# Projet ZZ1 - Modélisation

Hugo Bayoud et Antony Vigouret

9 avril 2019

Le but de ce rapport est de **modéliser au mieux le problème que nous avons en entrée. Avant même de s'intéresser à la généralisation puis à la résolution,** posons le problème de façon un peu plus mathématique.

## 1 Mise en place du problème

On considère qu'il y a :

- **n binômes.**
- **m projets** (de telle sorte que m > n).
- **Chaque binôme peut faire i choix différents** et les classer (on considérera tout au long de ce projet que i = 3 comme c'est le cas actuel dans l'affectation des projets). Néanmoins, nous traiterons de l'utilité de changer cette valeur plus loin, dans les autres rapports.
- Un entier P (qui prend des valeurs entières) pour chaque binôme qui permet de savoir, pour chaque projet choisi, quel est le degré d'importance **(poids)** dont sa valeur peut jouer un rôle dans l'affectation.

Le problème est donc :

**Comment affecter aux n binômes, un projet pour éviter les déçus au maximum ; autrement dit, pour laquelle N est le plus petit possible à chaque fois parmi les m projets disponibles.**

# 2 Mise en place d'une modélisation

## 2.1 Théorie :

Nous avons contacté le binôme du projet 8 (projet associé au notre) afin de discuter sur les formats d'entrée et sortie du programme. Malheureusement, ils n'avaient pas encore eu le temps de voir cette partie au moment où nous cherchions cette information.

**Nous pensons néanmoins qu'en entrée, nous pouvons nous attendre à recevoir un tableau de la sorte.** Ce tableau que nous modéliserons ensuite en matrice pour pouvoir l'exploiter dans le programme.

| N° Binôme | Choix projet 1<br>poids p1 | Choix projet 2<br>poids p2 | ··· | Choix projet i<br>poids pi |
|---|---|---|---|---|
| 1 | 20 | 14 | ··· | 2 |
| 2 | 15 | 5 | ··· | 20 |
| ··· | ··· | ··· | ··· | ··· |
| ··· | ··· | ··· | ··· | ··· |

## 2.2 Pratique :

Dans la pratique, nous aurons besoin d'un fichier d'entrée en format *.txt* que le programme de résolution pourra lire et le transformer en matrice.

### 2.2.1 Premier choix :

Dans un premier temps, nous étions partis sur un fichier d'entrée de la sorte :

```
3 5 3

1 0 3 0 2  /*Le binôme 1 souhaite le projet n° 1 en 1er choix,
0 2 0 1 3      le n° 5 en 2ieme choix, le n° 3 en 3ieme.*/
2 0 3 1 0
```

Où,

- La **première ligne** correspond dans l'ordre à n, m, i.
- Les **autres lignes** correspondent chacune à un binôme dont chaque colonne est un projet (0 = projet non choisi).

Mais le problème avec un tel type de fichier d'entrée est :

- On **ne connaît pas directement les poids pi** (importance entre le premier et le deuxième par exemple). De plus, il serait difficile de modifier ces poids.
- On **ne connaît pas les identifiants des groupes.** Chaque ligne correspond à un N' de binôme, mais lequel ?
- Beaucoup **(trop) d'informations inutiles** avec les zéros présents.

### 2.2.2 Deuxième choix :

Mais rapidement, Axel Delsol nous a orienté vers un fichier d'entrée qui palliait à ces problèmes. Désormais notre fichier d'entrée se présente de cette manière, en reprenant le même exemple qu'au dessus :

```
3 5 3
1 2 3

1 1 5 3
2 4 2 5
3 4 1 3
```

Où,

- La **première ligne** correspond toujours à n, m, i.
- Le **deuxième ligne** correspond aux poids. (il y a i colonnes).
- Les **autres lignes correspondent chacune à un binôme**. La première colonne est le n° du binôme, la deuxième colonne, le $1^{er}$ choix du binôme, la troisième colonne, le $2^{ieme}$ choix, etc...

# 3 Contraintes

## 3.1 Matrice C :

A partir du fichier d'entrée ; nous créons une **structure de données (écriture en C) pour permettre de construire une matrice qui sera utilisée pour la résolution** avec Gurobi.

Une matrice (notée C) à n lignes et m colonnes (**chaque coefficient de C correspond à un couple [N binôme ; Nprojet]**) qui contient un entier N (compris entre 0 et i) qui correspond au n° du projet choisi pour un binôme donné.
(ex : si chaque binôme peut faire 3 choix (i = 3), alors pour chaque ligne de C, il y aura un 1, un 2 et un 3, le reste : 0)

Il apparaît donc **plusieurs contraintes sur cette matrice**,

- La **somme de chaque ligne de C est égale à la somme des N premiers entiers.** (ex : pour i = 3, la somme de chaque ligne devra faire exactement 1+2+3 = 6).
- La **somme de chaque colonne doit faire entre 0** (aucun des n binômes ne choisit ce projet) **et i*n** (tous les binômes souhaitent le même projet comme dernier choix par exemple).

## 3.2 Matrice X :

### 3.2.1 Théorie/Contraintes :

**Le format de sortie des données par Gurobi peut être une matrice (notée X)** à n lignes et m colonnes qui contient l'affectation des projets à chaque binôme. Elle contient des 1 ou 0 en lien avec la matrice C.

- Il ne faut pas qu'un binôme ait 2 projets ou plus ni qu'un projet soit affecté à 2 binômes ou plus. Dans la matrice C, **il ne faut donc pas qu'il y ait plus d'un coefficient différent de 1 par ligne et par colonne pour qu'elle soit valide.**
- Néanmoins, il faut que chaque binôme ait un projet. **Il faut donc qu'il y est au moins un coefficient à 1 par ligne.**

| Le but est de **minimiser la somme des coefficients de la matrice X pondérés par le coefficient équivalent de la matrice C**. |
|---|

### 3.2.2 Sortie de l'algorithme/Présentation du résultat :

Nous présenterons un tableau bien plus compacte et lisible que la matrice X qui **évite les informations inutiles** (L'ensemble des zéros).
Nous aurons un tableau de la sorte :

| N° du binôme | N° du projet affecté |
|---|---|
| ... | ... |
| ... | ... |
| ... | ... |

# 4 Exemple d'un cas simple n = 3

Soit,

$$\mathrm{C} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 3 & 2 & 1 \end{pmatrix} \text{ et } \mathrm{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{pmatrix}$$

Avec les contraintes :

$$\begin{cases} x_{1,1} + x_{1,2} + x_{1,3} = 1 \\ x_{2,1} + x_{2,2} + x_{2,3} = 1 \\ x_{3,1} + x_{3,2} + x_{3,3} = 1 \end{cases}$$

et

$$\begin{cases} x_{1,1} + x_{1,2} + x_{3,1} \leqslant 1 \\ x_{1,2} + x_{2,2} + x_{3,2} \leqslant 1 \\ x_{1,3} + x_{2,3} + x_{3,3} \leqslant 1 \end{cases}$$

Et on veut minimiser :

$$\lambda = min(1.x_{1,1} + 2.x_{1,2} + 3.x_{1,3} + 1.x_{2,1} + 3.x_{2,2} + 2.x_{2,3} + 3.x_{3,1} + 2.x_{3,2} + 1.x_{3,3})$$