Projet 7

Moustapha LO - Maram LETAIEF - Ilyes HAMDI
16/12/2022

Introduction

Le projet consiste à élaborer, pour l'entreprise CompuOpti, une solution pour la planification de personnel et de l'affectation de projets. Un projet réalisé entraîne un gain, il doit être réalisé avant une date limite à partir de laquelle des pénalités financières sont appliquées. Chaque projet nécessite une liste de compétences avec un nombre décrivant la charge de travail pour chacune. Chaque membre possède un ensemble de compétences.

L'objectif est d'élaborer un emploi du temps pour chaque membre du personnel avec en entrée les qualifications de chacun ainsi que les besoins de chaque projet pour maximiser le gain de notre client.

Partie 1 : Modèle permettant de calculer la surface des solutions non-dominées du problème d'optimisation multiobjectif

Ensembles

- S : L'ensemble du personnel
- Q : L'ensemble des compétences
- J : L'ensemble des projets
- H : horizon temporel

Paramètres

Projet : gain, date limite, compétences requises avec nombre de jours pour chacune, pénalité

- c^j : pénalité sur le projet j par jour supplémentaire
- Q_j : ensemble des charges en compétences du projet j
- g^j : gain du projet j s'il est réalisé
- d^j : date de livraison du projet j

Membre : compétences, congés, projets

- Q^i l'ensemble des compétences du personnel i

- V^i l'ensemble des jours de congé du personnel i

Variables de décision

Les variables de décision sont définies pour définir les objectifs du modèle. Dans notre cas, on a 4 modèles différents qui seront expliqués en détail plus tard:

Pour la définition de l'objectif 1

- $p^i_{j,k,t}$: Le personnel i est affecté au projet j sur la compétence k pour le jour t (variable binaire)
- Y^j : 1 si le projet j est réalisé, 0 sinon (variable binaire)
- e^j : le jour de fin du projet j (variable entière)
- l^j : le nombre de jour en retard pour la livraison du projet j (variable entière)
- n^j_k: le nombre de jour en compétence k par projet j (variable entière)

Pour la définition de l'objectif 2

- a^i_j : 1 si le personnel i est assigné au projet j, 0 sinon (variable binaire)
- n^i: le nombre total des projets assignés au personnel i (variable entière)
- max^i: le nombre maximal des projet assigné à un personnel (variable entière)

Pour la définition des objectifs 3 et 4

- s_j: le jour de départ du projet j (variable entière)
- n^j: étendue du projet j (variable entière)
- max^j: l'étendue maximale de tous les projets (variable entière)

Contraintes

1. Un personnel i n'est affecté qu'à un seul projet j et une seule compétence k par jour t

   $$\forall i \in S, \forall t \in H, \sum_{j \in J, k \in Q} p^i_{j,k,t} \leqslant 1$$

   .

2. Un personnel i ne travaille pas s'il est en congé le jour t

   $$\forall i \in S, \forall t \in V^i, \sum_{j \in J, k \in Q} p^i_{j,k,t} \leq 0$$

3. Un personnel i ne peut être associé au projet j que s'il possède la compétence k nécessaire pour le projet j:

$$\forall i \in S, \forall t \in H, \forall j \in J, \forall k \in Q \setminus Q^j \cup Q^i, p^i_{j,k,t} = 0$$

4. Définition de n^j_k le nombre de jour en compétence k par projet j:
$$\forall j \in J, \forall k \in Q^j, \sum_{i \in S, t \in H} p^i_{j,k,t} \leq n^j_k$$

5. Définition de Y^j :
$$\forall k, Y^j \cdot n^j_k \leq \sum_{i \in S, t \in H} p^i_{j,k,t}$$

6. Définition de e^j la date de fin du projet j:
$$\forall k \in Q, \forall i \in S, \forall j \in J, \forall t \in H, p^i_{j,k,t} \cdot t \leq e^j$$

7. Définition de l^j :
$$e^j - d^j \leq l^j$$

8. e^j est supérieure ou égale à 1:
$$\forall j \in J, 1 \leq e^j$$

9. e^j est inférieure ou égale à la valeur maximale de l'horizon H:
$$e^j \leq H[-1]$$

