SDP Modélisation

Alexandre Sajus, Antoine Naccache, Eliott Barbot

Février 2023

1 Introduction

L'objectif de cette modélisation est de réaliser un planning de travail en optimisant certains objectifs à partir d'un jeu de données contenant une liste de travailleurs avec des compétences et des jours de vacances; et une liste de projets ayant chacun avec un gain associé, des exigences de réalisations et des pénalités de retard.

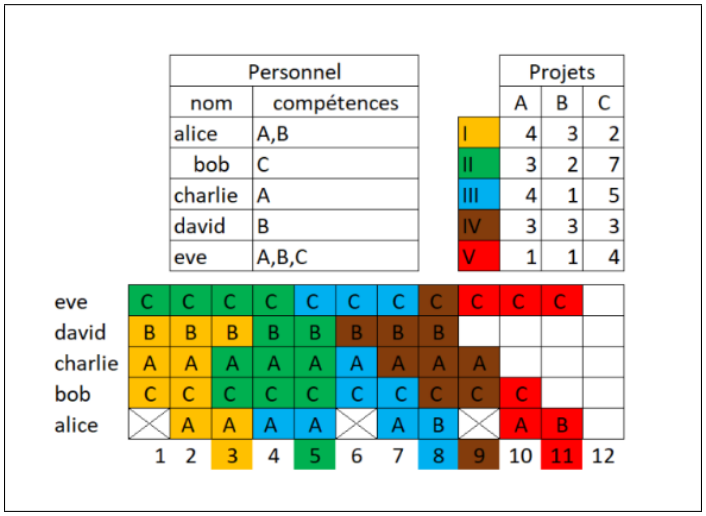

Figure 1: Exemple de Planning

2 Problème

Le problème nous donne les constantes suivantes:

- **horizon**: l'horizon de temps de la modélisation en jours
- **staff**: liste contenant chaque personne du staff, chaque élément est une liste contenant:
  - **qual**: une liste de qualifications en string
  - **vacs**: un liste de jours en int où le worker est en congé
- **jobs**: liste contenant chaque job, chaque élément est une liste contenant:
  - **gain**: le bénéfice en int
  - **due date**: la date maximale de rendu du projet
  - **working days per qual**: dictionnaire contenant le nombre de jours de travail en int par qualification en string

On ajoute les fonctions utilitaires suivantes, qui transcrivent les données du problème sous une forme plus utilisable pour formaliser le problème :

- **conge(worker, day)** est une constante binaire qui vaut 1 si worker est en congé le jour day

- **qualifie(worker, qual)** est une constante binaire qui vaut 1 si worker possède la compétence qual
- **requirements(job, qual)** est une constante entière qui représente le nombre de jours de travail requis pour la qualification qual pour le projet job
- **gain(job)** est une constante entière qui représente le gain associé au job
- **penalite(job)** est une constante qui représente la pénalité de retard journalière associée au job
- **itstoolate(job, day)** est une constante binaire qui vaut 0 si day est inférieur à la due date de job, 1 sinon

Ces fonctions utilitaires, ainsi que les variables de décision présentées à la section suivante, ont pour paramètres :

- $worker \in list\_workers$ qui représente un élément de l'ensemble des collaborateurs
- $day \in list\_days$ qui représente un élément de l'ensemble des jours
- $qual \in list\_quals$ qui représente un élément de l'ensemble des qualifications
- $job \in list\_jobs$ qui représente un élément de l'ensemble des projets

2.1 Variables de décision

Les variables de décision du problème représentent le choix des jobs et le planning:

- **chosenjob(job)** est une variable binaire qui vaut 1 si job est fait dans la modélisation, 0 sinon
- **planning(worker, qual, day, job)** est une variable binaire qui vaut 1 si worker travaille à day sur le job sur la qualification qual, 0 sinon

On définit également des variables auxiliaires qui nous servent à définir les fonctions objectifs :

- **workedtoday(job, day)** est une variable binaire qui vaut 1 si on travaille sur le projet job le jour day, et 0 sinon
- **estaffecte(worker, job)** est une variable binaire qui vaut 1 si l'employé $worker$ travaille sur le projet job au moins une fois
- **nbmaxjobs** est une variable entière qui représente le nombre maximum de projets auxquels un même collaborateur peut être affecté (i.e. chaque $worker$ travaille au maximum sur $nbmaxjobs$ projets)
- **end(job)** est une variable entière qui correspond à la date de fin d'un job
- **begin(job)** est une variable entière qui correspond à la date de début d'un job
- **maxlenjob** est une variable entière qui vaut la durée maximale entre le début et la fin d'un projet

