Une introduction à A Mathematical Programming Language (AMPL)

A Modeling Language for Mathematical Programming

Nicolas Jozefowiez
nicolas.jozefowiez@insa-toulouse.fr
INSA Toulouse

Qu'est ce qu'AMPL ?

- AMPL est un langage informatique pour décrire des problèmes de : production, distribution, mélange, ordonnancement ...
- AMPL est un environnement de commande intéractif utilisant des notations algébriques pour
  - Faciliter la formulation de modèles
  - Communiquer avec de nombreux solveurs (CPLEX, MINOS, KNITRO ...)
  - Analyser des solutions
- AMPL est un logiciel flexible et pratique pour prototyper et développer rapidement des modèles
- Ressources : www.ampl.com (download, FAQ, liens ...)

Un exemple d'aciérie

Une aciérie produit des bandes et des rouleaux métalliques. Elle fonctionne 40 heures par semaine. Les vitesses de production sont de 200 bandes par heure et 140 rouleaux par heure.

Les bandes sont vendues 25 euros l'unité ; les rouleaux 30 euros l'unité. Le marché est limité : il est impossible de vendre plus de 6000 bandes et plus de 4000 rouleaux par semaine.

Comment maximiser le profit sur une semaine ?

Modélisation

- Variables :
  - x_1 le nombre de bandes à produire
  - x_2 le nombre de rouleaux à produire
- Paramètres :
  - Heures ouvrées : 40
  - Vitesses de production : 200, 140
  - Prix de vente : 25, 30
  - Limite du marché : 6000, 4000

Programme linéaire

$$\max z = 25x_1 + 30x_2 \text{ (Maximisation du profit)}$$

$$x_1 \leq 6000 \text{ (Limitation du marché : bandes)}$$

$$x_2 \leq 4000 \text{ (Limitation du marché : rouleaux)}$$

$$\frac{1}{200}x_1 + \frac{1}{140}x_2 \leq 40 \text{ (Limitation de la production)}$$

$$x_1, x_2 \geq 0$$

Résolution par AMPL (interprétation)

```
ampl: var x1 >= 0;
ampl: var x2 >= 0;
ampl: maximize z : 25*x1 + 30*x2;
ampl: subject to bandes : x1 <= 6000;
ampl: subject to rouleaux : x2 <= 4000;
ampl: subject to production : (1/200)*x1 + (1/140)*x2 <= 40;
ampl: solve;
MINOS 5.5: optimal solution found.
2 iterations, objective 192000
ampl: display z, x1, x2;
z = 19200
x1 = 6000;
x2 = 1400;
```


Abstraction du modèle

- But : abstraire au maximum
- On dégage de l'énoncé du problème des *ensembles*
- Ce sont par ces ensembles que les variables, les paramètres et les contraintes seront identifiés
- Ensembles ⇔ types énumérés en informatique
- Exemple sur l'aciérie

Fichier : PL + données

```
set PROD;
param heures_ouvrees >= 0;
param vitesse_production {PROD} >= 0;
param prix_vente {PROD} >= 0;
param vente_max {PROD} >= 0;
var qte_produite {p in PROD} >= 0, <= vente_max [p];
maximize profit :
    sum {p in PROD} qte_produite [p] * prix_vente [p];
subject to production_limitee :
    sum {p in PROD}
       (qte_produite [p] / vitesse_production [p]) <= heures_ouvrees;

data;

set PROD := bandes rouleaux;
param heures_ouvrees := 40;
param: vitesse_production prix_vente vente_max :=
bandes          200               25        6000
rouleaux        140               30        4000;
```

Résolution dans AMPL

- Les suffixes lb et ub accolés aux variables désignent les bornes inférieures et supérieures

```
ampl: model acierie1;
ampl: solve;
MINOS 5.5: optimal solution found.
2 iterations, objective 192000
ampl: display qte_produite.lb, qte_produite, qte_produite.ub;
:        qte_produite.lb qte_produite qte_produite.ub    :=
bandes             0             6000           6000
rouleaux           0             1400           4000
```


Ajout d'un produit

- Il suffit de modifier les données, le modèle abstrait ne change pas

```
set PROD := bandes rouleaux poutres;
param heures_ouvrees := 40;
param: vitesse_production prix_vente vente_max :=
bandes         200              25        6000
rouleaux       140              30        4000
poutres        160              29        3500;
```