10. Définition de a^i_j:
$$\forall i \in S, \forall j \in J, \forall t \in H, \forall k \in Q, p^i_{j,k,t} \leq a^i_j$$

11. Définition de n^i:
$$\forall i \in S, \sum_{j \in J} a^i_j \leq n^i$$

12. Définition de max^i :
$$\forall i \in S, n^i \leq max^i$$

13. Définition de s^j:
$$\forall j \in J, \forall t \in H, \forall i \in S, \forall k \in Q, s^j \leq t \cdot p^i_{j,k,t} + H[-1] \cdot (1 - p^i_{j,k,t})$$

14. s^j est supérieure à 1:
$$\forall j \in J, 1 \leq s^j$$

15. Définition de n^j :

$$\forall j \in J, e^j + 1 - s^j \leq n^j$$

16. Définition de max^j :

$$\forall j \in J, n^j \leq max^j$$

objectifs

Objectif 1: Maximiser le bénéfice de l'entreprise

$$Maximize \quad \sum_j \left(g^j \cdot Y^j - l^j \cdot c^j\right)$$

Objectif 2: Minimiser le nombre de projets maximum par personnel

$$Minimize \quad max^i$$

Objectif 3 : Minimiser le temps de réalisation maximum d'un projet

$$Minimize \quad max^j$$

Objectif 4 : Minimiser la durée de travail n^j sur chaque projet j:

$$Minimize \quad \sum_{j \in J} n^j$$

Modèle

Test avec les 3 instances fournies

Choix des priorité des objectifs:

Le Premier objectif est prioritaire et on lui confie la priorité la plus importante. Le deuxième objectif vient en deuxième position. Les objectifs 3 et 4 sont combinés ensemble en s'assurant que le max^j soit le majorant de la somme des n^j en le multipliant par 100 (un majorant de l'horizon de toutes les instances de données). Ce nouvel objectif est aussi tenu en deuxième position.

Instance small:

Exemple de planning proposé par le modèle avec un gain de 65, $max^i = 4$ et $max^j = 2$.

	1	2	3	4	5
Olivia	Job1 B	Job1 C	Job4 B	Job3 C	Job5 C
Liam	X	Job1 A	Job4 B	Job3 A	nan
Emma	nan	X	Job4 C	Job3 C	Job5 C

Ici, on ne fait pas le *Job2* et tous les projets sont réalisés sans retard.

RQ: les cases rouges dénotent les jours de congé de chaque personnel i . Les cases nan pour le jour où aucun projet j n'était assigné au personnel i.

Instance medium:

Exemple de planning proposé par le modèle avec un gain de 413, $max^i = 7$ et $max^j = 8$.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Olivia	X	X	Job5 C	Job5 C	Job5 C	Job5 C	Job3 A	Job3 A	Job3 C	Job3 A	Job3 A	Job4 A	Job4 A	Job4 A	Job4 A	Job4 A	Job4 A	Job1 A	Job1 A	Job1 A	Job1 A	nan
Liam	X	X	Job5 D	Job5 D	Job5 D	Job5 D	Job5 D	Job5 D	Job6 E	Job6 D	Job6 E	Job6 E	Job6 D	Job6 D	Job6 E	Job6 E	Job9 E	Job9 E	Job7 D	Job7 E	Job7 E	Job7 E
Emma	Job5 B	Job5 B	Job5 B	Job12 H	Job12 H	Job12 H	Job12 H	X	X	Job12 H	Job12 H	Job15 H	Job15 H	Job15 H	Job15 H	Job9 H	Job9 H	Job1 B	Job1 B	Job1 B	Job1 B	nan
Noah	Job14 G	Job14 G	Job14 J	Job14 G	Job5 D	Job5 D	Job13 J	Job13 J	Job13 J	Job13 J	Job13 J	Job13 J	Job15 G	Job15 G	Job15 G	Job6 D	Job9 G	Job9 G	Job7 D	Job7 D	Job7 D	Job7 D
Amelia	Job14 J	Job14 J	Job14 J	Job14 G	Job12 J	Job12 J	Job12 J	Job13 J	Job13 J	Job13 J	Job13 J	Job13 J	Job13 J	Job15 G	X	X	Job9 E	Job9 E	Job9 G	Job7 E	Job7 E	Job7 F