2.2 Contraintes

La liste des contraintes est la suivante:

1. Contrainte de qualification : Un membre du personnel ne peut être affecté à une qualification d'un projet que s'il possède cette qualification

$$\forall worker, \forall day, \forall qual, \forall job, \quad planning(worker, qual, day, job) \leq qualifie(worker, qual)$$

2. Contrainte d'unicité d'affectation : un travailleur ne peut pas travailler le même jour sur deux jobs ou qualifications différentes:

$$\forall worker, \forall day, \sum_{qual,job} planning(worker, qual, day, job) \leq 1$$

3. Contrainte de congé : Un membre de personnel ne peut pas être affecté à une qualification de projet un jour de congé

$$\forall (worker, qual, day, job), \quad planning(worker, qual, day, job) + conge(worker, day) \leq 1$$

4. Si un job est choisi, il doit être complété:

$$\forall job, \forall qual,$$

$$\sum_{worker,day} planning(worker, qual, day, job) \geq requirements(job, qual) - M(1 - chosenjob(job))$$

(avec M une constante suffisamment grande)

On a de plus des contraintes liées à la définition de variables auxiliaires :

5. La variable binaire estaffecte(worker, job) vaut 1 si l'employé $worker$ travaille sur le projet job au moins une fois :

$$\forall (worker, job, day, qual), \quad estaffecte(worker, job) \geq planning(worker, qual, day, job)$$

6. Chaque $worker$ travaille au maximum sur $nbmaxjobs$ projets :

$$\forall worker, \sum_{job} estaffecte(worker, job) \leq nbmaxjobs$$

7. la variable binaire $workedtoday(job, day)$ vaut 1 si on travaille sur le projet job le jour day, et 0 sinon

$$\forall (worker, qual, day, job), \quad planning(worker, qual, day, job) \leq workedtoday(job, day)$$

8. La variable entière $maxlenjob$ vaut la durée maximale entre le début et la fin d'un projet :

$$\forall job, end(job) - begin(job) \leq maxlenjob$$

9. La variable entière $begin(job)$ correspond à la date de début d'un job :

$$\forall (day, job), \quad day - begin(job) \geq M * (workedtoday(job, day) - 1)$$

(avec M une constante suffisamment grande)

10. La variable entière $end(job)$ correspond à la date de fin d'un job :

$$\forall (day, job), \quad end(job) - day \geq M * (workedtoday(job, day) - 1)$$

(avec M une constante suffisamment grande)

2.3 Fonctions objectifs

Les fonctions objectifs sont:

1. La somme des gains sur les jobs choisis :

$$max \sum_{job} \left(gain(job) * chosenjob(job) - \sum_{day} (itstoolate(job, day) * workedtoday(job, day) * penalite(job)) \right)$$

