Université d'Oran 1 Ahmed Ben Bella
Module Programmation linéaire
Année universitaire 2017/2018

Oran, Décembre 2017

Projet PL TP :

Résolution et implémentation d'un problème de d'affectation et de linéarisation sous AMPL

Travail réalisé par :

- CHOUIB Chawki (groupe 1)
- DJEZIRI Ibtissem (groupe 1)

Exercice 1 – Problème du mini-projet :

- On considère un groupe de 7 étudiants notés A, B, C, D, E, F et G.
- Pour un mini-projet, les étudiants se mettent en équipes mais il faut respecter les incompatibilités entre les étudiants.
- Dans le tableau ci-dessous, chaque croix indique une incompatibilité entre les étudiants correspondants.

•	A	B	C	D	E	F	G
A	•	X	•	•	X	X	•
B	X	•	X	•	•	•	•
C	•	X	•	X	X	X	•
D	•	•	X	•	X	X	X
E	X	•	X	X	•	X	X
F	X	•	X	X	X	•	•
G	•	•	•	X	X	•	•

1. Modélisation :

- L'objective est de faire une affectation des étudiants avec les contraintes d'incompatibilité de chaque étudiant et de créer le nombre de groupe au minimum possible.
- On peu déduire cette matrice "D" du tableaux ci dessus :

D =

1	0	1	1	0	0	1
0	1	0	1	1	1	1
1	0	1	0	0	0	1
1	1	0	1	0	0	0
0	1	0	0	1	0	0
0	1	0	0	0	1	1
1	1	1	0	0	1	1

- Soit le PL suivant :

Variables :

x_{ij} : Etudiant i affecter avec l'étudiant j (j , i = { 1..7 }).

La fonction objective :

Minimiser [R = $\sum_{i=1}^{7} \sum_{j=1}^{7} (D_{ij} * x_{ij})$]

L'ensemble des contraintes :

$$\sum_{j=1}^{7}(x_{ij}) = 1 \quad \forall i \ = \ \{1..7\}$$ (#contrainte pour chaque lignes(7 ligne))

$$\sum_{i=1}^{7}(x_{ij}) \ \leq \ 4 \ \ pour j \ = 1$$ (#contrainte colonne de l'étudiant A)

$$\sum_{i=1}^{7}(x_{ij}) \ \leq \ 5 \ \ pour j \ = 2$$ (#contrainte colonne de l'étudiant B)

$$\sum_{i=1}^{7}(x_{ij}) \ \leq \ 3 \ \ pour j \ = 3$$ (#contrainte colonne de l'étudiant C)

$$\sum_{i=1}^{7}(x_{ij}) \ \leq \ 3 \ \ pour j \ = 4$$ (#contrainte colonne de l'étudiant D)

$$\sum_{i=1}^{7}(x_{ij}) \ \leq \ 2 \ \ pour j \ = 5$$ (#contrainte colonne de l'étudiant E)

$$\sum_{i=1}^{7}(x_{ij}) \ \leq \ 3 \ \ pour j \ = 6$$ (#contrainte colonne de l'étudiant F)

$$\sum_{i=1}^{7}(x_{ij}) \ \leq \ 5 \ \ pour j \ = 7$$ (#contrainte colonne de l'étudiant G)

$x_{ij} = \{0,1\} \ \forall \mathrm{i\,j} \ = \ \{1..7\}$ (#valeur binaire)