Générateur d'instances

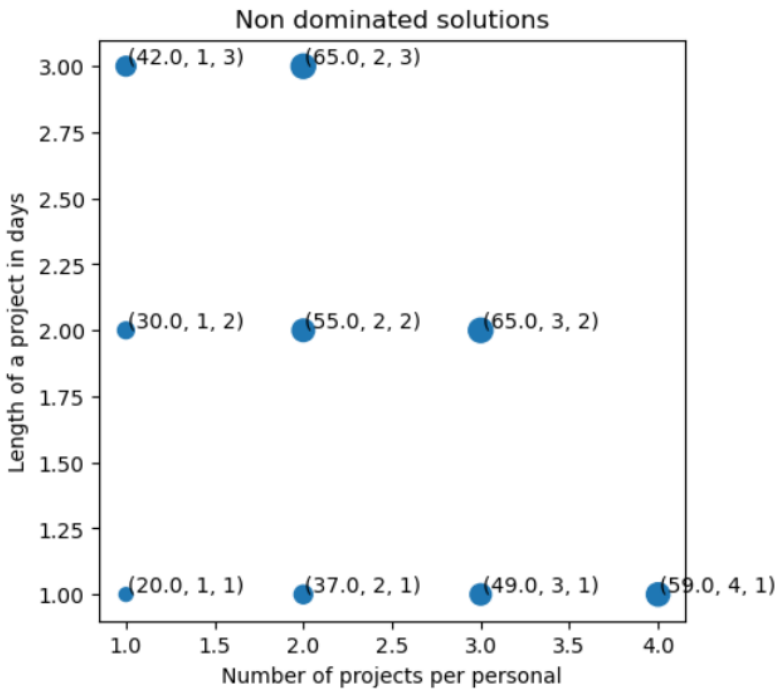
Méthode d'epsilon pour trouver la solution optimale:

Démarche suivie:

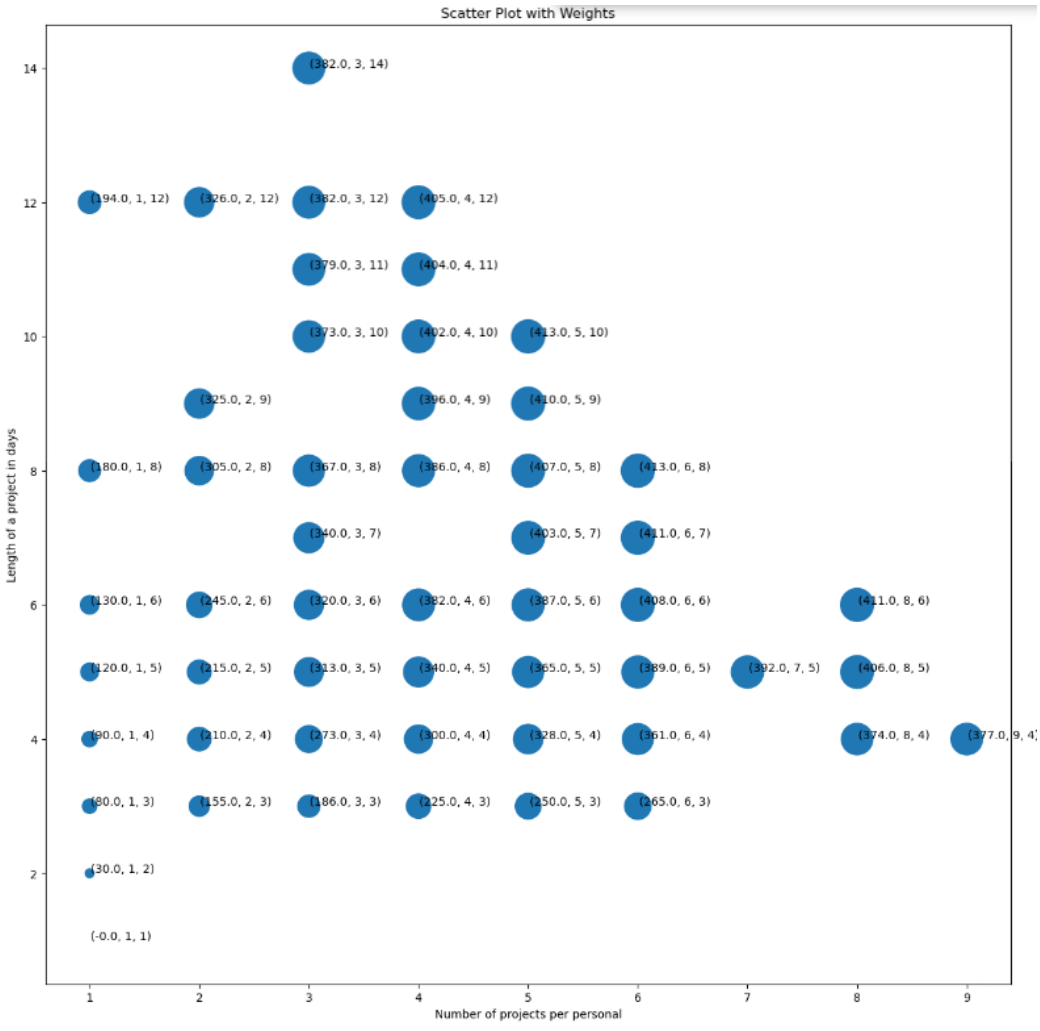
On crée un modèle *model_eps* pour lequel on définit les mêmes variables de décision et les mêmes contraintes que notre premier modèle. On lui fixe un seul objectif: celui de la maximisation du gain total et on lui ajoute deux autres contraintes qui permettent d'imposer les valeurs de max^i et max^j. On met à jour *model_eps* en définissant dans des boucles *for* les différentes valeurs prises par max^i et max^j appartenant respectivement à $1, 2, ..., J$ et $1, 2, .., H$. Cette démarche nous permet d'identifier toutes les solutions possibles que *Gurobi* peut trouver. Néanmoins, certaines de ces solutions sont dominées. On définit alors une fonction *filter_dominated_solutions* qui garde que celles non dominées. Cette dernière vérifie pour chaque solution identifiée si on a déjà trouvé une solution qui la domine ou pas. Si c'est le cas, on la supprime sinon on l'ajoute.

Exemple appliqué à la petite instance:

Les solutions non dominées obtenues sont :



Exemple appliqué à la instance intermédiaire:



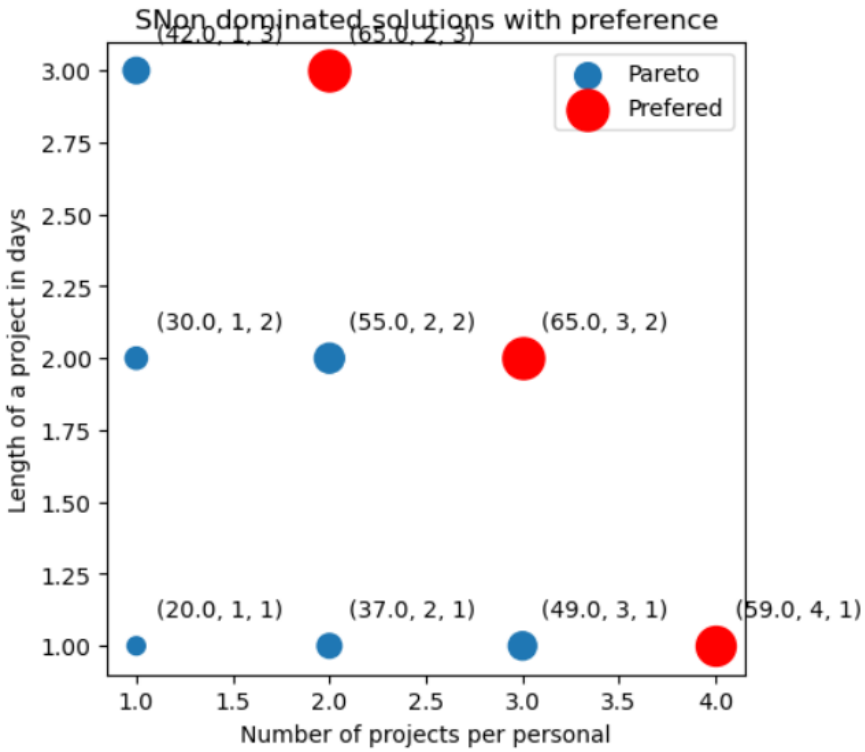
Partie 2 : Modèle de préférence permettant de discriminer entre les solutions de la surface des solutions non-dominées

