ISE 426
Optimization models and applications

Lecture 6 — September 16, 2015

- AMPL
- Graph problems + AMPL

Useful author's lecture notes on AMPL can be found here
http://www.4er.org/CourseNotes/

AMPL

- a modeling language for optimization problems

⇒ an interface between problems and solvers

- easy, intuitive syntax
- it's interpreted[1]

⇒ that means errors are spotted as soon as they are executed.

- can be used from a command line interface or
- by editing command files and submitting them

[1] As opposed to compiled.

AMPL: Variables

The command `var` specifies a variable of the problem, its bounds, its type, and (possibly) an initial value.

```
var x1 >= 0 <= 4;
var numtrucks >= 3 integer;
var buyObj1 binary;
```

AMPL: constraint

Constraints are preceded by a name and a ":"

```
myconstr1: x1 + 3*x2 + x3 <= 4;
c2: numtrucks >= ntr_AZ + ntr_NY + ntr_PA;
at_most_1: buyObj1 + buyObj2 + buyObj3 <= 1;
```

AMPL: Objective function

Preceded by either `minimize` or `maximize`, name of the objective, and ":"

```
minimize xpense:  10*numtrucks + 3*numcars;
minimize myfun:  x1*x2 - 2*x3;
maximize distance:  x1 + 2*cos(x3);
```

AMPL: to remember

- the **semicolon**: all commands must end with one[2];
- comments: `# this won't be interpreted`
- to solve a problem, choose a solver:
  - CPLEX: for linear and integer programming
  - MINOS: for nonlinear programming

  with the command `option solver cplex;`
- guess what the `reset;` command does?
- remember the tin can problem?

[2] AMPL gives seemingly unrelated errors when it doesn't find one.

The tin can problem

```
var radius >= 0.0001 default 1;
var height >= 0.0001 default 1;
minimize tin_foil:
   2*3.14*radius^2 + 2*3.14*radius*height;
vol_fixed: 3.14*radius^2 * height = 20;
```

Parameters

Parameters can be defined with `param`. They are input to the problem.

```
param pi = 3.14159265358979323846;
param Vol = 20; # volume of each tin can
param price_per_gallon; # Can set it later!
```

Tin can problem

```
param pi = 3.14159265358979323846;
param Vol = 20; # volume of each tin can
var radius >= 0.0001 default 1;
var height >= 0.0001 default 1;
minimize tin_foil:
   2*pi*radius^2 + 2*pi*radius*height;
vol_fixed: pi*radius^2 * height = Vol;
```

Sets

- remember the Knapsack problem?

```
var x1 binary;
var x2 binary;
var x3 binary;
var x4 binary;
var x5 binary;
var x6 binary;
var x7 binary;
var x8 binary;
var x9 binary;
param w1 = 3;
param w2 = 4;
```

- Flea markets in Rome have more than 9 objects...

Sets

```
set S = 1 2 3 4 5 6 7 8 9;
var x {S} binary; # a vector of variables
param w {S}; # a vector of parameters
param p {S};

# even nicer...
set S = 1..9;
var x {S} binary;
param w {S};
param p {S};

# not happy yet?
param n = 9;
set S = 1..n;
var x {S} binary;
param w {S};
param p {S};
```

Sets and indices

Sets can be referred to with *indices*.
Useful to specify an element of a vector parameter/variable.
Example:

```
param lb {S};
param ub {S};
var x {i in S} >= lb[i] <= ub[i];

set T = 1..100;
param firstPar {T};
param secondPar {i in T} = firstPar[i] / 2 + 3;
```

Operations with sets

In Linear and Integer Programming, we'll see very often the notation $\sum_{i=1}^{n} a_i x_i$. How can that be expressed in AMPL?

```
param n;
set S = 1..n;
param a {S};
var x {S};
minimize linFun:  sum {i in S} a [i] * x[i];
```

Model and data

- In AMPL, model and data can be kept separated
- Useful when you have several **instances** of the same **problem** (e.g. one flea market every day)
- Data section starts with command `data;`
- In data sections, we assign values to parameters;

