

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Formations par Alternance / Informatique industrielle par apprentissage

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 11 26

[polytech.univ-tours.fr](http://polytech.univ-tours.fr)

## Rapport de projet développement DII3

2017-2018

# Outil d'aide à l'affectation des projets

**Tuteur académique**

Ameur SOUKHAL

**Étudiants**

Jérémy LOCHE (DII3)

Louis THOMAS (DII3)

21 mars 2018



# Liste des intervenants

Nom	Email	Qualité
Jérémy LOCHE	<a href="mailto:jeremy.loche@etu.univ-tours.fr">jeremy.loche@etu.univ-tours.fr</a>	Étudiant DII3
Louis THOMAS	<a href="mailto:louis.thomas@etu.univ-tours.fr">louis.thomas@etu.univ-tours.fr</a>	Étudiant DII3
Ameur SOUKHAL	<a href="mailto:ameur.soukhal@univ-tours.fr">ameur.soukhal@univ-tours.fr</a>	Tuteur académique, Informatique industrielle par apprentissage



# Avertissement

Ce document a été rédigé par Jérémy Loche et Louis Thomas susnommés les auteurs.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Ameer Soukhal susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

Les auteurs reconnaissent assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

Les auteurs attestent que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

Les auteurs attestent ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

Les auteurs attestent que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

Les auteurs reconnaissent qu'ils ne peuvent diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

Les auteurs autorisent l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



## Pour citer ce document

Jérémy Loche et Louis Thomas, *Outil d'aide à l'affectation des projets*, Rapport de projet développement DII3, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2017-2018.

```
@mastersthesis{
  author={Loche, Jérémy and Thomas, Louis},
  title={Outil d'aide à l'affectation des projets},
  type={Rapport de projet développement DII3},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2017-2018}
}
```

# Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	iii
Liste des tableaux	iv
Liste des Algorithmes	v
Introduction	1
<b>1 Le problème d'affectation</b>	<b>2</b>
1 Présentation .....	2
2 La modélisation mathématique.....	2
2.1 Les données et les paramètres .....	2
2.2 Les variables et les contraintes .....	3
2.3 Objectif simple, maximiser la préférence moyenne.....	4
2.4 Objectif de compromis, la contrainte de la préférence minimale .....	5
<b>2 Organisation et choix techniques de réalisation de l'outils d'affectation</b>	<b>8</b>
1 Résoudre le problème d'affectation .....	8
1.1 L'algorithme Hongrois, de Kuhn-Munkres .....	8
1.2 Résolution par un Solver Linéaire.....	9

2	Implémentation de la méthode de résolution.....	9
3	Saisie et visualisation.....	10
4	Organisation Front et Back end .....	10



# Table des figures

<b>2</b>	<b>Organisation et choix techniques de réalisation de l'outils d'affectation</b>	
1	Modélisation étudiants-projets dans un graphe biparti complet.....	9
2	Couplages parfait sur le graphe biparti (fig. 1 (Chapitre 2)) .....	9



# Liste des tableaux

## 1 Le problème d'affectation

- |   |   |   |
|---|---|---|
| 1 | Indiçage des entités du modèle mathématique ..... | 3 |
| 2 | Données de préférence pour l'exemple.....         | 5 |





# Liste des Algorithmes

1	Résolution du modèle avec <i>ValeurPrefMin</i> choisi par tentative .....	6
2	Recherche de <i>ValeurPrefMin</i> dichotomique .....	7

Report\_LOCHE\_THOMAS\_3A\_DII12.loa



# Introduction

Tous les ans, dans toutes les promotions d'étudiants ingénieurs de Polytech Tours, d'intenses débats ont lieu au moment des affectations de projets. En effet, chacun souhaite avoir le projet qui lui plaît le plus. Pour aider à dénouer ces débats, nous avons proposé de travailler sur un outil d'aide à l'affectation des projets étudiants.