Pour établir un modèle de préférence qui nous permet de discriminer entre les solutions non-dominées qu'on a identifiées par la première partie de notre code, on a mis en place 2 méthodes différentes:

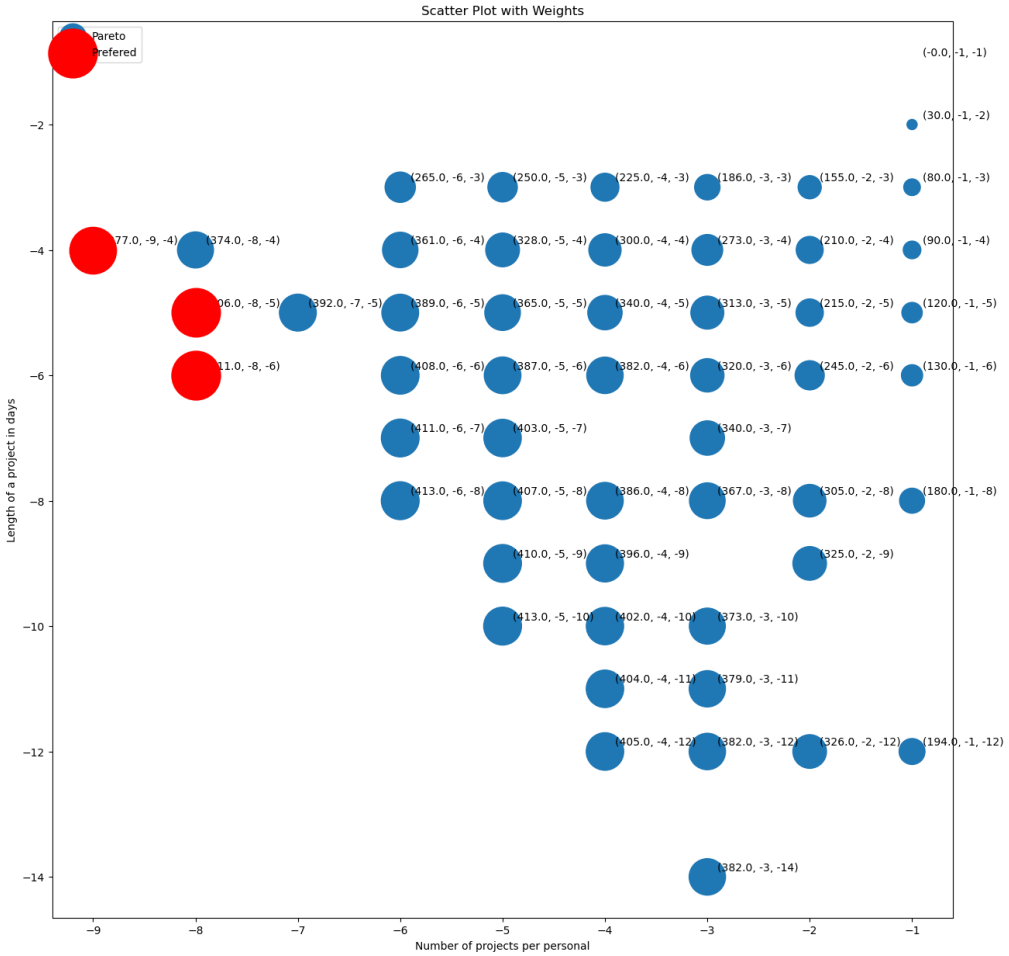
- **La Somme Pondérée:**

  On associe manuellement 3 différents poids à chaque objectif pour exprimer la priorité de chacun au niveau de la résolution du problème. Le gain, étant l'objectif le plus important, a un poids égal à 3. Puis, on accorde un poids de valeur 2 et 1 respectivement aux objectifs 2 et 3. De cette façon, on calcule le score de chaque solution non-dominée. Le score étant une somme pondérée des valeurs normalisées des trois objectifs . La solution ayant le meilleur score est celle ayant un gain de 65, une étendue maximale de 2 et un nombre maximal de projets associés égal à 3.

  Le résultat obtenu pour l'instance small est en dessous:



Ceux pour l'instance intermédiaire sont:



- **Méthode Boldas**

  L'idée, ici, est de réaliser une somme pondérée non pas des valeurs des objectifs mais de leurs rangs. Pour chaque objectif, le rang d'une valeur est déterminé parmi toutes les valeurs prises par cet objectif. Contrairement à la méthode de la somme pondérée, on ne normalise pas les valeurs des rangs.

Ces deux méthodes nous ont permis d'identifier la solution optimale. Pour mieux classer nos solutions selon leur faisabilité, on les répartit sur ces 4 différentes classes:

- **Classe inacceptable:** les solutions de planification qui ne sont pas intéressantes à mettre en place.
- **Classe correcte**: les solutions de planification qui sont intéressantes mais pas optimales.
- **Classe satisfaisante:** les solutions optimales pour notre client.

Pour y arriver, on fait l'hypothèse que l'objectif du maximisation du gain est le plus intéressant et que la minimisation de nombre de projets associés à chaque personnel est plus importante que la minimisation de l'étendue d'un projet.

Pour les méthodes, on a procédé comme suit:

- **Méthode des seuils:**