Example: knapsack

```
param n;
set S = 1..n;

var x {S} binary;

param C;
param w {S};
param p {S};

minimize tot_weight: sum {i in S} w[i] * x[i];
pay_ticket: sum {i in S} p[i] * x[i] >= C;
```

Example: knapsack (cont'd)

```
data;

param n := 9;
param C := 70;

param w {S} :=
1 3     2 2
3 2     4 4
5 5     6 4
7 3     8 1
9 4;

param p {S} :=
1 30    2 24
3 11    4 35
5 29    6 8
7 31    8 18
9 12;
```

Example: Transportation problem

- A large manufacturing company produces liquid nytrogen in five plants spread out in East Pennsylvania
- Each plant has a monthly production capacity

Plant	i	1	2	3	4	5
Capacity	p_i	120	95	150	120	140

- It has seven retailers in the same area
- Each retailer has a monthly demand to be satisfied

Retailer	j	1	2	3	4	5	6	7
Demand	d_j	55	72	80	110	85	30	78

- transportation between any plant i and any retailer j has a cost of c_{ij} dollars per volume unit of nytrogen
- c_{ij} is **constant** and depends on the distance between i and j

$\Rightarrow$ find how much nitrogen to be transported from each plant to each retailer

- …while minimizing the total transportation cost

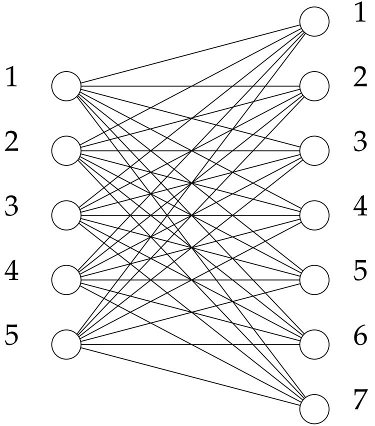
Transportation model

Variables: qty from plant i to retailer j: x_{ij} (non-negative)

Constraints:

1. capacity: $\sum_{j=1}^{7} x_{ij} \leq p_i \quad \forall i$
2. demand: $\sum_{i=1}^{5} x_{ij} \geq d_j \quad \forall j$

Objective function: total transportation cost, $\sum_{i=1}^{5} \sum_{j=1}^{7} c_{ij} x_{ij}$



Variables with multiple indices in AMPL

Variable and parameter vectors can be defined with sets:

```
set S;
param c {S};
var x {S} binary;
var y {i in S} <= c[i] >= c[i]/2;
```

Variables and parameters can be defined on multiple index sets:

```
set Cities;
set Months = 1..12;

param maxBicycles {Cities};

var nBikes {i in Cities} integer <= maxBicycles [i];
var y {Cities, Months} >= 0 <= 4;
var z {i in Cities, j in Cities: i != j} integer;
```

Defining “classes” of constraints in AMPL

Index sets are also useful with constraints!
Constraints can also be defined over a set of indices:

```
set S;
con1 {i in S}: a[i]*x[i] + b[i]*y >= d;

set Cities;
set Months;

param maxYearlyPollution {Cities};
param maxMonthlyPollution {Cities, Months};

var pollution {i in Cities, m in Months}
  >= 0 <= maxMonthlyPollution [i,m];

con_YR_Pollution {i in Cities}:
  sum {m in Months} pollution [i,m]
    <= maxYearlyPollution [i];
```

Back to our transportation problem

```
param np;
param nr;

set P = 1..np;
set R = 1..nr;

param maxCap {P};
param demand {R};

param cost {P,R};

var x {P,R} >= 0;

minimize tcost:
  sum {i in P, j in R} cost[i,j] * x[i,j];

capCon {i in P}: sum {j in R} x [i,j] <= maxCap [i];
demCon {j in R}: sum {i in P} x [i,j] >= demand [j];
```

Problem data (in a separate `.dat` file)

```
param np = 5;
param nr = 7;

param maxCap :=
1 34      2 22
3 41      4 19       5 30;

param demand :=
1 12      2 21
3 19      4 15
5 25      6  9       7 12;

param cost:
   1 2 3 4 5 6 7 :=
1  3 7 5 2 8 9 1
2  5 8 3 6 1 5 4
3  5 4 9 4 5 6 9
4  4 5 7 8 2 4 6
5  5 6 7 2 5 6 6;
```

Example: Production planning

A small firm produces plastic for the car industry.

- At the beginning of the year, it knows exactly the demand d_i of plastic for every month i.
- It also has a maximum production capacity of P

  and an inventory capacity of C.
- The inventory is empty on 01/01

  and has to be empty again on 12/31
- production has a monthly cost c_i

What do we produce at each month to minimize total production cost while satisfying demand?

Production planning model

$$
\begin{array}{ll}
\min & \sum_{i=1}^{12} c_i x_i \\
 & x_i + y_{i-1} = d_i + y_i \quad \forall i = 1, 2 \ldots, 12 \\
 & 0 \le x_i \le P \qquad\qquad\quad \forall i = 1, 2 \ldots, 12 \\
 & 0 \le y_i \le C \qquad\qquad\quad \forall i = 1, 2 \ldots, 11 \\
 & y_0 = y_{12} = 0
\end{array}
$$

Production planning model

```
set Months     = 1..12;
set MonthsPlus = 0..12;

param cost {Months};
param ProdCap;
param InvCap
param demand {Months};

var production {Months}     >= 0 <= ProdCap;
var inventory  {MonthsPlus} >= 0 <= InvCap;

minimize prodCost:
  sum {i in Months} cost [i] * production [i];

conservation {i in Months}:
  production [i] + inventory [i-1] =
  demand     [i] + inventory [i];

Jan1Inv:  inventory  [0] = 0;
Dec31Inv: inventory [12] = 0;
```

Problem data

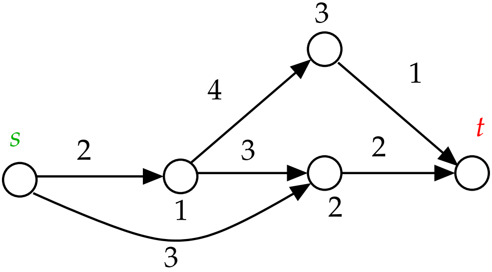
```
param ProdCap = 120;
param InvCap = 70;

param cost :=
1 14       2 19       3 15       4 11
5 10       6  7       7  4       8  5
9  7      10 10      11 11      12 13;

param demand :=
1 110      2  70      3  85      4 90
5 140      6  90      7  40      8 80
9 100     10 105     11 140     12 80;
```

Problem 1: oil pipeline[4]

An oil pipeline pumps oil from an oil well s to an oil refinery t.



Each pipe has its own monthly capacity (in mega-barrels[3]).
Assuming for now an infinite supply of oil at s,

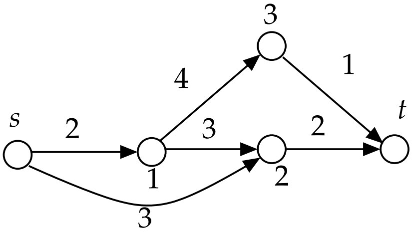
$\Rightarrow$ maximize the amount of oil arriving at t each month

- while not exceeding pipe capacities

[3] One mega-barrel = 10^6 barrels

[4] Winston&Venkataramanan, page 420.

Max-Flow



- Variables: oil flowing on each arc:
  $x_{s1}, x_{s2}, x_{12}, x_{13}, x_{2t}, x_{3t}$
- Constraints: oil is conserved at intermediate nodes
  i.e. what enters node 1 exits node 1: $x_{s1} = x_{12} + x_{13}$
  what enters node 2 exits node 2: $x_{s2} + x_{12} = x_{2t}$
  what enters node 3 exits node 3: $x_{13} = x_{3t}$
- Constraints: there is a maximum capacity on each pipe,
  $x_{s1} \leq 2$, $x_{s2} \leq 3$, $x_{12} \leq 3$, $x_{13} \leq 4$, $x_{2t} \leq 2$, $x_{3t} \leq 1$
- Objective function: The total oil at node t: $x_{2t} + x_{3t}$
- this should be the same oil that left s: $x_{s1} + x_{s2}$

Max-Flow: the model

$$\max \quad x_{2t} + x_{3t}$$

$$\begin{aligned} x_{s1} &= x_{12} + x_{13} \\ x_{s2} + x_{12} &= x_{2t} \\ x_{13} &= x_{3t} \end{aligned}$$

$$\begin{aligned} 0 \leq x_{s1} &\leq 2 \\ 0 \leq x_{s2} &\leq 3 \\ 0 \leq x_{12} &\leq 3 \\ 0 \leq x_{13} &\leq 4 \\ 0 \leq x_{2t} &\leq 2 \\ 0 \leq x_{3t} &\leq 1 \end{aligned}$$

Could re-write objective function as $\quad \max \ x_{s1} + x_{s2}$

Max-Flow in AMPL

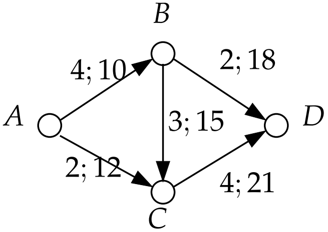
```
var x_s1 >= 0 <= 2;
var x_s2 >= 0 <= 3;
var x_12 >= 0 <= 3;
var x_13 >= 0 <= 4;
var x_2t >= 0 <= 2;
var x_3t >= 0 <= 1;

maximize outflow: x_2t+x_3t; # same as x_s1+x_s2

cons_n1: x_s1          = x_12 + x_13;
cons_n2: x_s2 + x_12 = x_2t;
cons_n3: x_13          = x_3t;
```

See also material on http://www.4er.org/CourseNotes/ webpage.

Min-Cost-Flow



- Variables: oil flowing on each arc:
  $x_{AB}, x_{AC}, x_{BC}, x_{BD}, x_{CD}$
- Constraints: oil does not evaporate at interm. nodes
  i.e. what enters B exits B: $x_{AB} = x_{BC} + x_{BD}$
  what enters C exits C: $x_{AC} + x_{BC} = x_{CD}$
- Constraints: there is a maximum capacity on each pipe,
  $x_{AB} \leq 4, x_{AC} \leq 2, x_{BC} \leq 3, x_{BD} \leq 2, x_{CD} \leq 4$
- Constraint: required flow must leave A, i.e., $x_{AB} + x_{AC} = 5$
- Objective function: The total pumping cost:
  $10x_{AB} + 12x_{AC} + 15x_{BC} + 18x_{BD} + 21x_{CD}$

Min-Cost-Flow: the model

$$\min \quad 10x_{AB} + 12x_{AC} + 15x_{BC} + 18x_{BD} + 21x_{CD}$$

$$x_{AB} = x_{BC} + x_{BD}$$
$$x_{AC} + x_{BC} = x_{CD}$$
$$x_{AB} + x_{AC} = 5$$

$$0 \leq x_{AB} \leq 4$$
$$0 \leq x_{AC} \leq 2$$
$$0 \leq x_{BC} \leq 3$$
$$0 \leq x_{BD} \leq 2$$
$$0 \leq x_{CD} \leq 4$$

Min-Cost-Flow in AMPL

```
var x_AB >= 0 <= 4;
var x_AC >= 0 <= 2;
var x_BC >= 0 <= 3;
var x_BD >= 0 <= 2;
var x_CD >= 0 <= 4;

minimize flow_cost:   10*x_AB + 12*x_AC + 15*x_BC
                    + 18*x_BD + 21*x_CD;

cons_nB: x_AB        = x_BC + x_BD;
cons_nC: x_AC + x_BC = x_CD;

outflow: x_AB + x_AC = 5; # same as x_BD + x_CD = 5
```

See also material on http://www.4er.org/CourseNotes/ webpage.