A travers ce projet, nous verrons comment nous avons choisi de résoudre problème d'affectation afin que chacun soit satisfait de son affectation.

Vous découvrirez comment nous avons réparti notre charge de travail dans ce développement.

Nous aborderons alors comment nous avons modélisé ce problème puis avec quels outils nous avons travaillé pour arriver à un outils de saisie et de résolution des affectations.

Pendant la lecture de ce rapport de projet, nous vous exposerons une méthode de résolution du problème d'affectation.

# 1

## Le problème d'affectation

### 1 Présentation

Ce travail est un problème d'affectation. En effet, le but est d'associer à un ou plusieurs individus un projet précis en respectant un objectif simple : maximiser la satisfaction générale de l'affectation. L'idée est que chacun soit satisfait du projet auquel il a été affecté et qu'il soit en mesure de le mener à bien dans les meilleures conditions.

### 2 La modélisation mathématique

Pour résoudre ce problème, nous allons avoir besoin de le formaliser de manière mathématique afin d'essayer de trouver des solutions. Nous rappelons que le but est d'affecter un étudiant, un binôme ou un groupe de personnes à un projet.

Le fait de formaliser un problème en langage mathématique nous permettra de demander à un **solver** de trouver la solution à notre problème.

Pour cela, il faut définir quels sont les **paramètres**<sup>1</sup>, les **variables**<sup>2</sup>, les **contraintes**<sup>3</sup>, et la **fonction objectif**<sup>4</sup> de notre problème.

Commençons par définir nos données c'est à dire les paramètres de notre modèle.

#### 2.1 Les données et les paramètres

Pour cela on commence par créer deux entités appartenant à deux ensembles qui seront nos données de départ :

1. une entité *individu*  $\in$  *Individus* : représente une personne, un binôme ou un groupe de personnes qui accomplira un **projet** ;
2. une entité *projet*  $\in$  *Projets* : représente un projet auquel sera affecté un **individu**.

Pour plus de simplicité, on va indexer les entités comme dans le tableau 1. On a choisi que les binômes seront représentés par des individus.

- 
1. Dans un modèle mathématique, un paramètre est une donnée.
  2. Dans un modèle mathématique, une variable est une valeur calculée, une inconnue.
  3. Les contraintes représentent les règles à respecter lors de la résolution du problème.
  4. C'est l'ultime but à atteindre pour établir que la solution au problème est optimale.

Table 1 – Indichage des entités du modèle mathématique

$\forall \text{individu} \in \text{Individus}$		$\forall \text{projet} \in \text{Projets}$	
Indice Individu	Nom individu	Indice Projet	Nom projet
1	Binôme : Arthur, Léo	1	Projet : Lampe connecté
2	Binôme : Sophie, Jean	2	Projet : Moniteur UVA
...	...	...	...
nbIndividus	Binôme : Paul, Pierre	nbProjets	Projet : Voiture RC

Pour que l'affectation de chaque individu à un projet, il faut assez de projet pour tout le monde ce qui implique la relation suivante :

$$nbProjets \geq nbIndividus$$

Pour tout les individus, on souhaite connaître sa préférence pour un projet afin de procéder à l'affectation. En utilisant le formalisme mathématique, on peut définir la quantité *preference* qui définira l'appréciation d'un individu à un projet donné :

$$\forall (i, p) \in \text{Individus} \times \text{Projets}, preference[i, p] \in \llbracket 1, nbProjets \rrbracket$$

On choisit de dire que cette préférence va de *nbProjet* à 1 dans l'ordre décroissant d'appréciation. Ainsi, un individu qui aurait la préférence *nbProjets* pour un projet indiquera que ce projet est son préféré et 1 pour celui qui lui plait le moins.

