Systèmes de Décision et Préférences

Carl RIZK, David DE LA HERA CARRETERO, Dylan SECHET

9 février 2023

1 Introduction

La société *CompuOpti* emploie un certain nombre d'ingénieurs-développeurs qui sont staffés sur les projets des clients. Chaque projet nécessite de staffer un certain nombre de jours sur des compétences spécifiques et on souhaite de développer un algorithme pour définir les affectations du personnel aux projets.

Pour bien modéliser cette problématique, il est nécessaire de comprendre les différentes contraintes du modèle :

- Un membre du personnel ne peut être affecté à une qualification d'un projet que s'il possède cette qualification
- A tout instant, un membre du personnel ne peut être affecté qu'à un seul projet et qu'à une seule qualification intervenant dans ce projet
- Un membre de personnel ne peut pas être affecté à une qualification de projet un jour de congé
- Un projet n'est considéré réalisé que si tous les jours de travail dédiés à chacune des qualifications intervenant dans le projet ont été couverts par des employés.
- Un projet ne peut être réalisé qu'une fois sur une période de temps donnée.

2 Modélisation

2.1 Modélisation mathématique

Afin de modéliser mathématiquement ce problème on considère les suivantes variables :

- Un horizon de N jours, numérotés de 1 à N.
- E le nombre total d'employés, numérotés de 1 à E.
- Q le nombre total de qualifications, numérotés de 1 à Q.
- P le nombre total de projets, numérotés de 1 à P.
- M une constante arbitrairement grande.
- $penalite\_cste$ la pénalité journalière en cas de retard.

On introduit quelques fonctions :

- $conges(e) : [\![1; E]\!] \to \mathscr{P}([\![1; N]\!])$ qui a un employé associe l'ensemble de ses jours de congé.
- $qualifications(e) : [\![1; E]\!] \to \mathscr{P}([\![1; Q]\!])$ qui a un employé associe l'ensemble de ses qualifications.
- $duree(p, q)$ indique le nombre d'heures de compétence q nécessaires pour réaliser le projet p.
- $gain(p)$ indique le gain réalisé si le projet p est complété.
- $due\_date(p) : [\![1; P]\!] \to [\![1; N]\!]$ qui a un projet associe sa date limite.
- $penalite(p, j) = \begin{cases} penalite\_cste & si \quad j > due\_date(p) \\ \\ 0.00001 & sinon \end{cases}$

Enfin, on pose les variables de décision suivantes :

- $projet_{e,j,p,q}$ sont des variables booléennes valant 1 ssi l'employé e est affecté au projet p avec la qualification q le jour j.
- $realise_{p,j}$ indique si p est fini le jour j (vaut 0 tous les jours avant la fin du projet p, 1 tous les jours strictement après sa fin).
- $affecte_{e,p}$ est une variable booléenne qui vaut 1 ssi l'employé e est affecté au projet p.

— $debute_{p,j}$ est une variable booléenne qui vaut 1 ssi le projet p a débuté un jour k tel que $k \leq j$.

Ensuite on a modélisé les différentes critères à prendre en compte dans l'élaboration du planning :
— Objectif 1. Maximiser le bénéfice de l'entreprise :
$\max \sum_p gain(p) \cdot realise_{p,N+1} - \sum_{p,j \leq N+1} penalite(p,j) \cdot (1 - realise_{p,j})$
— Objectif 2. Réduire le nombre de changements de projet de chaque employé :
$max \left(- \sum_{e,p} affecte_{e,p} \right)$
— Objectif 3. Réduire l'étendue des projets :
$max \sum_p \sum_j realise_{p,j} - debute_{p,j}$

