Répartition d'étudiants

8 septembre 2014

1 Le problème

On doit affecter des étudiants à des créneaux de surveillance d'une salle. Les étudiants fournissent leurs disponibilités qui sont des plages horaires pour chaque journée.

Objectif : On veut affecter des créneaux aux étudiants de manière à ce que la salle soit toujours surveillée par exactement un étudiant (pas question d'avoir la salle fermée, ou que deux étudiants travaillent en même temps).

Comment vérifier que le problème a une solution ? Si il en a une, laquelle ?

2 Modélisation

On va simplifier le problème :
- on ne considère qu'une seule journée (au lieu des 5 jours de la semaine)
- on arrondit tous les créneaux aux quarts d'heure

Conséquence :
- on numérote les créneaux de 1 à NB_C, si il y a NB_C créneaux. Si la journée va de 8h à 19h, le créneau numéro 1 est 8h-8h15, le créneau numéro 2 est 8h15-8h30, ...
- on numérote les étudiants de 1 à NB_E
- chaque étudiant indique si oui ou il peut faire chaque créneau

Exemple :

Etudiant \ Créneau	1	2	3	4	5
1	X	X	X		X
2				X	
3		X			X

2.1 Première tentative

Paramètres (ce qu'on connaît) :
- est_dispo[e,c]=vrai
  si l'étudiant e est disponible pour le créneau c, faux sinon

Variables (ce qu'on veut calculer) :
– créneau[c] = le numéro de l'étudiant affecté au créneau c
Contraintes à respecter :
– créneau[c] est entre 1 et NB_E
– pour tout creneau c, est_dispo[creneau[c], c] doit être vrai
Sent le roussi pour AMPL

2.2 Deuxième tentative

Paramètres :
– est_dispo[e,c]=vrai
si l'étudiant e est disponible pour le créneau c, faux sinon
Variables :
– est_affecte[e,c] = vrai
si l'étudiant e est affecté au créneau c
Contraintes :
– un seul étudiant par créneau :
pour tout c, est_affecte[e,c] n'est vrai que pour un seul étudiant e
– les créneaux sont compatibles avec les disponibilités des étudiants :
est_affecte[e,c] implique est_dispo[e,c]
C'est réalisable en AMPL. Comment ?

3 Réalisation en AMPL

On va utiliser des variables entières à 0 ou 1, pour représenter les booléens (0=faux, 1=vrai).

3.1 Paramètres et variables

```
param NB_C > 0 ;
param NB_E > 0 ;

set CRENEAUX  = 1 .. NB_C ;
set ETUDIANTS = 1 .. NB_E ;

param est_dispo {ETUDIANTS, CRENEAUX } binary;
/* est_dispo[e,c] =
   - 1 si l'étudiant e est disponible pour le créneau c
   - 0 sinon
*/

var est_affecte {ETUDIANTS, CRENEAUX } binary;
```

```
/* est_affecte[e,c] =
   - 1 si l'étudiant e est affecté au créneau c
   - 0 sinon
*/
```

3.2 Contraintes

Contrainte 1 pour tout c, est_affecte[e,c] n'est vrai que pour un seul étudiant e

On veut ici modéliser un ou exclusif. L'astuce est de considérer la somme sur e de est_affecte[e,c].

Proposition 1 *Soit $x_1, \ldots, x_n$ des entiers égaux à 0 ou 1.*

$$\sum_{i=1}^{n} x_i = 1 \iff \textit{un et un seul des } x_i \textit{ vaut 1}$$

```
/* un et un seul étudiant par créneau */

subject to exactement_un_etudiant_par_creneau {c in CRENEAUX} :
  sum {e in ETUDIANTS} est_affecte[e,c] = 1 ;
```

Contrainte 2 est_affecte[e,c] implique est_dispo[e,c]

Valeurs possibles :

est_affecte[e,c]	est_dispo[e,c]
0	0
0	1
1	1

Proposition 2

$$A \textit{ implique } B \iff A \leq B$$

```
/* créneaux compatibles */

subject to compatibilite_creneau_etudiant {c in CRENEAUX, e in ETUDIANTS} :
  est_affecte[e,c] <= est_dispo[e,c];
```

Objectif Ici on peut mettre n'importe quel objectif, on veut juste trouver une solution.

```
/* objectif */
maximize objectif_evident :
   1 ;
```

Données

```
/* data */

data;
param NB_C = 5;
param NB_E = 3;

param est_dispo :
      1  2  3  4  5 :=
1     1  1  1  0  1
2     0  0  0  1  0
3     0  1  0  0  1;
```

Résolution

```
/* pour résoudre en nombres entiers */
option solver gurobi;
solve;

/* affichage de la solution */
display est_affecte;


/* affichage des créneaux de l'étudiant 2 */
display {c in CRENEAUX} : est_affecte[2,c]

/* affichage des étudiants pour le créneau 1 */
display {e in ETUDIANTS} : est_affecte[e,1]
```

4 Variantes

4.1 Double affectation

On a besoin de 2 étudiants, et non pas un seul pour surveiller la salle.

```
/* exactement 2 étudiants par créneau */

subject to exactement_deux_etudiants_par_creneau {c in CRENEAUX} :
  sum {e in ETUDIANTS} est_affecte[e,c] = 2 ;
```

4.2 Minimiser le nombres d'étudiants

On veut minimiser le nombre d'étudiants à employer.

Il suffit de calculer un entier est_recrute[e], qui vaut 1 si l'étudiant e est utilisé dans un créneau, et 0 sinon. On minimise alors la somme des est_recrute[e].

```
var est_recrute {ETUDIANTS} binary;
/* est_recrute[e] =
   - 1 si l'étudiant e est affecté à au moins un créneau
   - 0 sinon
*/


/* objectif */
minimize nombre_de_recrutes :
   sum {e in ETUDIANTS} est_recrute[e] ;
```

Comment calculer est_recrute[e] ?

```
/* calcul de la variable est_recrute */

subject to contrainte_est_recrute_min {e in ETUDIANTS} :
  est_recrute[e] <= sum {c in CRENEAUX} est_affecte[e,c] ;

subject to contrainte_est_recrute_max {e in ETUDIANTS} :
  sum {c in CRENEAUX} est_affecte[e,c] <= NB_C * est_recrute[e];
```