Pour que l'affectation soit équitable, un individu ne peut pas donner la même préférence à deux projets différents. De manière mathématique cela donne :

$$\forall (i, p1, p2) \in \text{Individus} \times \text{Projets}^2, p1 \neq p2 \Rightarrow preference[i, p1] \neq preference[i, p2]$$

Lorsque ces pré-conditions sont respectés, alors on crée un **paramètre** appelé préférence utile pour résoudre le problème d'affectation.

$$\forall (i, p) \in \text{Individus} \times \text{Projets}, preference[i, p]$$

Nous allons maintenant voir quels contraintes et variables sont à déterminer pour résoudre le problème.

## 2.2 Les variables et les contraintes

Pour modéliser entièrement le problème d'affectation, la préférence de chaque individus pour un projet n'est pas suffisante. Nous allons devoir mettre en place des **variables** et des **contraintes** qu'elles doivent respecter.

Une **variable** est une valeur qui doit être déterminée pour résoudre le problème. Il faut donc la calculer.

Nous cherchons à déterminer a quel projet est affecté un individu. Pour cela on crée la variable *affectation* :

$$\forall (i, p) \in \text{Individu} \times \text{Projets}, affectation[i, p] = \begin{cases} 1 & \text{si l'individu } i \text{ est affecté au projet } p \\ 0 & \text{sinon} \end{cases}$$

On connaît la quantité que l'on souhaite déterminer pour résoudre le problème d'affectation, cependant, il faut définir des **contraintes** qui vont définir les règles de l'affectation.

Nous allons énoncer et définir deux contraintes utiles à la résolution du problème. La première est le nombre de projets par individus et la seconde est le nombre d'individus par projets. Pour une affectation d'un individu pour un projet, alors on a les contraintes (1) et (2).

$$\forall i \in \text{Individus}, nbProjetsParIndividu = \sum_{p \in Projets} affectation[i, p] = 1 \quad (1)$$

Pour notre problème, un individu doit être affecté à un seul projet, ce qui donne la contrainte (1).

$$\forall p \in Projets, nbIndividusParProjet = \sum_{i \in Individus} affectation[i, p] \leq 1 \quad (2)$$

La contrainte (2) dit qu'un projet ne peut être réalisé que par au plus un individu.

Nous avons défini nos paramètres, variables et contraintes, donc il ne reste plus qu'à définir l'objectif à atteindre avec tout ça.

### 2.3 Objectif simple, maximiser la préférence moyenne

Pour choisir quelle configuration de variable on va garder, c'est à dire quel choix d'affectation on va faire, on va devoir mettre en place un indicateur permettant de juger de la qualité de l'affectation. On appelle cet indicateur la **fonction objectif**. Trouver la solution à un problème mathématique se résume souvent à établir la bonne fonction objectif qu'on souhaite faire converger vers une valeur donnée, maximiser ou minimiser.

Dans notre cas, on souhaite maximiser la satisfaction générale, c'est à dire maximiser la somme des préférences des projets affectés à chacun. Ainsi le résultat optimal sera atteint lorsqu'on aura réussi à trouver la combinaison de variable qui donne la plus grande valeur de la **fonction objectif**.

$$\text{maximiser} \left( \sum_{(i \in Individus)} \sum_{(p \in Projets)} affectation[i, p] \times preference[i, p] \right) \quad (3)$$

Pour le problème d'affectation maximiser la fonction objectif (3) donnera la satisfaction générale la plus élevée car elle correspondra à faire en sorte que si un individu est affecté à un projet, la valeur de la préférence sera en moyenne la plus grande.

Cette méthode fonctionne est efficace pour trouver la meilleure préférence moyenne, cependant, elle peut être injuste et sélectionner pour la majorité des individus la préférence maximale et pour un individu la pire des préférences.

Il peut donc apparaître un sentiment d'injustice derrière cette affectation avec la sensation que quelqu'un a été mis de côté pour satisfaire le bien commun.

A cela, nous proposons une solution de compromis qui fournit une solution plus homogène (si elle existe évidemment).