Enfin, il faut modéliser les différentes contraintes du modèle :
— Un membre du personnel ne peut être affecté à une qualification d'un projet que s'il possède cette qualification :
$\forall e \in [\![1;E]\!], \forall p \in [\![1;P]\!], \forall j \in [\![1;N]\!], \forall q \in [\![1;Q]\!] \backslash qualifications(e),\ projet_{e,j,p,q} = 0$
— À tout instant, un membre du personnel ne peut être affecté qu'à un seul projet et qu'à une seule qualification intervenant dans ce projet :
$\forall e \in [\![1;E]\!], \forall j \in [\![1;N]\!], \sum_p \sum_q projet_{e,j,p,q} \leq 1$
— Un membre de personnel ne peut pas être affecté à une qualification de projet un jour de congé :
$\forall e \in [\![1;E]\!], \forall p \in [\![1;P]\!], \forall q \in [\![1;Q]\!], \forall j \in conges(e), projet_{e,j,p,q} = 0$
— On ne peut pas travailler plus de la durée nécessaire pour une qualification dans un projet :
$\forall p \in [\![1;P]\!], \forall q \in [\![1;Q]\!], \sum_{e,j} projet_{e,j,p,q} \leq duree(p,q)$
— On impose des contraintes sur les variables de décision introduites pour modéliser les objectifs :
$\forall p \in [\![1;P]\!], \forall j \in [\![1;N+1]\!], \sum_q duree(p,q) - \sum_{j' < min(j,N)} \sum_{e,q} projet_{e,j',p,q} \leq M \cdot (1 - realise_{p,j})$
$\forall p \in [\![1;P]\!], \forall e \in [\![1;E]\!], \sum_{j,q} projet_{e,j,p,q} \leq M \cdot affecte_{e,p}$
$\forall p \in [\![1;P]\!], \forall j \in [\![1;N]\!], \sum_{j' \leq j} \sum_{e,q} projet_{e,j',p,q} \leq M \cdot debute_{p,j}$

2.2 Visualisation des résultats

Afin de pouvoir vérifier les bonnes résultats de notre modélisation, on a développé un outil de visualisation pour pouvoir vérifier visuellement les résultats obtenus par notre algorithme.

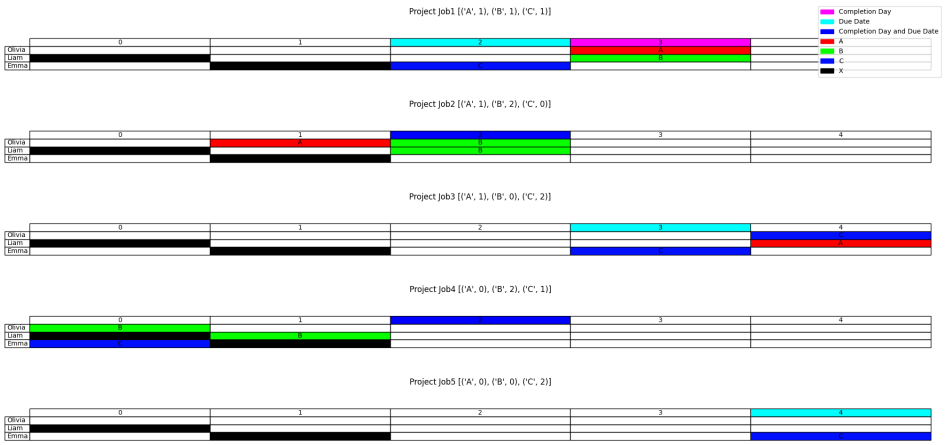

FIGURE 1 – Qualification à faire par projet, par jour et employée

3 Algorithme ϵ-constraint

L'algorithme introduit en cours ne gère que le cas de deux fonctions objectif. Présentons un algorithme qui fonctionne pour 3 fonctions objectif à maximiser, $f1, f2, f3$.

- Déterminer x^0 une solution optimale de $max\ f_1$
- $ND \leftarrow \{x^0\}$
- $\epsilon_2^{min} \leftarrow f_2(x^1);\ \epsilon_3 \leftarrow f_3(x^1) + \epsilon$
- Tant que le problème $min\ f_1 : f_3 \geq \epsilon_3$ admet une solution optimale x^{nd}
  - $ND \leftarrow ND \cup \{x^{nd}\}$
  - $\epsilon_2^{min} \leftarrow min\left(f_2(x^1), \epsilon_2^{min}\right)$
  - $\epsilon_3 \leftarrow f_3(x^{nd}) + \epsilon$
- $\epsilon_2 \leftarrow \epsilon_2^{min} + \epsilon$
- Tant que le problème $min\ f_1 : f_2 \geq \epsilon_2$ admet une solution optimale x^{nd}
  - $\epsilon_2 \leftarrow \epsilon_2 + \epsilon;\ \epsilon_3 \leftarrow f_3(x^{nd}) + \epsilon$
  - $ND \leftarrow ND \cup \{x^{nd}\}$
  - Tant que le problème $min\ f_1 : f_2 \geq \epsilon_2, f_3 \geq \epsilon_3$ admet une solution optimale x^{nd}
    - $\epsilon_3 \leftarrow f_3(x^{nd}) + \epsilon$
    - $ND \leftarrow ND \cup \{x^{nd}\}$

L'ensemble ND renvoyé par cet algorithme contient également des solutions non-dominées, et doit être ultérieurement filtré. Pour des problèmes à objectifs entiers comme le nôtre, on peut fixer $\epsilon = 1$.

