction zones.



How long to get from A to B? Robust 15 min. (better than 20)

FICO

What if 2+ constraints are affected by uncertains?

Remember: the opponent simply wants to give us a hard time.

His/her method:

1. Observe our solution
2. Find point of uncertainty set so that our solution violates at least one of our constraint
3. So the opponent can focus on one constraint at a time

$\Rightarrow$ Quite a conservative approach: equivalent to one opponent per constraint!

Finally, some theory

So far, we've seen:

- What RO does to a problem
- How to solve some ROs
- How to implement them using Mosel

But how do we solve a general RO?

Robust optimization: a general-enough case

$$
\begin{array}{ll}
\min & \boldsymbol{c}^\top \boldsymbol{x} \\
\text{s.t.} & A\boldsymbol{x} = \boldsymbol{b} \\
& \sum_{i=1}^{n} \alpha_i u_i x_i \leq \beta \\
& \boldsymbol{x} \geq \boldsymbol{0}.
\end{array}
$$

Uncertainty set: $U = \{\boldsymbol{u} \in \mathbb{R}^n_+ : P\boldsymbol{u} \leq \boldsymbol{q}\}$.
U is a polyhedron defined by a system of linear inequalities.

What would an opponent do?

The opponent would look at the single uncertain constraint:

$$\sum_{i=1}^{n} \alpha_i u_i x_i^\star \leq \beta,$$

and think: "$x_i^\star$'s are known. How do I make this violated?"

What would an opponent do?

The opponent would look at the single uncertain constraint:

$$\sum_{i=1}^{n} \alpha_i u_i x_i^\star \leq \beta,$$

and think: "$x_i^\star$'s are known. How do I make this violated?"

The opponent has a PhD in Optimization, so the answer is:

"I'll maximize $\sum_{i=1}^{n} \alpha_i x_i^\star u_i$! Mwahahahahahahaha!"

What would an opponent do?

The opponent would look at the single uncertain constraint:

$$\sum_{i=1}^{n} \alpha_i u_i x_i^\star \leq \beta,$$

and think: "$x_i^\star$'s are known. How do I make this violated?"

The opponent has a PhD in Optimization, so the answer is:

"I'll maximize $\sum_{i=1}^{n} \alpha_i x_i^\star u_i$! Mwahahahahahahaha!"

The reason: maximizing the left-hand side gives the highest chance to make the constraint violated.

If the solution found is such that $\sum_{i=1}^{n} \alpha_i x_i^\star u_i > \beta$, the opponent wins.

What would an opponent do?

If the constraint were of the opposite sign,

$$\sum_{i=1}^{n} \alpha_i u_i x_i \geq \beta,$$

the plan (equally evil) would be
"I'll m**in**imize $\sum_{i=1}^{n} \alpha_i u_i x_i$! Mwahahahahahahaha!"

What would an opponent do?

If the constraint were of the opposite sign,

$$\sum_{i=1}^{n} \alpha_i u_i x_i \geq \beta,$$

the plan (equally evil) would be
"I'll m**in**imize $\sum_{i=1}^{n} \alpha_i u_i x_i$! Mwahahahahahahaha!"
Equality constraints are simply split into $\leq$ and $\geq$ constraints

The robust counterpart

$$
\begin{array}{ll}
\min & \boldsymbol{c}^\top \boldsymbol{x} \\
\text{s.t.} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\
& \max_{\boldsymbol{u} \in U} \sum_{i=1}^{n} \alpha_i u_i x_i \leq \beta \\
& \boldsymbol{x} \geq \boldsymbol{0}.
\end{array}
$$

The opponent's optimization problem

$$\begin{array}{ll} \max & \sum_{i=1}^{n} \alpha_i x_i u_i \\ \text{s.t.} & \boldsymbol{P}\boldsymbol{u} \leq \boldsymbol{q} \\ & \boldsymbol{u} \geq \boldsymbol{0}. \end{array}$$

Here, x_i's are given. The variables (i.e. the decision that are controlled by the opponent) are the u_i's

Duality, our savior

If the opponent has this problem:

$$\begin{array}{ll} \max & \sum_{i=1}^{n} \alpha_i x_i u_i \\ \text{s.t.} & \boldsymbol{P}\boldsymbol{u} \leq \boldsymbol{q} \\ & \boldsymbol{u} \geq \boldsymbol{0}. \end{array}$$

Duality, our savior

If the opponent has this problem:

$$\begin{array}{ll} \max & \sum_{i=1}^{n} \alpha_i x_i u_i \\ \text{s.t.} & \boldsymbol{P}\boldsymbol{u} \leq \boldsymbol{q} \\ & \boldsymbol{u} \geq \boldsymbol{0}. \end{array}$$

The dual is (for dual variables $\boldsymbol{y}$)

$$\begin{array}{ll} \min & \boldsymbol{q}^\top \boldsymbol{y} \\ \text{s.t.} & P_{i,\cdot}^\top \boldsymbol{y} \geq \alpha_i x_i \quad \forall i = 1, 2, \ldots, n \\ & \boldsymbol{y} \geq \boldsymbol{0}. \end{array}$$