- On résout :

```
ampl: model acierie2;
ampl: solve;
MINOS 5.5: optimal solution found.
2 iterations, objective 196400
ampl: display qte_produite.lb, qte_produite, qte_produite.ub;
:        qte_produite.lb qte_produite qte_produite.ub    :=
bandes             0             6000           6000
poutres            0             1600           3500
rouleaux           0             1400           4000;
```


Une modification

- Il est irréaliste de ne plus produire de rouleaux
- Modification du programme pour forcer la production de rouleaux

```
set PROD;
param heures_ouvrees >= 0;
param vitesse_production {PROD} >= 0;
param prix_vente {PROD} >= 0;
param vente_max {PROD} >= 0;
param vente_min {PROD} >= 0;
var qte_produite {p in PROD} >= vente_min [p], <= vente_max [p];
maximize profit :
    sum {p in PROD} qte_produite [p] * prix_vente [p];
subject to production_limitee :
    sum {p in PROD} (qte_produite [p] / vitesse_production [p]) <= heures_ouvrees;

data;

set PROD := bandes rouleaux;
param heures_ouvrees := 40;
param: vitesse_production prix_vente vente_max vente_min :=
bandes         200             25        6000      1000
rouleaux       140             30        4000       500
poutres        160             29        3500       750;
```


Une modification (résolution)

```
ampl: model acierie3;
ampl: solve;
MINOS 5.5: optimal solution found.
2 iterations, objective 194828.5714
ampl: display qte_produite.lb, qte_produite, qte_produite.ub;
:        qte_produite.lb qte_produite qte_produite.ub    :=
bandes          1000             6000           6000
poutres          750             1028.57        3500
rouleaux         500              500           4000;
```

Résolution en nombres entiers

- Utilisation du qualificatif `integer` sur les variables
- Utilisation du solveur cplex

```
...
var qte_produite {p in PROD} integer >= vente_min [p], <= vente_max [p];
...
```

- On résout :

```
ampl: model acierie3_bis;
ampl: option solver cplex;
solve;
CPLEX 8.0.0: optimal integer solution within mipgap or absmipgap;
objective 194818
1 MIP simplex iterations
0 branch-bound nodes
ampl: display qte_produite.lb, qte_produite, qte_produite.ub;
:        qte_produite.lb qte_produite qte_produite.ub    :=
bandes          1000            5999          6000
poutres          750            1027          3500
rouleaux         500             502          4000;
```


Planification sur plusieurs semaines

- On souhaite planifier l'activité sur N = 3 semaines
- Le prix de vente et le nombre d'heures ouvrées varient avec les semaines
- Possibilité de stockage limitée : 1000 unités de produits maximum
- Le stock initial est vide
- Le stock final doit être vide

Que fait-on ?

- Ensemble SEMS : intervalle d'entiers consécutifs
- Prix de vente, quantités produites : indicés par les produits et les semaines
- Les contraintes sur les capacités de production sont maintenant aussi indicées par les produits et par les semaines
- Différentes variables pour les quantités produites, les quantités vendues, les quantités disponibles en stock en début de semaine
- Contraintes d'égalité pour lier ces différentes quantités : indicées par les produits et les semaines
- Contraintes d'égalité pour exprimer que les stocks initiaux et finaux sont vides

Planification sur plusieurs semaines (II)

```
set PROD;
param vitesse_production {PROD} >= 0;
param vente_min {PROD} >= 0;
param vente_max {PROD} >= 0;
param N integer >= 0;
set SEMS := 1 .. N;
param heures_ouvrees {SEMS} >= 0;
param prix_vente {SEMS, PROD} >= 0;
param stock_max >= 0;
var qte_produite {SEMS, PROD} >= 0;
var qte_vendue {s in SEMS, p in PROD} >= vente_min [p], <= vente_max [p];
var qte_stock {1 .. N+1, PROD} >= 0;
```