  L'objectif est de créer des règles simples pour la classification avec des seuils. On compare chacun des objectifs à des seuils qui sont choisis manuellement. Concrètement, si l'objectif 1 est trop faible ou si l'objectif 2 ou 3 est trop élevé, on associe à la solution la classe inacceptable. Pour la classe satisfaisante, on lui associe les solutions ayant des valeurs satisfaisantes pour les 3 objectifs (maximiser l'objectif 1 en minimisant les 2 autres). Le reste est accordé à la classe corrects.

- **Méthode d'optimisation sur Gurobi:**

  On reprend la méthode des seuils, mais cette fois-ci, on les détermine par la résolution d'un problème d'optimisation. En plus de la méthode des seuils, on reprend la méthode de la somme pondérée, en identifiant les poids encore une fois par la résolution du même problème.

  On commence par la création des trois classes en définissant deux seuils th1 et th2 sur la valeur du score (somme pondérée des valeurs des objectifs). Les solutions de la classe inacceptable ont un score inférieur à th1, ceux de la classe satisfaisante ont un score supérieur à th2 et ceux de la classe correcte ont un score entre les deux. Pour pouvoir définir notre problème, on initialise aléatoirement les poids de la somme et les th1 et th2 pour définir l'ensemble des solutions des trois classes. Cette initialisation satisfait ces contraintes:

  - $th1 < th2$
  - $w1 > w2 > w3$

Une fois fait, on définit les variables de décision ($w1, w2, w3, th1, th2, eps$) et les contrainte de notre modèle de la même façon que ce qui est décrit au dessus avec une seule modification: les seuils sont $th1 - eps$ et $th2 + eps pour les classe inacceptable et satisfaisante. L'intrt de l'utilisation de$ eps$$ est de maximiser le cardinal de la classe correcte de façon à minimiser à la fois les cardinaux des deux

autres classes. Ainsi, on facilitera le choix de la solution optimale parmi les solutions satisfaisantes (qui seront peu nombreuses).

L'exemple des résultats obtenus pour l'instance small est en dessous: