Foundations of Operations Research
Introduction to AMPL and Modellisation

Pierre Hosteins
hosteins@di.unito.it

Politecnico di Milano

October 17th 2013

- Basis of modellisation and use of AMPL
- Illustration on a practical problem
- "Do it yourself"

Basis of Modellisation/Optimisation

Many concrete industrial and scientific problems can be modelled as optimisation problems (minimising a travel distance or costs, maximising profits or quantities produced, minimising the energy of molecule configuration etc...).
These problems need to modellised correctly through an adequate **mathematical formulation** which means one must identify the relevant **variables** and **constraints**. The basic ingredients of an optimisation model are summed up below:

- Variables: they can be continuous (usually positive), integer or binary variables.
- Objectives: what are we trying to minimise/maximise and how does it relate to the decision variables.
- Constraints: what are the limitations of the problem at hand (amount of resources available, time constraints, machine limitations, etc...)

AMPL Framework

- **AMPL** is a computer interface for mathematical optimisation that is able to call different kinds of numerical solver (optimisation softwares).
- The mathematical model can be written in a simple formal language named **Mathprog** $\Rightarrow$ we will learn to formulate practical examples in Mathprog language.
- AMPL takes care of translating Mathprog formulation to a format readable by commercial softwares like **CPLEX** or **MINOS**, which makes it very practical.

AMPL Framework

Mathprog commands and declarations:
The main commands to be used when formulating a problem in AMPL are the following:

- **var**: every variable must be declared, preceded by the `var` command, and followed by the variable type (and if relevant its natural range). For example for a positive continuous variable c, an integer variable i or a binary variable b, we have the following declarations:

  ```
  var c >=0;
  var i, integer;
  var b, binary;
  ```

  Note that each command must be followed by a semi-column caracter: ;

AMPL Framework

Mathprog commands and declarations:
The main commands to be used when formulating a problem in AMPL are the following:

- **Objective**: we must formulate an objective function, starting with a `maximize` or `minimize` command followed by the objective name and its mathematical formulation. For example, a two-variable objective trying to optimise the overall profit can be written down as:
  ```
  maximize Profit:  5 * var1 + 10 * var2;
  ```
- **Constraints**: constraints must be preceded by a `subject to` command followed by a name, e.g.:
  ```
  subject to Total_Production:  var1 + var2 <= 50;
  ```

AMPL Framework

Recommendations:

- Try to **comment** your files (e.g. write comments above each constraint and possibly above the objective and variable declarations) $\Rightarrow$ this does not only help to understand your programs when you read them again, it also saves your teachers some headaches !
  A comment line must start in Mathprog with a # symbol:
  `# This is a line of comment`
- Think also of giving explicit names to your variables whenever possible. In cases it is not practical, try to pick something vaguely related: t or T for a time variable, h or H for the number of hours, p for the percentage etc...

A practical example: Steel Company

A steel company must decide how to allocate next weeks time on a rolling mill. The mill takes unfinished slabs of steel as input and can produce either of two semi-finished products, which we will call **bands** and **coils**. The mills two products come off the rolling line at different **rates**:

- bands: 200 tons/hour
- coils: 140 tons/hour

They also have different **profitabilities**:

- bands: 25€/ton
- coils: 30€/ton

The **maximum production** amounts allowed for the coming week are the following:

- bands: 6000 tons
- coils: 4000 tons

A practical example: Steel Company

The company wants to achieve the following objective:
If 40 hours of production time are available this week, how many tons of bands and how many tons of coils should be produced to bring in the greatest total profit?

Mathematical formulation:

- **Decision variables:** Given the formulation of the problem, the natural decision variables conditionning the total profit would be the amount of tons of coils and bands the factory needs to produce during the coming week:
  T_B = Number of tons of bands to be produced
  T_C = Number of tons of coils to be produced

A practical example: Steel Company

- **Constraints:**
  There are only 40 hours available for production. T_B tons of bands at 200 tons/hour take $T_B/200$ hours of production while T_C tons of coils take $T_C/140$ hours. Therefore:

$$\frac{1}{200}T_B + \frac{1}{140}T_C \leq 40$$

  Moreover, production is limited to the following amounts:

$$0 \leq T_B \leq 6000$$

$$0 \leq T_C \leq 4000$$

A practical example: Steel Company

- **Objective:**
  Using the decision variables, the total profit (in €) is written as:
  (profit per ton of bands) $\times$ T_B $+$ (profit per tons of coils) $\times$ T_C
  thus the objective can be formalised as:

$$\text{Maximise} \quad 25\, T_B + 30\, T_C$$

The answer is easy to find by hand: $T_B = 6000$ tons, $T_C = 1400$ tons and overall profit is 192000€.

ATTENTION: It is crucial to verify you are comparing variables in the same unit $\Rightarrow$ hours get to be compared to hours, NOT minutes and even less tons or euros !