## 2.4 Objectif de compromis, la contrainte de la préférence minimale

Pour obtenir une affectation plus homogène, on introduit une contrainte au modèle : c'est la contrainte de la préférence minimale. Le but est d'assurer que lorsqu'un individu est affecté à un projet, la préférence soit supérieure ou égale à une valeur de seuil *ValeurPrefMin*. On peut donc assurer avec cette contrainte que la préférence n'atteindra jamais 1 ou 2 etc...

On pose donc un nouveau **paramètre** :

$$ValeurPrefMin \in \llbracket 1, nbProjet \rrbracket$$

Et une nouvelle **contrainte**, qui impose que l'affectation soit faite avec une préférence supérieure à *ValeurPrefMin* :

$$\forall i \in Individus, preferenceMinimale = \sum_{p \in Projets} affectation[i, p] \times preference[i, p] \geqslant ValeurPrefMin$$

### Problème

Cette contrainte est très restrictive sur les solutions. Il se peut même *ValeurPrefMin* de manière trop audacieuse et que le problème soit insoluble avec ces paramètres. Prenons un exemple de l'affectation de 4 individus et 4 projets (Cf. tableau :2).

Table 2 – Données de préférence pour l'exemple

préférence[individu,projet]	Projet 1	Projet 2	Projet 3	Projet 4
Individu 1	4	3	2	1
Individu 2	4	3	1	2
Individu 3	3	4	2	1
Individu 4	2	4	1	3

Si on choisit *ValeurPrefMin* = 3, alors le modèle mathématique est insoluble car la contrainte *preferenceMinimale* ne pourra pas être respectée pour tout les individus. On a 4 projets pour 4 individus, on remarque qu'il faudra donc exactement 1 projet par individu pour respecter la contrainte *nbProjetsParIndividu* = 1. Le problème est que si un individu est affecté au projet P3, alors la valeur de la préférence est inférieure ou égale à 2 donc inférieure à *ValeurPrefMin* = 3. On ne respecte alors pas la contrainte *preferenceMinimale* pour un des individu donc le modèle est insoluble. Pour trouver une solution il faudra revoir le paramètre *ValeurPrefMin* à la baisse. Pour la valeur 2, on commencera à trouver des solutions pour laquelle la *ValeurPrefMin* est maximale. Donc les bonnes solutions de compromis.

Il y a plusieurs solutions pour automatiser la détermination de la valeur de *ValeurPrefMin* pour laquelle on assure que tout le monde aura la meilleur préférence possible tout en maximisant la satisfaction générale.

### Solution 1

Le but est d'essayer de trouver la valeur maximale de *ValeurPrefMin* pour laquelle on arrive à résoudre le modèle. Pour cela, on peut rechercher la valeur maximale en faisant plusieurs résolutions du modèle mathématique en fixant différentes valeurs de *ValeurPrefMin*. On peut faire une recherche par tentative ou dichotomique. *ValeurPrefMin* n'intervient pas dans la fonction objectif à ce stade.

*Recherche de ValeurPrefMin par tentative*

La fonction *SolveModel*(param : *ValeurPrefMin*) est un sous-algorithme appelé par notre algorithme pour effectuer la résolution du modèle mathématique avec la valeur de préférence minimale passée en paramètre. Elle renvoie une structure *ResultSolver* contenant les résultats de la résolution *ResultSolver.data* et l'état de la résolution *ResultSolver.state* qui peut valoir soit *Success* si une solution au modèle a été trouvée et *Fail* sinon.