On pourrait avoir l'impression au premier abord que la première boucle "Tant que" permettant de déterminer ϵ_2^{min}. Montrons qu'elle est nécessaire à l'aide d'un exemple : si l'on prend f_1 la fonction objectif 2, f_2 la fonction objectif 3 et f_3 la fonction objectif 1, la solution x^0 obtenue lors de l'initialisation aura ses deux premiers termes à 0. En effet, minimiser le nombre d'employés affectés f_1 conduit à n'affecter aucun employé, ce qui fait que la somme des longueurs des projets vaut aussi 0. La deuxième boucle "Tant que" ne pourra alors pas s'exécuter (il n'existe pas de solutions où le nombre d'employés affecté est inférieur à -1), et l'algorithme renverra $ND = \{x^0\}$, ce qui est évidemment incorrect.

En pratique, on choisit f_1 correspondant à la fonction objectif 1, f_2 à la fonction objectif 2, et f_3 à la fonction objectif 3. Cela permet d'éviter d'avoir à gérer les sauts de valeurs plus grands que 1 dans les valeurs de f_1, ce qui demanderait d'adapter ϵ pour une implémentation efficace. Par ailleurs, la fonction objectif utilisée n'est pas réellement f_1, mais $f_1 + \alpha \cdot f_2 + \alpha \cdot f_3$ avec $\alpha << 1$ pour encourager l'algorithme à maximiser $f2$ et $f3$ à $f1$ égaux.

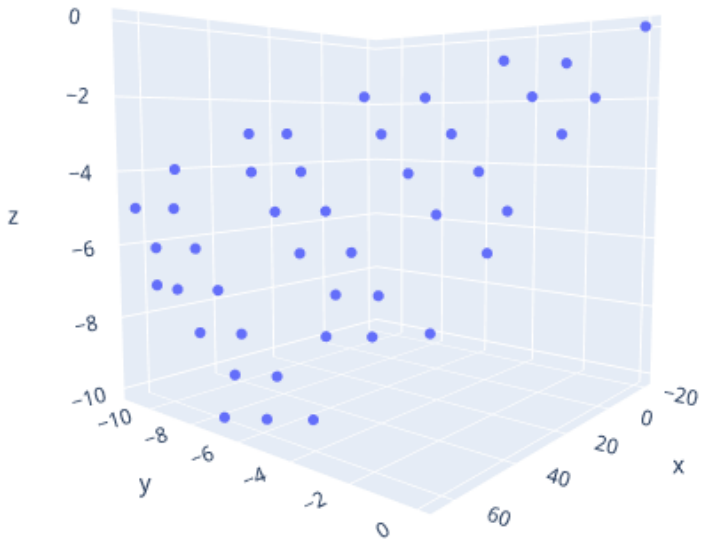

FIGURE 2 – Surface des solutions non-dominées

4 Modélisation des préférences

Une fois les solutions non-dominées obtenues, il faut les départager. Pour cela, dans une situation réelle, on ferait appel à un décisionnaire métier, et l'on pourrait essayer d'apprendre ses préférences.

Dans notre cas, on implémente le décisionnaire comme une boîte noire : il prend en entrée une planification, et renvoie en sortie un booléen qui indique si cette planification lui semble acceptable ou non. Dans notre cas, on a implémenté le décideur comme une simple somme pondérée des trois fonctions objectif.

On génère dans un premier temps des exemples aléatoires de problèmes, puis on les résout pour un des 3 objectifs (choisi aléatoirement). En enregistrant la décision du décideur pour le planning obtenu, on constitue un dataset sur lequel on entraîne une régression logistique.

Les performances sont excellentes, puisqu'avec seulement 30 échantillons on obtient un f1-score de 0.949 (écart-type de 0.03 sur 10 tests).

5 Conclusion

Lors de ce projet, nous avons été introduits aux méthodes de modélisation d'un problème concret et à l'application de problèmes multiobjectifs. Nous avons été particulièrement surpris par l'explosion des temps de calcul dès que la taille du problème devient réaliste : l'algorithme ϵ-constraint devient intraitable sur les instances larges, et pêine déjà sur les instances moyennes. Il serait intéressant d'étudier des méthodes approchées pour la détermination de la surface des solutions non-dominées.