Duality, our savior

If the opponent has this problem:

$$\begin{array}{ll} \max & \sum_{i=1}^{n} \alpha_i x_i u_i \\ \text{s.t.} & \boldsymbol{P}\boldsymbol{u} \leq \boldsymbol{q} \\ & \boldsymbol{u} \geq \boldsymbol{0}. \end{array}$$

The dual is (for dual variables $\boldsymbol{y}$)

$$\begin{array}{lll} \min & \boldsymbol{q}^\top \boldsymbol{y} & \\ \text{s.t.} & P_{i,\cdot}^\top \boldsymbol{y} \geq \alpha_i x_i & \forall i = 1, 2, \ldots, n \\ & \boldsymbol{y} \geq \boldsymbol{0}. & \end{array}$$

However, strong duality dictates:

If an optimal solution $(\boldsymbol{u}, \boldsymbol{y})$ exists, then $\sum_{i=1}^{n} \alpha_i x_i u_i = \boldsymbol{q}^\top \boldsymbol{y}$

The robust counterpart[2]

From a difficult (nonlinear) problem

$$
\begin{array}{ll}
\min & \boldsymbol{c}^\top \boldsymbol{x} \\
\text{s.t.} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\
 & \max_{\boldsymbol{u} \in U} \sum_{i=1}^{n} \alpha_i u_i x_i \leq \beta \\
 & \boldsymbol{x} \geq \boldsymbol{0}.
\end{array}
$$

[2] Soyster, Ben Lev, Toof, "Conservative linear programming with mixed multiple objectives." *Omega 5.2* (1977): 193-205.

The robust counterpart[2]

From a difficult (nonlinear) problem

$$\begin{array}{ll} \min & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \max_{\boldsymbol{u}\in U} \sum_{i=1}^{n} \alpha_i u_i x_i \leq \beta \\ & \boldsymbol{x} \geq \boldsymbol{0}. \end{array}$$

we get a much nicer problem (an LP!):

$$\begin{array}{ll} \min & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{q}^\top \boldsymbol{y} \leq \beta \\ & P_{i,\cdot}^\top \boldsymbol{y} \geq \alpha_i x_i \quad \forall i = 1, 2, \ldots, n \\ & \boldsymbol{x}, \boldsymbol{y} \geq \boldsymbol{0}. \end{array}$$

[2] Soyster, Ben Lev, Toof, "Conservative linear programming with mixed multiple objectives." *Omega 5.2* (1977): 193-205.

Non-polyhedral uncertainty sets

In the general case, we have seen that a polyhedral uncertainty set can be dealt with easily.

- If the original problem is a LP, the RC is an LP as well
- If the original problem is a MILP, the RC is a MILP

So polyhedral uncertainty doesn't add to the complexity of the problem

Ellipsoidal uncertainty

Suppose the uncertainty set has been estimated as an ellipsoid defined by mean/covariance data:

$$U = \{\boldsymbol{u} \in \mathbb{R}^n : (\boldsymbol{u} - \bar{\boldsymbol{u}})^\top Q(\boldsymbol{u} - \bar{\boldsymbol{u}}) \leq \epsilon\}$$

Hence $\bar{\boldsymbol{u}}$ is the mean value of $\boldsymbol{u}$ and Q is the covariance matrix of $\boldsymbol{u}$.

Ellipsoidal uncertainty $\rightarrow$ a conic RC

It is easy to show (we won't) that the resulting robust counterpart becomes

- a SOCP if the initial problem was a LP
- a MISOCP if the initial problem was a MILP

Ellipsoidal uncertainty in Mosel

```
model "quadknapsack"
declarations
  P: range
  c: array(P) of real
  a: array(P) of real
  b: real
  x: array(P) of mpvar
  u: array(P) of uncertain  ! unc parameters
end-declarations
initializations from 'knapsack_robust_eps.dat'
  c a b eps
end-initializations
forall (i in P) x(i) is_binary
sum (i in P) (a(i) + u(i)) * x(i) <= b
sum (i in P) u(i)^2 <= eps
[...]
```


FICO

Scenario uncertainty

Suppose there is a database of K historical values of the vector $\boldsymbol{u}$ of uncertain parameters: $U = \{\boldsymbol{u}^1, \boldsymbol{u}^2, \cdots, \boldsymbol{u}^K\}$.

- Problem: be robust against all previous occurrences of the vector $\boldsymbol{u}$
- Solution: specify $\boldsymbol{u}$ as a (discrete) set of occurrences

So the robust counterpart is

$$\begin{array}{ll} \min & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{s.t.} & A\boldsymbol{x} = \boldsymbol{b} \\ & \sum_{i=1}^{n} \alpha_i u_i^k x_i \leq \beta \quad \forall k = 1, 2, \ldots, K \\ & \boldsymbol{x} \geq \boldsymbol{0}. \end{array}$$

Example: knapsack with historical values

```
declarations:
  [...]
  hist_data: array (1..4, set of uncertain) of real
  [...]
end-declarations

hist_data (1,u(1)) := 12
hist_data (1,u(2)) := 14
hist_data (1,u(3)) := 20
[...]
hist_data (4,u(3)) := 10

scenario (hist_data)
```

References

- Ben-Tal, A., El Ghaoui, L., Nemirovski, A. (2009). Robust optimization. Princeton University Press.
- Bertsimas, D., Sim, M. (2004). The price of robustness. *Operations Research*, 52(1), 35-53.

Thank You

Pietro Belotti
pietrobelotti@fico.com

Real-world cases: Liquid gas production[3]

Production planning at a liquid oxygen/nitrogen plant:

- Given customer demands at the beginning of the month
- Plan production for each day of the month
- Store extra daily production in inventory (big tanks)
- Maximum production P, tank capacity C

[3] Latifoglu, C., Belotti, P., Snyder, L.V. (2013). Models for production planning under power interruptions. *Naval Research Logistics* 60(5):413-431.

Production planning under energy uncertainty

- Production very energy-intensive $\Rightarrow$ expensive electric bill
- Interruptible Load Contract (ILC): power company can suspend supply in periods of high demand (summer)
- At most k interruptions each month (8 hours each)
- **Cheaper** (per kWh) than with uninterrupted contract

Production planning under energy uncertainty

- Production very energy-intensive $\Rightarrow$ expensive electric bill
- Interruptible Load Contract (ILC): power company can suspend supply in periods of high demand (summer)
- At most k interruptions each month (8 hours each)
- **Cheaper** (per kWh) than with uninterrupted contract

The power supplier won't tell us when the interruptions will be.

- Treat interruptions as uncertains

$\Rightarrow$ Plan production so that even with the worst-case k interruptions we satisfy customer demand

Original model

```
declarations
  produce:    array (PERIODS, GASES) of mpvar
  inventory:  array (PERIODS, GASES) of mpvar

end-declarations

forall(t in PERIODS, g in GASES) do
  inventory(0,g) + sum(tp in PERIODS | tp <= t)
    (produce (tp,g) - DEMAND(tp,g)) >= 0
  inventory (t,g) <= INV_CAP  (g)
  produce   (t,g) <= PROD_CAP (g)
end-do


minimize(sum (t in PERIODS, g in GASES)
 (PROD_COST * produce(t,g) + INV_COST * inventory(t,g)))
```

Robust model

```
declarations
  produce:    array (PERIODS, GASES) of mpvar
  inventory:  array (PERIODS, GASES) of mpvar
  interrupt:  array (PERIODS) of uncertain
end-declarations

forall(t in PERIODS, g in GASES) do
  inventory(0,g) + sum(tp in PERIODS | tp <= t)
    ((1-interrupt(tp))*produce(tp,g)-DEMAND(tp,g))>=0
  inventory (t,g) <= INV_CAP  (g)
  produce   (t,g) <= PROD_CAP (g)
end-do

sum(t in PERIODS) interrupt (t) <= MAX_NINTERR

minimize(sum (t in PERIODS, g in GASES)
 (PROD_COST * produce(t,g) + INV_COST * inventory(t,g)))
```

Modeling advantage

Modeling this is much easier with the `mmrobust` module.

The alternative: add the dual constraints to the model.

- Error prone
- Only suited to a user with good knowledge of optimization, duality, and robust optimization

Original problem

$$
\begin{array}{lll}
\min & \sum_{t \in T} c_g^{\text{prod}} \text{prod}_{tg} + c_g^{\text{inv}} \text{inv}_{tg} & \\
\text{s.t.} & \text{inv}_{0,g} + \sum_{\tau \le t} (\text{prod}_{\tau,g} - \text{dem}_{\tau,g}) \ge 0 & \forall t \in T \\
& 0 \le \text{prod}_{t,g} \le P_g & \forall t \in T, \tau \in T \\
& 0 \le \text{inv}_{t,g} \le C_g & \forall t \in T, \tau \in T
\end{array}
$$

Robust problem

$$
\begin{array}{lll}
\min & \sum_{t\in T} c_g^{\text{prod}}\text{prod}_{tg} + c_g^{\text{inv}}\text{inv}_{tg} & \\
\text{s.t.} & \text{inv}_{0,g} + \sum_{\tau\le t}(\text{prod}_{\tau,g} - \text{dem}_{\tau,g}) & \\
& \quad -\sum_{\tau\in T}\mu_\tau^t + k\sigma^t \ge 0 & \forall t \in T \\
& \mu_\tau^t + \sigma^t \ge \text{prod}_{\tau,g} & \forall t \in T, \tau \le t \\
& \mu_\tau^t + \sigma^t \ge 0 & \forall t \in T, \tau > t \\
& \mu_\tau^t \ge 0, \sigma^t \ge 0 & \forall t \in T, \tau \in T \\
& 0 \le \text{prod}_{t,g} \le P_g & \forall t \in T, \tau \in T \\
& 0 \le \text{inv}_{t,g} \le C_g & \forall t \in T, \tau \in T
\end{array}
$$

Exercise session II (25-30 minutes)

Complete exercise4.mos.

Exercise session I (25-30 minutes)

Complete exercise1.mos, exercise2.mos, and exercise3.mos.