TD #1: Basic modelling

Large-scale Mathematical Programming

Leo Liberti, CNRS LIX Ecole Polytechnique
liberti@lix.polytechnique.fr

INF580

Software

Modelling

Implementation

Section 1

Software

Structured and flat formulations

- Mathematical Programs (MP) describing *problems* involve sets and parameters
  e.g. $\min\{c^\top x \mid Ax \geq b\}$
- For each set of values assigned to the parameters, MP describes a different *instance*
  e.g. $\min\{x_1 + 2x_2 \mid x_1 + x_2 >= 1\}$

Structured and flat formulations

- Mathematical Programs (MP) describing *problems* involve sets and parameters
  e.g. $\min\{c^\top x \mid Ax \geq b\}$
- For each set of values assigned to the parameters, MP describes a different *instance*
  e.g. $\min\{x_1 + 2x_2 \mid x_1 + x_2 >= 1\}$
- Humans reason in terms of problems (*structured formulations*)
- Solvers provide solutions for instances (*flat formulations*)
- Need a translation from problems to instances: **modelling languages**
  (e.g. AMPL, Python+amplpy/cvxpy/pyomo, Matlab+YALMIP, Julia+JuMP, …)

AMPL vs. Python

- AMPL
  - wonderful syntax close to mathematics
  - interfaces with lots of solvers, including MINLP (but little SDP)
  - imperative sub-language: poor (no function calls, no libraries)
  - good for rapid prototyping or "just use the solver"
- Python
  - mixture of declarative (pyomo) and imperative (Python)
  - interfaces with many solvers, including SDP (but little MINLP)
  - excellent imperative sub-language (Python itself)
  - good for "doing further stuff with the solution"

Installing AMPL

- Linux bundle:

```
cd ~
tar zxvf ~/Downloads/ampl_lin64-bundle.tgz
mv ampl_linux-intel64 ampl
cd ; echo "export PATH=$PATH:~/ampl" >> ~/.bash_profile
source ~/.bash_profile
```

- Windows bundle
  1. make directory `C:\ampl`
  2. copy `ampl-win64_bundle.zip` inside `C:\ampl` and unzip it
  3. insert `C:\ampl` in the `PATH` environment variable

     System Properties dialog/*Advanced* tab/*Environment Variables* button/*Path* field/*Edit* button/add `C:\ampl` to the string, separated by semicolons
- Windows installer: run `ampl-win64_installer.exe`

  choose `C:\ampl` as installation directory
- MacOS installer: run `ampl-macos.pkg`, same as Windows

Testing AMPL

1. open a command prompt / terminal window
2. Save the following to `test.run`

```
set M := 1..50;
set N := 1..10;
param c{N} default Uniform01();
param A{M,N} default Uniform(0,1);
param b{M} default Uniform(1,2);
var x{N} >= 0;
minimize f: sum{j in N} c[j]*x[j];
subject to C{i in M}:
  sum{j in N} A[i,j]*x[j] >= b[i];
option solver cplex;
solve;
display x,f,solve_result;
```

3. type `ampl test.run`
4. optimal objective function value is `f = 1.34199`

Section 2

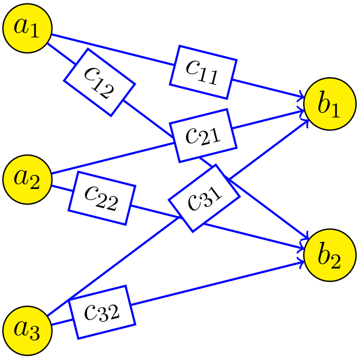
Modelling

The transportation problem

Given a set P of production facilities with production capacities a_i for $i \in P$, a set Q of customer sites with demands b_j for $j \in Q$, and knowing that the unit transportation cost from facility $i \in P$ to customer $j \in Q$ is c_{ij}, find the optimal transportation plan

The art of modelling!