On l'utilise donc pour savoir si on a réussi à résoudre le

```

/* Init variable résultat du solver, contient ResultSolver.data et
   ResultSolver.state */
ResultSolver ← Null
/* Commencer par la valeur la plus restrictive jusqu'à la moins
   restrictive */
for ValeurPrefMin = nbProjets to 1 do
  /* Effectuer une résolution avec la valeur de préférence minimale */
  ResultSolver ← SolveModel(ValeurPrefMin)
  if ResultSolver.state = Success then
    /* Le solver a trouvé une solution */
    /* on a trouvé le maximum pour ValeurPrefMin */
    /* on arrête la boucle ici */
    break
  end
end
/* Retourner le résultat du solver qui est donc la solution avec la
   ValeurPrefMin maximale et la plus grande satisfaction */
return ResultSolver

```

**Algorithme 1 :** Résolution du modèle avec *ValeurPrefMin* choisi par tentative

Cette méthode n'a pas la meilleure complexité et peut considérablement allonger les temps de résolution du modèle. C'est pour cela que nous proposons la méthode dichotomique.

*Recherche de ValeurPrefMin dichotomique***Solution 2**

On change la nature de *ValeurPrefMin* et elle devient donc une **variable** intervenant dans la fonction objectif. L'idée derrière cette modification est de demander, en plus de la maximisation de la préférence moyenne, une maximisation de la *ValeurPrefMin*.

*Méthode par aggrégation*

On peut rajouter *ValeurPrefMin* dans la fonction objectif en faisant une simple somme (Cf. (4))

$$\text{maximiser} \left( \text{ValeurPrefMin} + \sum_{(i \in \text{Individus})} \sum_{(p \in \text{Projets})} \text{affectation}[i, p] \times \text{preference}[i, p] \right) \quad (4)$$

Cette agrégation permettra donc de chercher aussi à maximiser la *ValeurPrefMin*. Cependant, elle ne le sera pas au même titre que la satisfaction moyenne car il peut être plus avantageux de chercher à maximiser d'avantage la satisfaction que la *ValeurPrefMin*.

```

/* Init variable résultat du solver, contient ResultSolver.data et
   ResultSolver.state                                     */
ResultSolverFinal ← Null
deb ← 1
fin ← nbProjet
mil ←  $\frac{deb+fin}{2}$ 
while deb < fin do
  ResultSolverTemp = SolverModel(mil)
  if ResultSolverTemp.state = Success then
    deb = mil + 1
    ResultSolverFinal = ResultSolverTemp
    mil ←  $\frac{deb+fin}{2}$ 
  end
  else
    fin = mil - 1
    mil ←  $\frac{deb+fin}{2}$ 
  end
end
return ResultSolverFinal

```

**Algorithme 2 :** Recherche de *ValeurPrefMin* dichotomique**Méthode de combinaison linéaire des variables**

Pour cela, on peut ajouter des poids  $\alpha, \beta$  deux nombres donnés qui peuvent servir à donner plus d'importance à la *ValeurPrefMin* ou à la satisfaction générale (Cf. (5)).

$$\text{maximiser} \left( \alpha \text{ValeurPrefMin} + \beta \sum_{(i \in \text{Individus})} \sum_{(p \in \text{Projets})} \text{affectation}[i, p] \times \text{preference}[i, p] \right) \quad (5)$$

**Transition :**

Après avoir vu le fond du problème que nous avons souhaité résoudre dans ce projet, nous allons passer à notre proposition de réalisation d'un outil d'aide à l'affectation des projets.



# 2

## Organisation et choix techniques de réalisation de l'outil d'affectation

### 1 Résoudre le problème d'affectation

Il existe plusieurs manières de résoudre le problème d'affectation. Nous rappelons que le but est d'attribuer pour chaque étudiant un unique projet en maximisant la satisfaction générale.

Dans un premier temps, nous avons recherché des algorithmes permettant de résoudre ce problème. Nous verrons 2 méthodes, l'une utilisant un algorithme Hongrois et l'autre utilisant un solveur linéaire.

#### 1.1 L'algorithme Hongrois, de Kuhn-Munkres

L'algorithme Hongrois permet de trouver un couplage parfait dans un graphe biparti de poids maximum (ou minimum) d'après Wikipédia.