Planification sur plusieurs semaines (III)

```
maximize profit :
    sum {s in SEMS, p in PROD} qte_vendue [s, p] * prix_vente [s, p];

subject to production_limitee {s in SEMS} :
    sum {p in PROD}
       (qte_produite [s, p] / vitesse_production [p]) <=
           heures_ouvrees [s];

subject to stock_initial {p in PROD} :
    qte_stock [1, p] = 0;

subject to stock_final {p in PROD} :
    qte_stock [N+1, p] = 0;

subject to equilibre {s in SEMS, p in PROD} :
    qte_stock [s, p] + qte_produite [s, p] =
         qte_stock [s+1, p] + qte_vendue [s, p];
```


Planification sur plusieurs semaines (IV)

```
data;

set PROD := bandes rouleaux poutres;
param: vitesse_production vente_max vente_min :=
bandes         200           6000        1000
rouleaux       140           4000         500
poutres        160           3500         750;

param heures_ouvrees :=
1       40
2       20
3       35;

param N := 3;
param stock_max := 1000;
param prix_vente:
    bandes rouleaux poutres :=
1       25      30      29
2       27      30      28
3       29      30      20;
```


Planification sur plusieurs semaines (V)

```
apml: model acierie4;
ampl: solve;
...
ampl: display qte_produite, qte_stock, qte_vendue;
:           qte_produite qte_stock qte_vendue :=
1 bandes      4044.64        0       2044.64
1 poutres     1500           0        750
1 rouleaux    1456.25        0        500
2 bandes      4000        2000       6000
2 poutres        0         750        750
2 rouleaux       0         956.25     500
3 bandes      6000           0       6000
3 poutres      750           0        750
3 rouleaux      43.75      456.25     500
4 bandes         -           0          -
4 poutres        -           0          -
4 rouleaux       -           0          -
```


Analyse de sensibilité

- Modification d'un coefficient de la fonction objectif

  Déterminer l'intervalle de variation d'un coefficient c_j de la fonction objectif pour lequel la solution optimale reste la même

- Modification d'un coefficient du membre de droite

  Déterminer l'intervalle de variation d'un coefficient b_i du membre de droite pour lequel les variables de base associées à la solution optimale restent les mêmes

- AMPL permet d'accéder à ces informations

- Illustration sur le premier modèle de l'aciérie

Coût réduit d'une variable

- Soit *rc* le coût réduit d'une variable *v* dans une fonction à maximiser
- Si on augmente de ε la borne sur v alors la valeur optimale de l'objectif augmente de *rc*×ε
- Commande : *v.rc*

```
ampl: reset;
ampl: model acierie1;
ampl: solve;
MINOS 5.5: optimal solution found.
2 iterations, objective 192000
ampl: display qte_produite.rc;
qte_produite.rc [*] :=
  bandes  4
rouleaux  7.10543e-15
ampl: let vente_max ["bandes"] := 6001;
ampl: solve;
MINOS 5.5: optimal solution found.
1 iterations, 192004
```

- S'applique à la borne la plus proche de la valeur de la variable

Valeur et bornes d'une contrainte

- Valeur du corps (partie gauche) d'une contrainte *c* : *c.body*
- Borne supérieure d'une contrainte *c* : *c.ub*
- Borne inférieure d'une contrainte *c* : *c.lb*
- Si le sens de *c* est ≤ et *c.body = c.ub* alors la contrainte est active

```
ampl: display production_limitee.body, production_limitee.ub;
production_limitee.body = 40
production_limitee.ub = 40
ampl: display production_limitee.lb;
production_limitee.lb = -Infinity
```

- Si le sens de *c* est ≥ et *c.body = c.lb* alors la contrainte est active

Valeur marginale d'une contrainte

- Sens : si on augmente la valeur de la borne (inférieure ou supérieure) de la contrainte d'une "petite" quantité ε, l'objectif réalisé de la solution optimale augmente de *sp*×ε où *sp* est la valeur marginale de la contrainte.
- Obtenir la valeur marginale de la contrainte *c* : display *c*

```
ampl: display production_limitee;
production_limitee = 4200;
ampl: let heures_ouvrees := 41;
ampl: solve;
MINOS 5.5: optimal solution found.
1 iterations, objective 196200
ampl: display production_limitee.body, production_limitee.ub;
production_limitee.body = 41
production_limitee.ub = 41
```

- La valeur marginale s'applique à la borne la plus proche de la valeur de la contrainte
- Uniquement pour les PL en variables réelles