- *Use drawings — they help to think*



First fundamental question

1. What decisions does the problem require?

First fundamental question

1. What decisions does the problem require?

   1. what's given?
   2. costs — unit, refers to quantities
   3. capacities and demand based on quantities
   4. ⇒ *let's decide quantities*

First fundamental question

1. What decisions does the problem require?

    1. what's given?
    2. costs — unit, refers to quantities
    3. capacities and demand based on quantities
    4. ⇒ *let's decide quantities*

- *As you go on with the model, you might find your initial choices were poor — you might have to go back and change them*

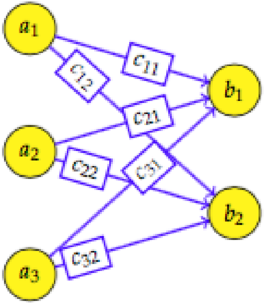
Second fundamental question

1. How can the decision be encoded?

Second fundamental question

1. How can the decision be encoded?

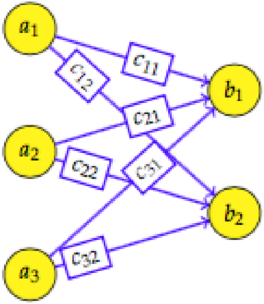
let's go back to the drawing



Second fundamental question

1. How can the decision be encoded?

let's go back to the drawing



- How about:
  z_i = qty. produced at i
  y_j = qty. demanded at j

Let's try this choice

1. *Sets and indices*
   a. $i \in P \subset \mathbb{N}$
   b. $j \in Q \subset \mathbb{N}$
2. *Parameters*
   a. $\forall i \in P \quad a_i \in \mathbb{R}_+$
   b. $\forall j \in Q \quad b_j \in \mathbb{R}_+$
   c. $\forall i \in P, j \in Q \quad c_{ij} \in \mathbb{R}_+$
3. *Decision variables*
   a. $\forall i \in P \quad z_i \in [0, a_i]$
   b. $\forall j \in Q \quad y_j \in [b_j, \infty]$
4. *Constraints*
   a. All that is produced must be delivered: $\sum_{i \in P} z_i = \sum_{j \in Q} y_j$

      necessary condition, but is it sufficient?
5. *Objective function*: ???

   no way of knowing what fraction of the production out of i went to j, so how do we consider transportation costs?

Bad choice, let's go back

- Failure to express "*fraction of i going to j*" must inspire us
  Let's try x_{ij} = qty. transported from i to j

1. *Sets*: as before
2. *Parameters*: as before
3. *Decision variables*
   a. $\forall i \in P, j \in Q \quad x_{ij} \in \mathbb{R}_+$
4. *Objective function*

$$\min \sum_{i \in P} \sum_{j \in Q} c_{ij} x_{ij}$$

5. *Constraints*
   a. No facility can produce more than the maximum:
   $\forall i \in P \quad \sum_{j \in Q} x_{ij} \leq a_i$
   b. No customer must receive less than its demand:
   $\forall j \in Q \quad \sum_{i \in P} x_{ij} \geq b_j$

Much better!

Section 3

Implementation

The AMPL encoding

- Three files:
  - `file.mod`: the *model file*
    containing the description of the structured formulation
  - `file.dat`: the *data file*
    containing the description of the instance
  - `file.run`: the *run file*
    the "imperative part": choice of solver, run, analyze solution…
  - Run "`ampl file.run`" and get results on file or screen

The transportation problem in AMPL: `.mod`

```
# transportation.mod
param Pmax integer;
param Qmax integer;
set P := 1..Pmax;
set Q := 1..Qmax;
param a{P};
param b{Q};
param c{P,Q};
var x{P,Q} >= 0;
minimize cost: sum{i in P, j in Q} c[i,j]*x[i,j];
subject to production{i in P}:
  sum{j in Q} x[i,j] <= a[i];
subject to demand{j in Q}:
  sum{i in P} x[i,j] >= b[j];
```

The transportation problem in AMPL: `.dat`

```
# transportation.dat
param Pmax := 2;
param Qmax := 1;
param a :=
 1  2.0
 2  2.0
;
param b :=
 1  1.0
;
param c :=
 1 1  1.0
 2 1  2.0
;
```

The transportation problem in AMPL: `.run`

```
# transportation.run
model transportation.mod;
data transportation.dat;
option solver cplex;
solve;
display x, cost;
```