

Mathematical Programming: Modeling and Applications

Giacomo Nannicini

`giacomon@lix.polytechnique.fr`

The Basics

- Open the terminal

- Type `ampl`

```
ILOG AMPL 10.100, licensed to "ecolepolytechnique-palaiseau".  
AMPL Version 20060626 (Linux 2.6.9-5.ELsmp)  
ampl:
```

- Now you are in interactive mode; everything you type will be interpreted by AMPL
- Most commands are ended by “;” (otherwise AMPL will not understand)
- type `quit;` to exit

Our First Problem

- Declare two variables:

```
var x1;
```

```
var x2 > = 0;
```

- Declare the objective function:

```
minimize objective: -x1 + 2*x2;
```

- Declare the constraints:

```
subject to constr: x1 + x2 = 10;
```

- Now we have to choose a solver; choose `cplex`:

```
option solver cplex;
```

- And we are ready to go:

```
solve;
```

- Display the result:

```
display x1, x2;
```

Our First Problem

```
ILOG AMPL 10.100, licensed to "ecolepolytechnique-palaiseau".
```

```
AMPL Version 20060626 (Linux 2.6.9-5.ELsmp)
```

```
AMPL: var x1;
```

```
AMPL: var x2 >= 0;
```

```
AMPL: minimize objective: -x1 + 2*x2;
```

```
AMPL: subject to constr: x1 + x2 = 10;
```

```
AMPL: option solver cplex;
```

```
AMPL: solve;
```

```
ILOG CPLEX 10.100, licensed to "ecolepolytechnique-palaiseau",
```

```
CPLEX 10.1.0: optimal solution; objective -10
```

```
0 dual simplex iterations (0 in phase I)
```

```
AMPL: display x1, x2;
```

```
x1 = 10
```

```
x2 = 0
```

The Diet Problem

- The “diet problem” is probably the first mathematical program ever formulated ('30s – '40s)
- We have to buy some foods in order to determine a diet which satisfies minimum nutritional requirements
- We would like to pay as little as possible: we want to **minimize** the cost
- It is very important to model a whole class of problems, and not just a particular instance
- However, this is our *first* serious problem, so we will keep it simple

The Diet Problem: Data

- We have a choice between 3 possible foods: apple, bread, candy
- One apple contains 5 units of vitamin A, 5 units of vitamin B, 10 units of vitamin C
- One bread contains 2 units of vitamin A, 10 units of vitamin B, 1 units of vitamin C
- One candy contains 3 units of vitamin A
- The minimum requirement over one day is of 30 units of vitamin A, 50 units of vitamin B, 30 units of vitamin C
- One apple costs 2 euros, one bread costs 1 euro, one candy costs 0.2 euros

The Diet Problem: Solution



```
ILOG AMPL 10.100, licensed to "ecolepolytechnique-palaiseau".
AMPL Version 20060626 (Linux 2.6.9-5.ELsmp)
AMPL: var apples >= 0;
AMPL: var breads >= 0;
AMPL: var candies >= 0;
AMPL: minimize cost: 2*apples + 1*breads + 0.2*candies;
AMPL: subject to vitamin_a: 5*apples + 2*breads + 3*candies >= 30;
AMPL: subject to vitamin_b: 5*apples + 10*breads >= 50;
AMPL: subject to vitamin_c: 10*apples + 1*breads >= 30;
AMPL: option solver cplex;
AMPL: solve;
ILOG CPLEX 10.100, licensed to "ecolepolytechnique-palaiseau",
options: e m b q use=8
CPLEX 10.1.0: optimal solution; objective 9.578947368
3 dual simplex iterations (0 in phase I)
AMPL: display apples, breads, candies;
apples = 2.63158
breads = 3.68421
candies = 3.15789
```

Integer Solution

```
AMPL: var apples >= 0, integer;
AMPL: var breads >= 0, integer;
AMPL: var candies >= 0, integer;
AMPL: minimize cost: 2*apples + 1*breads + 0.2*candies;
AMPL: subject to vitamin_a: 5*apples + 2*breads + 3*candies >= 30;
AMPL: subject to vitamin_b: 5*apples + 10*breads >= 50;
AMPL: subject to vitamin_c: 10*apples + 1*breads >= 30;
AMPL: option solver cplex;
AMPL: solve;
ILOG CPLEX 10.100, licensed to "ecolepolytechnique-palaiseau",
options: e m b q use=8
CPLEX 10.1.0: optimal integer solution; objective 10.6
3 MIP simplex iterations
0 branch-and-bound nodes
AMPL: display apples, breads, candies;
apples = 3
breads = 4
candies = 3
```