**Pourquoi est-ce un algorithme adapté au problème d'affectation ?**

*Un graphe biparti ? Pourquoi ?*

Un graphe biparti un type particulier de graphe dont on distingue 2 nuages de noeuds dont les noeuds appartenant à un nuage ne peuvent être liés qu'aux noeuds de l'autre nuage par des arcs. Autrement dit, il est interdit d'avoir un arc entre 2 noeuds d'un même nuage.

On a une analogie directe entre les 2 nuages de noeuds du graphe, le premier nuage représente les étudiants et le second représente les projets.

Chaque étudiant est lié à chaque projet par  $n$  arcs pour les  $n$  projets. Par exemple dans la figure 1, on dispose de 3 étudiants ayant émis une préférence pour chaque projet. La valeur de la préférence correspond au poids sur l'arc reliant l'étudiant au projet.

*Un couplage parfait ? À quoi ça correspond ?*

Un couplage parfait dans un graphe correspond à choisir des arcs dans ce graphe de façon à ce qu'il n'existe qu'un et un seul arc entre chaque noeuds. Dans notre cas, si on réalise un couplage parfait sur le graphe biparti, alors on aura réalisé une affectation des étudiants à un projet.

La figure 2 illustre ces propos en mettant en évidence quelques couplages possibles à partir du graphe biparti 1.

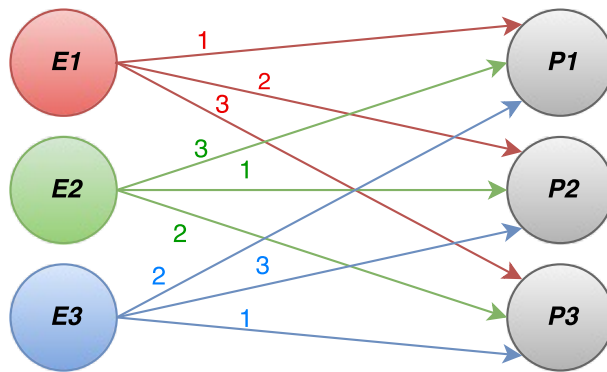


Figure 1 – Modélisation étudiants-projets dans un graphe biparti complet

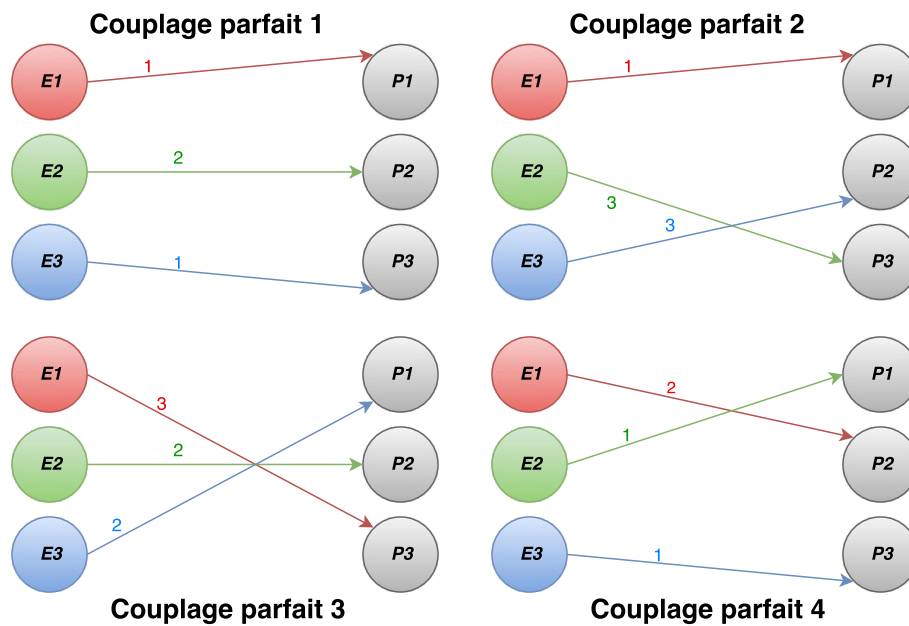


Figure 2 – Couplages parfaits sur le graphe biparti (fig. 1)

## 1.2 Résolution par un Solver Linéaire

## 2 Implémentation de la méthode de résolution

Pour mettre en application notre modèle mathématique, nous avons choisi d'utiliser le solveur GLPK et d'écrire notre modèle en GMPL aussi connu sous le nom de Mathprog.