Steel Company: AMPL Implementation

We type the following Mathprog formulation in a file (e.g. example1.mod):

```
var TB >= 0;
var TC >= 0;
maximize Profit: 25 * TB + 30 * TC;
subject to Time: (1/200) * TB + (1/140) * TC <= 40;
subject to B_max: TB <= 6000;
subject to C_max: TC <= 4000;
```

The AMPL commands one needs to type in order to solve the model are:

```
ampl: model example1.mod;
ampl: solve;
ampl: display TB,TC;
```

The second command should display the solver name and the optimal value of the objective, while the last one will give the optimal value of variable T_B and T_C.

Steel Company: AMPL Implementation

There is not a unique way to formulate an optimisation problem: a different set of decision variables can be chosen as long as the overall problem remains equivalent.

Exercise:
Reformulate the Steel Company problem using the **number of hours to dedicate to bands and coils**: H_B and H_C: rewrite objective and constraints, type them into a file `example1bis.mod`, run it in AMPL and find the optimal value of the objective and of variables H_B and H_C.

Steel Company: AMPL Implementation

Answer:

$$\text{Maximise:} \quad 25 * 200 * H_B + 30 * 140 * H_C$$
$$H_B + H_C \leq 40$$
$$200 * H_B \leq 6000$$
$$140 * H_C \leq 4000$$

Formal description

The previous problem was formulated **explicitly** in terms of 2 variables. Should we want to make the model more precise, e.g. add more variables or update the data, we need to modify the model file: it is desirable to write the model in a more **formal manner**, and stock the data in a specific data file.

- Numerical parameters tend to be grouped into indexed structures like **vectors** or **matrices** like b_i or p_{ij} with i and j belonging to **sets**.
- Decision variables tend to become also vectors x_i with index i belonging to a **set** $X = \{1, ..., n\}$ or matrices x_{ij} etc...

Formal description

- Formulae are now written in a formal manner with the use of the **sum operator**, for example the objective of a problem with variables x_i, $i \in X$ associated to costs c_i can be formulated as:

$$\text{Minimise} \quad \sum_{i \in X} c_i * x_i$$

and constraints are formulated in a similar manner:

$$\forall j \in A, \quad \sum_{i \in X} p_{ij} * x_i \leq b_j$$

Note that constraints now become **families of constraints** that are indexed (index j in the example above).

Formal description

Example of declaration of a **data file** in Mathprog language:

```
set M := 1, 2, 3, 4, 5, 6, 7;
set A := Milan, NewYork, Paris, Vienna, Berlin, London;
param b := 5;
param c :=
1 0.5
2 0.7
3 0.25
;
param p :=
[1,1] 2.2
[1,2] 4.0
[2,1] 1.8
[2,2] 3.4
;
```

Formal description

The data sets and parameters **must** be declared in the model file, such that the model knows what is the range of each parameter's indices, for example:

```
set A;
set B;
param b;
param c{A};
param p{A,B};
...
```

Families of constraints that are indexed are declared in the model file in the following way:

```
subject to Max_production_per_machine{i in Machines}:
sum{t in Time} prod[i,t] <= Max[i];
```

Formal description

The data file is loaded after the model in AMPL and before launching the solve command:

```
ampl:  modelfile.mod;
ampl:  datafile.dat;
ampl:  option solver cplex;
ampl:  solve;
```

Exercise: Diet Plan

We need to elaborate a diet plan in order to guarantee a sufficient quantity of nutriments while maintaining the food budget as low as possible. We consider a **set of food types** (pasta, bread, chicken, vegetables, fruits, etc...) that we can buy, each one having a **specific cost**, and a **set of nutriments** that need to be assimilated by the organism (e.g. proteins, etc...). Each type of food brings a **given amount** of each type of nutriment: the body needs to assimilate a **minimum** quantity of each nutriment and should not assimilate more than a **maximum** quantity either.

Find the composition of the diet that minimises the food cost while maintaining sufficient values for each type of nutriment: define the decision values, the set and parameters describing the data, write down the formal formulation of the model and its transcription in Mathprog language. Invent a data file with values of your choice and run the model in AMPL.

Exercise: Diet Plan

Possible solution:
F is the set of food and N the set of nutrients. c_i is the cost of food $i \in F$, q_{ij} the quantity of nutrient j that gives food i. m_j and M_j are the minimum and maximum quantities of nutrient $j \in N$ that should be absorbed. Variables x_i represent the quantity of food i that we buy.

$$\text{Minimise:} \quad \sum_{i \in F} c_i x_i$$

$$\forall j \in N, \quad \sum_{i \in F} q_{ij} x_i \geq m_j$$

$$\forall j \in N, \quad \sum_{i \in F} q_{ij} x_i \leq M_j$$

Have fun at home !: The Shoe Company

A shoe company wants to plan its production. It can produce a given set of shoes, using a set of employees. Each worker takes a certain number of minutes to produce a pair of shoes of each model and can work a maximum amount of minutes. Each model is sold at a certain price and can be hoped to be sold at a maximum amount of units on the market. Formulate the problem of maximizing the overall money gain (be careful to consider each worker separately).