2. La minimisation du nombre de jobs par worker:

$$min(nbmaxjobs)$$

3. La minimisation du nombre du jour sur lequel va s'étaler le projet le plus long.

$$min(maxlenjob)$$

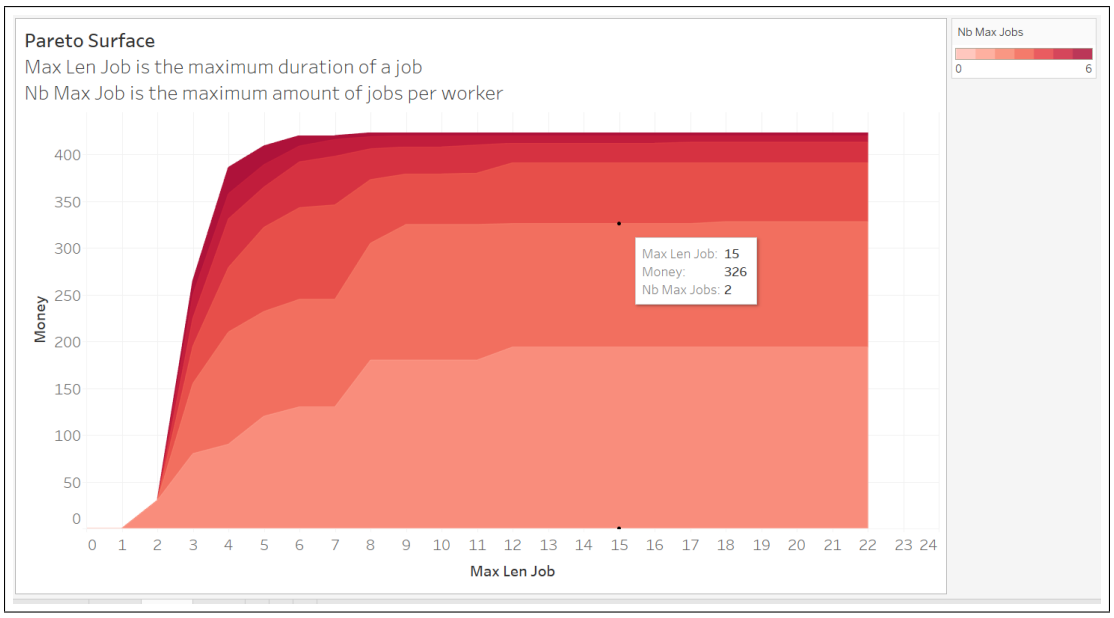

Figure 2: Surface de Pareto sur la medium instance

3 Surface de Pareto

A partir de cette modélisation, pour la medium instance on obtient la surface des solutions non-dominées présentée à la Figure 2.

On peut choisir un planning à partir de cette surface. Par exemple si on veut que les travailleurs travaillent au maximum sur 2 projets et que les projets durent au maximum 15 jours, pour s'assurer que les travailleurs restent concentrés sur les projets, on peut au maximum obtenir 326 en gain monétaire, comme on le voit dans la Figure 3.

On peut aussi choisir en fonction de la somme pondérée d'objectifs. Par exemple, sur la large instance, en changeant les poids dans une pondération entre l'objectif argent gagné et l'objectif nombre de projet maximum par travailleur, on obtient les plannings de la Figure 4.

On observe ici, que si on optimise uniquement en fonction de l'argent gagné, les travailleurs doivent travailler sur plus de 5 projets sur un mois (les couleurs par ligne représentent les différents projets) ce qui peut être un problème de concentration pour les travailleurs. Lorsqu'on ajoute le nombre de projets par travailleur dans l'objectif, les couleurs sur une ligne deviennent plus uniformes ce qui montre que les travailleurs travaillent sur moins de projets. Mais cela nous contraint à prendre moins de projets ce qui réduit le gain monétaire qui passe de 833 à 290 si on ne prend qu'un seul projet.