En effet, nous avons précédemment formalisé le modèle du problème d'affectation et il ne reste plus qu'à le traduire en langage informatique pour qu'un solveur trouve une solution optimisée qui respecte toutes les contraintes.

Un solveur est une machine à calculer équipée d'algorithmes d'optimisation permettant de trouver la solution à un modèle mathématique qui maximise (minimise, ou se rapproche d'une valeur donnée) au mieux la fonction objectif.

Nous avons choisi GLPK car c'est un solveur libre multi-plateforme qu'il est possible d'intégrer dans tout type d'applications à travers des bibliothèques.

Tel que nous l'avons défini, le problème d'affectation est un problème d'optimisation linéaire.

Ce qui peut rendre la résolution du problème compliqué est le fait que les variables de décision, dans notre cas les *affectations* doivent être des nombres entiers. En effet, un étudiant ne peut pas être affecté à 0,5 projet.

Le problème d'affectation peut être résolu par GLPK car il s'agit d'un problème linéaire. Les données à déterminer ne sont que des combinaisons linéaires les unes des autres. Comme il s'agit d'un problème dont les solutions sont des nombres entiers, nous avons utilisé la programmation linéaire en nombre entiers (PLNE) pour le résoudre.

Alors il existe des méthodes de résolution efficaces de résolution.

Une façon naïve de résoudre le problème serait de lister toutes les valeurs possibles pour les variables à déterminer et de les tester une par une. Alors il suffira de voir si une valeur donnée respecte toutes les contraintes et obtient une meilleur note à la fonction objectif. Si c'est le cas, alors on gardera cette solution jusqu'à en trouver une meilleure où lorsque l'on aura testé toutes les possibilités.

Lors de ce projet, nous n'avons pas fait le tour de toutes les méthodes d'optimisations utilisées par les solver, nous avons surtout cherché à comprendre comment utiliser le solver et le langage GMPL pour l'utiliser à la résolution de notre problème.

Cependant, pour dire quelques mots sur les méthodes d'optimisations, le solver est capable de limiter le nombre d'itérations et le nombre de tests à faire pour trouver une solution optimisée problème.

En effet, en fonction du jeu de contraintes que l'on fournit dans le modèle mathématique, le solver va appliquer un algorithme discriminant certaines valeurs des variables de décision. Par exemple, dans notre cas, la contrainte imposant qu'un étudiant ne doit être affecté qu'à un seul projet permet au solver d'éliminer les solutions où la matrice d'affectations aurait sur une ligne plusieurs 1 (ce qui signifie qu'un étudiant serait affecté à plusieurs projets). On peut éliminer ainsi un bon nombre de valeurs et n'en tester qu'une portion restreinte.

Passons désormais à notre implémentation GMPL du problème d'affectation.

### 3 Saisie et visualisation

### 4 Organisation Front et Back end

# Outil d'aide à l'affectation des projets

## Résumé

Ce projet a pour but de fournir un outil d'aide a l'affectation des projets.

## Mots-clés

affectation, solver, modèle, mathématique, google, sheet

## Abstract

This is an example of an abstract for the report.

## Keywords

example,  $\text{\LaTeX}$

**Tuteur académique**  
Ameur SOUKHAL

**Étudiants**  
Jérémy LOCHE (DII3)  
Louis THOMAS (DII3)