2. Implémentation sous AMPL :

Fichier *.mod :

```
#nombre d'etudiant (1-7)
param NombreEtudiant:= 7 ;

#matrice des compactibilite et incompactibilte des etudiants
param Matrice { i in 1..NombreEtudiant , j in 1..NombreEtudiant} binary ;

#matrice d'affection de l'etudiant i dans avec l'etudiant j ou le groupe j
var Affectation { i in 1..NombreEtudiant , j in 1..NombreEtudiant } binary ;

#fonction objective
minimize FonctionObjective :  sum{ i in 1..NombreEtudiant ,j in 1..NombreEtudiant } (Matrice[i,j] * Affectation[i,j]);



#contrainte des lignes c'est l'etudiant A ... G
subject to CompactibleEtudiantA :  sum{ j in 1..NombreEtudiant} Affectation[1,j] = 1 ;
subject to CompactibleEtudiantB :  sum{ j in 1..NombreEtudiant} Affectation[2,j] = 1 ;
subject to CompactibleEtudiantC :  sum{ j in 1..NombreEtudiant} Affectation[3,j] = 1 ;
subject to CompactibleEtudiantD :  sum{ j in 1..NombreEtudiant} Affectation[4,j] = 1 ;
subject to CompactibleEtudiantE :  sum{ j in 1..NombreEtudiant} Affectation[5,j] = 1 ;
subject to CompactibleEtudiantF :  sum{ j in 1..NombreEtudiant} Affectation[6,j] = 1 ;
subject to CompactibleEtudiantG :  sum{ j in 1..NombreEtudiant} Affectation[7,j] = 1 ;


#contrainte des colones c'est les groupes possible dans chaque ligne pour un etudiant
subject to CompactibleGroupe1 :  sum{ i in 1..NombreEtudiant} Affectation[i,1] <= 5 ;
subject to CompactibleGroupe2 :  sum{ i in 1..NombreEtudiant} Affectation[i,2] <= 4 ;
subject to CompactibleGroupe3 :  sum{ i in 1..NombreEtudiant} Affectation[i,3] <= 3 ;
subject to CompactibleGroupe4 :  sum{ i in 1..NombreEtudiant} Affectation[i,4] <= 3 ;
subject to CompactibleGroupe5 :  sum{ i in 1..NombreEtudiant} Affectation[i,5] <= 2 ;
subject to CompactibleGroupe6 :  sum{ i in 1..NombreEtudiant} Affectation[i,6] <= 3 ;
subject to CompactibleGroupe7 :  sum{ i in 1..NombreEtudiant} Affectation[i,7] <= 5 ;
```

Fichier *.dat :

```
#codification A=1 ........ G=7
param Matrice:1 2 3 4 5 6 7:=
            1 1 0 1 1 0 0 1
            2 0 1 0 1 1 1 1
            3 1 0 1 0 0 0 1
            4 1 1 0 1 0 0 0
            5 0 1 0 0 1 0 0
            6 0 1 0 0 0 1 1
            7 1 1 1 0 0 1 1 ;
```

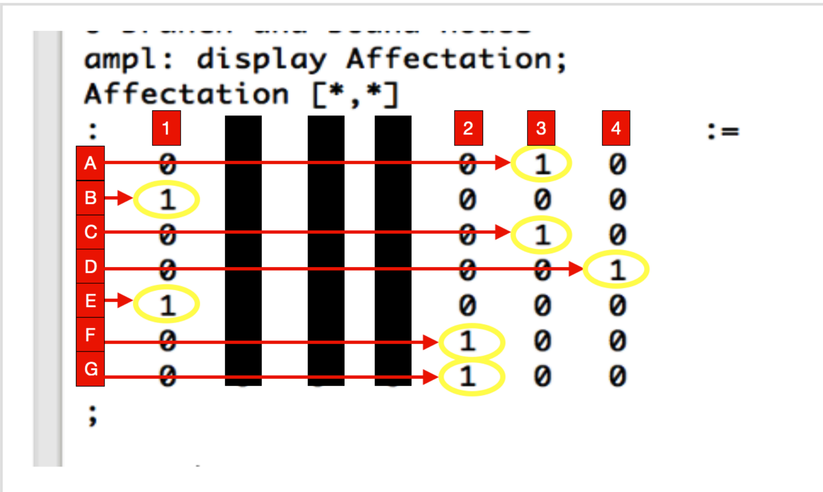
3. Recherche des solutions :

L'exécution :

```
AMPL
ampl: reset;
ampl: model exo1.mod;
ampl: data exo1.dat;
ampl: solve;
CPLEX 12.7.1.0: optimal integer solution; objective 0
0 MIP simplex iterations
0 branch-and-bound nodes
ampl: display Affectation , FonctionObjective;
Affectation [*,*]
:   1   2   3   4   5   6   7    :=
1   0   0   0   0   0   1   0
2   1   0   0   0   0   0   0
3   0   0   0   0   0   1   0
4   0   0   0   0   0   0   1
5   1   0   0   0   0   0   0
6   0   0   0   0   1   0   0
7   0   0   0   0   1   0   0
;

FonctionObjective = 0
```

La résolution : la Solution optimale (obtenue par CPLEX) :



Donc on conclut qu'il faut 4 groupe avec comme affectation :
- Groupe 1 : B et E
- Groupe 2 : F et G
- Groupe 3 : A et C
- Groupe 4 : D

Fonction objective = 0 car c'est une affectation optimal avec un coup nul.

Exercice 2 – Problème d'affectation des équipages :

- Un problème important des compagnies aériennes consiste à constituer de façon efficace des équipages pour ses vols.
- Pour un équipage donné, une rotation consiste en une succession de services de vol débutant et se terminant en une même ville.
- Comme il y a un coût associé à chaque séquence de vols, la compagnie cherche à minimiser les coûts d'affectation des équipages aux séquences tout en assurant le service sur chacun des vols.
- Considérons par exemple un problème avec 11 vols et 12 séquences de vols, dont les données sont décrites dans la Table 1.

Vol | Séquence	1	2	3	4	5	6	7	8	9	10	11	12
1	1			1			1			1		
2		1			1			1			1	
3			1			1			1			1
4				1			1		1	1		1
5	1					1				1	1	
6				1	1				1			
7							1	1		1	1	1
8		1		1	1				1			
9					1			1			1	
10			1				1	1				1
11						1			1	1	1	1
Coût	2	3	4	6	7	5	7	8	9	9	8	9

Table 1: Affectation d'équipages.

$$x_j = \begin{cases} 1 & \text{si la séquence de vols } j \text{ est affectée;} \\ 0 & \text{sinon.} \end{cases}$$

1. Modélisation :

- L'objectif est de minimiser le coût des affectations tout en sachant le nombre d'équipage qui est égale à 3, c'est a dire $\sum_{j=1}^{12} x_j = 3$.
- Le service doit être assuré sur chacun des vols c'est à dire l'union des sous-ensemble (x_j doit satisfaire l'ensemble des vols avec un coût minimal) .
- Soit le PL suivant :

  Variables :

  x_j: affectation des équipes a la séquence j (j = { 1..12 }).

  La fonction objective :

  Minimiser [G = $2x_1 + 3x_2 + 4x_3 + 6x_4 + 7x_5 + 5x_6 + 7x_7 + 8x_8 + 9x_9 + 9x_{10} + 8x_{11} + 9x_{12}$]

L'ensemble des contraintes :

$x_1+x_4+x_7+x_{10}$	≥ 1	(#contrainte du vol 1)
$x_2+x_5+x_8+x_{11}$	≥ 1	(#contrainte du vol 2)
$x_3+x_6+x_9+x_{12}$	≥ 1	(#contrainte du vol 3)
$x_4+x_7+x_9+x_{10}x_{12}$	≥ 1	(#contrainte du vol 4)
$x_1+x_6+x_{10}+x_{11}$	≥ 1	(#contrainte du vol 5)
$x_4+x_5+x_9$	≥ 1	(#contrainte du vol 6)
$x_7+x_8+x_{10}+x_{11}+x_{12}$	≥ 1	(#contrainte du vol 7)
$x_2+x_4+x_5+x_9$	≥ 1	(#contrainte du vol 8)
$x_5+x_8+x_{11}$	≥ 1	(#contrainte du vol 9)
$x_3+x_7+x_8+x_{12}$	≥ 1	(#contrainte du vol 10)
$x_6+x_9+x_{10}+x_{11}+x_{12}$	≥ 1	(#contrainte du vol 11)
$\sum_{j=1}^{12} x_j$	$= 3$	(#contrainte des 3 équipes)

$x_j = \{ 0 - 1 \}$ avec $j = \{ 1..12 \}$ (#valeur binaire)

2. Implémentation sous AMPL :

Fichier *.mod :

```
param NombreSequence := 12 ;

param CoutSequence{ j in 1..NombreSequence } ;

var SequenceVols{ j in 1..NombreSequence } binary ;


minimize affectation : sum{ j in 1..NombreSequence } SequenceVols[j] * CoutSequence[j] ;


subject to vol01 : SequenceVols[1] + SequenceVols[4] + SequenceVols[7] + SequenceVols[10]                      >= 1 ;
subject to vol02 : SequenceVols[2] + SequenceVols[5] + SequenceVols[8] + SequenceVols[11]                      >= 1 ;
subject to vol03 : SequenceVols[3] + SequenceVols[6] + SequenceVols[9] + SequenceVols[12]                      >= 1 ;
subject to vol04 : SequenceVols[4] + SequenceVols[7] + SequenceVols[9] + SequenceVols[10] + SequenceVols[12]   >= 1 ;
subject to vol05 : SequenceVols[1] + SequenceVols[6] + SequenceVols[10] + SequenceVols[11]                     >= 1 ;
subject to vol06 : SequenceVols[4] + SequenceVols[5] + SequenceVols[9]                                         >= 1 ;
subject to vol07 : SequenceVols[7] + SequenceVols[8] + SequenceVols[10] + SequenceVols[11] + SequenceVols[12]  >= 1 ;
subject to vol08 : SequenceVols[2] + SequenceVols[4] + SequenceVols[5] + SequenceVols[9]                       >= 1 ;
subject to vol09 : SequenceVols[5] + SequenceVols[8] + SequenceVols[11]                                        >= 1 ;
subject to vol10 : SequenceVols[3] + SequenceVols[7] + SequenceVols[8] + SequenceVols[12]                      >= 1 ;
subject to vol11 : SequenceVols[6] + SequenceVols[9] + SequenceVols[10] + SequenceVols[11] + SequenceVols[12]  >= 1 ;

subject to equipage : sum { j in 1..NombreSequence } SequenceVols[j] = 3 ;
```

Fichier *.dat :

```
param CoutSequence:=
          1 2
          2 3
          3 4
          4 6
          5 7
          6 5
          7 7
          8 8
          9 9
          10 9
          11 8
          12 9;
```

3. Recherche des solutions :

L'exécution :

```
AMPL
ampl: reset;
ampl: model exo2.mod;
ampl: data exo2.dat;
ampl: solve;
CPLEX 12.7.1.0: optimal integer solution; objective 18
8 MIP simplex iterations
0 branch-and-bound nodes
ampl: display SequenceVols , affectation ;
SequenceVols [*] :=
 1   1
 2   0
 3   0
 4   0
 5   1
 6   0
 7   0
 8   0
 9   0
10   0
11   0
12   1
;

affectation = 18
```

La résolution : la Solution optimale (obtenue par CPLEX) :

$(x_1, x_2 , x_3 , x_4, x_5, x_6 , x_7 , x_8 ,x_9 , x_{10}, x_{11}, x_{12})$ = (1, 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 1).

- Donc pour avoir un coût minimal égale à 18, on affecte les 3 équipes dans la séquence de vole 1 , 5 et 12.

Exercice 3 : Problème de linéarisation

- On considère le problème mathématique suivant :

La fonction objectif :
Minimiser [Z = Max ($7x_1 + 4x_2 + x_3$, $x_1 - 3x_3$)].

L'ensemble des contraintes :

$6x_1 + 3x_2 + 4x_3 \geq 2$

$3x_1 + x_2 + 2x_3 \leq 7$

$x_1, x_2, x_3 \geq 0$

1. Modélisation :

- On remarque que la fonction objectif n'est pas linéaire , donc on va la linéariser en utilisant la formule qui permet d'obtenir le maximum de 2 nombres :

Max(a, b) = 1/2 (a + b + |a - b|)

On pose **a** = ($7x_1 + 4x_2 + x_3$)
On pose **b** = ($x_1 - 3x_3$)

L'application de la formule :

$[(7x_1 + 4x_2 + x_3) + (x_1 - 3x_3) + \left| (7x_1 + 4x_2 + x_3) - (x_1 - 3x_3) \right|] \div 2 =$

$[(8x_1 + 4x_2 - 2x_3) + \left| 6x_1 + 4x_2 + 4x_3 \right|] \div 2 =$

$[(8x_1 + 4x_2 - 2x_3) + (6x_1 + 4x_2 + 4x_3)] \div 2 =$

$[14x_1 + 8x_2 + 2x_3] \div 2 =$

$7x_1 + 4x_2 + x_3$

Donc en appliquant la formule on obtient la fonction objectif :
Minimiser [$Z = 7x_1 + 4x_2 + x_3$]

2. Implémentation sous AMPL :

Fichier *.mod :

```
var X1 >= 0;
var X2 >= 0;
var X3 >= 0;

minimize objective : 7*X1 + 4*X2 + X3;

subject to contrainte1 : 6*X1 + 3*X2 + 4*X3 >= 2 ;
subject to contrainte2 : 3*X1 + X2 + 2*X3   <= 7 ;
```

3. Recherche des solutions :

L'exécution :

```
AMPL
ampl: reset;
ampl: model exo3.mod;
ampl: solve;
CPLEX 12.7.1.0: optimal solution; objective 0.5
1 dual simplex iterations (0 in phase I)
ampl: display X1,X2,X3, objective ;
X1 = 0
X2 = 0
X3 = 0.5
objective = 0.5
```

La résolution : la Solution optimale (obtenue par CPLEX) :

La solution (x_1 , x_2 , x_3) = (0 , 0 , 0.5)
La fonction objectif Z = 0.5 .