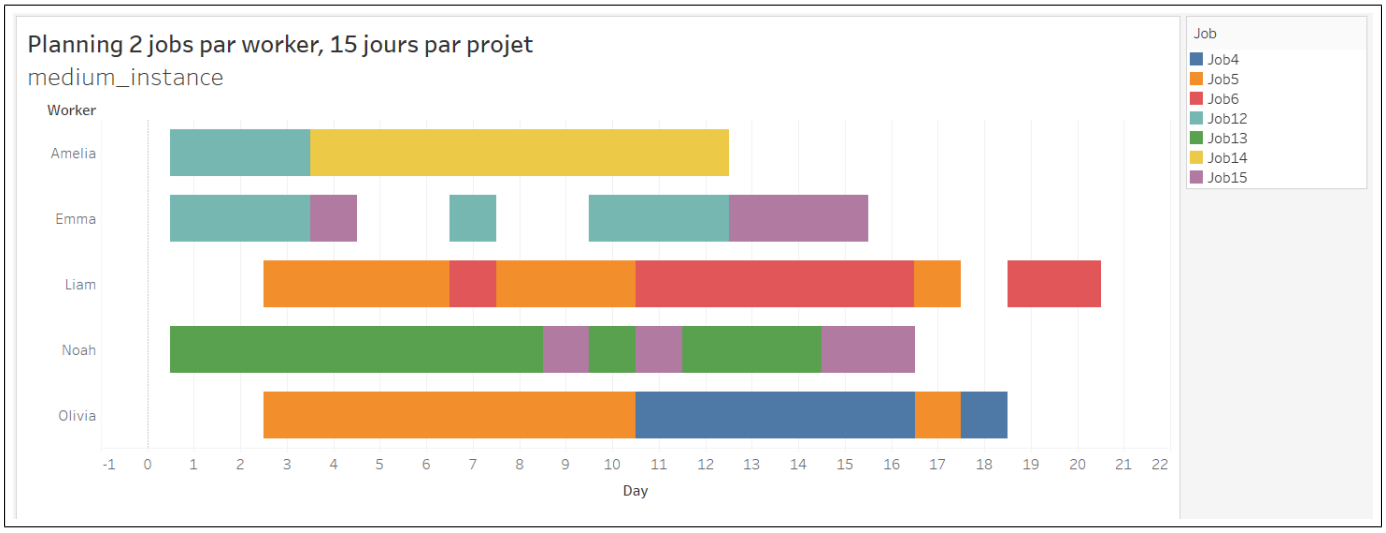

Figure 3: Planning 2 jobs par worker, 15 jours par projet sur medium instance

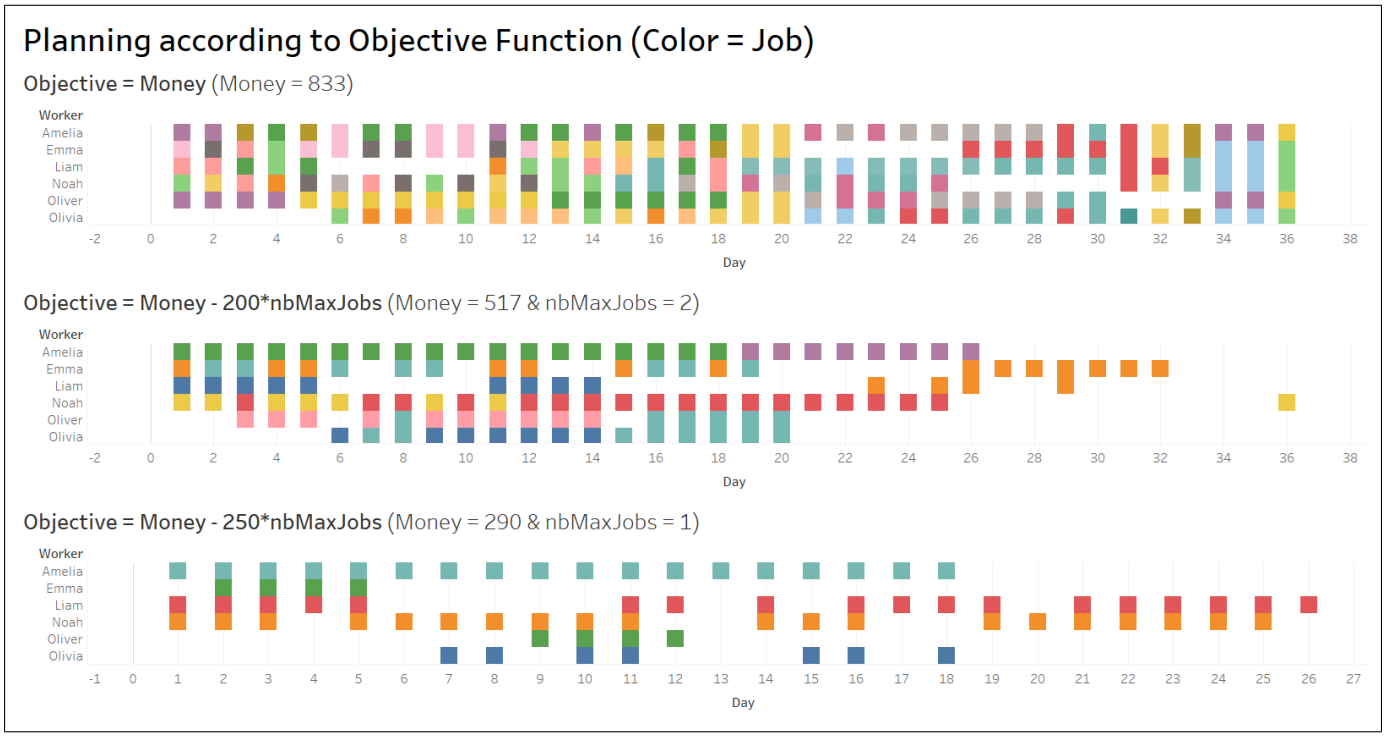

Figure 4: Plannings compromis entre argent et nombre de projets par travailleur sur la large instance