Some More History

- The problem of minimizing the cost of a diet for the army was formulated as a mathematical program by George Stiegler
- He guessed a solution of \$39.93 per year (1939 prices)
- The problem was solved in 1947 using the simplex method: 9 clerks worked on it with hand-operated desk calculators
- It took approximately 120 man-days to find the solution: the problem had 9 equations and 71 unknowns
- Optimal solution: \$39.69
- Luckily, now we have Cplex!

AMPL Basics: Model and Data



- We should write a general model that works for the whole class of problems
- The problem data should be independent from the model
- Purpose: reuse the same model with different data
- The model should be “abstract”

Parameters and Sets

- A parameter is a symbol that stands for *any* scalar:
`param c i`

- We can declare a set:
`set A i`

- A set can be used to index *many* things!

1. `var x{A} i`

2. `param c{A} i`

3. `subject to constr{i in A}: x[i] >= 0 i`

4. `subject to cover: sum{i in A} c[i]*x[i] >= 1 i`

⋮

Multidimensional Indices

- How do we declare a matrix?

```
set Rows;
```

```
set Columns;
```

```
param matrix{Rows,Columns};
```

```
subject to constr{i in Rows}: sum{j in  
Columns} matrix[i,j]*x[j] >= 1;
```

- All the parameters must be defined somewhere — that's the purpose of the .dat file!

The Data File

- Entities are defined by using :=

- Some examples:

```
set A := vitamin_a, vitamin_b, vitamin_c;
```

```
param c:=
```

```
vitamin_a 1
```

```
vitamin_b 10
```

```
vitamin_c 100;
```

```
param matrix: 1 2 3 :=
```

```
1 0 0 1
```

```
2 0 1 1
```

```
3 1 1 1;
```

Using Model and Data

- To load a model:
`model file.mod;`
- To load a data file:
`data file.dat;`
- To reset (in case anything goes wrong):
`reset;`
- The solution process is started as usual:
`option solver cplex; solve;`

Model The Diet Problem

- Write a model file for the diet problem:
 1. Define the sets
 2. Define the variables
 3. Define the parameters
 4. Write constraints and objective function
- Then write the data file for the instance that we solved before
- Remember: the solution should cost 9.578947368

Solution: Model

```
# Declare sets
set Foods;
set Nutrients;

# Declare variables
var quantity{Foods} >= 0;

# Declare parameters
param cost{Foods} >= 0;
param amount{Nutrients, Foods} >= 0;
param minimum{Nutrients} >= 0;

# Objective function
minimize obj: sum{j in Foods} cost[j]*quantity[j];

# Declare constraints
subject to minnutr{i in Nutrients}:
    sum{j in Foods} amount[i,j]*quantity[j] >= minimum[i];
```

Solution: Data

```
set Foods := apples, breads, candies;  
set Nutrients := vitamin_a, vitamin_b, vitamin_c;
```

```
param cost :=  
    apples 2  
    breads 1  
    candies 0.2;
```

```
param amount:    apples breads candies :=  
    vitamin_a 5      2      3  
    vitamin_b 5      10     0  
    vitamin_c 10     1      0;
```

```
param minimum :=  
    vitamin_a 30  
    vitamin_b 50  
    vitamin_c 30;
```

Run File

- Instead of writing several times the same commands, we can write a diet.run file that contains a list of instructions to execute

```
reset;  
model diet.mod;  
data diet.dat;  
option solver cplex;  
solve;  
display quantity;
```

- To run it:
ampl < diet.run
or
cat diet.run | ampl

That's all!

```
ILOG AMPL 10.100, licensed to "ecolepolytechnique-palaiseau".  
AMPL Version 20060626 (Linux 2.6.9-5.ELsmp)  
ILOG CPLEX 10.100, licensed to "ecolepolytechnique-palaiseau",  
options: e m b q use=8  
CPLEX 10.1.0: optimal solution; objective 9.578947368  
3 dual simplex iterations (0 in phase I)  
quantity [*] :=  
  apples  2.63158  
  breads  3.68421  
candies  3.15789  
;
```