Projet - Systèmes de Décision et Préférences - Rapport

Mathian Akanati, Saad Chtouki, Ibrahim Rakib

1 Problème global **1**

2 Modélisation mathématique **1**
2.1 Horizons et espaces considérés 1
2.2 Fonctions intermédiaires 1
2.3 Variables de décision 2

3 Contraintes et critères **2**
3.1 Liste des contraintes du problème 2
3.2 Variables multi-contraintes 3
3.3 Fonctions-critère 3

4 Calcul des solutions non-dominées et analyse **4**
4.1 Construction de la fonction multi-objectif 4
4.2 Génération des solutions non-dominées 5
4.3 Discrimination des solutions non-dominées 6

---

1 Problème global

L'objectif de ce projet est de résoudre un problème de planification de personnel et d'affectation de projets pour la société *CompuOpti*, en respectant un certain nombre de contraites et en optimisant de multiples critères.
Plus précisément, l'algorithme devra réaliser une affectation pertinente des collaborateurs, chacun disposant de compétences personnelles et d'un emploi du temps individuel, à des projets, nécessitant eux un certain nombre de jours de travail par compétence, et produisant un certain gain (ou une pénalité en cas de retard).

2 Modélisation mathématique

2.1 Horizons et espaces considérés

On considère que cette planification s'étend sur un horizon des temps $h \subsetneq \mathbb{N}^*$ représentant les jours de projet.
L'ensemble des projets à traiter, numérotés de manière croissante à partir de 1, sera noté n, l'ensemble des compétences différentes q et l'ensemble des collaborateurs p, de sorte que

$$n, q, p \subsetneq \mathbb{N}^*$$

2.2 Fonctions intermédiaires

Pour faciliter la modélisation des variables de décision, on pose les grandeurs suivantes :

- Le vecteur Q associant à chaque collaborateur ses compétences. Ainsi,

$$Q : p \mapsto \mathcal{P}(q) \setminus \{\varnothing\}$$

- Le vecteur V associant à chaque collaborateur ses jours de congés posés. Ainsi,

$$V : p \mapsto \mathcal{P}(h)$$

- Le gain (G_i) défini pour tout projet $i \in n$
- La pénalité (P_i) définie pour tout projet $i \in n$
- Le vecteur C associant à chaque projet le nombre de jours de travail requis par compétence pour ce projet. Ainsi,

$$C : n \times q \mapsto \mathbb{N}$$

- La date limite de livraison du projet (D_i) définie pour tout projet i, à valeur dans l'horizon temporel h

2.3 Variables de décision

Les variables de décision définies sont alors les suivantes :

- Le tenseur X défini tel que

$$X : p \times n \times q \times h \mapsto \{0; 1\}$$

  Le scalaire $X_{i,j,k,l}$ vaudra 1 si le collaborateur i travaille sur la compétence k du projet j pendant la journée l, 0 sinon

- Le vecteur $(Y_i)_{i \in n} \in \{0; 1\}^{card(n)}$ renvoie 1 si le projet est fini avant la fin de l'horizon temporel, 0 sinon.
- Le vecteur $(L_i)_{i \in n}$ donnant le nombre de jours de retard à la livraison d'un projet
- Le vecteur $(E_i)_{i \in n}$ donnant la date de fin effective d'un projet

3 Contraintes et critères

3.1 Liste des contraintes du problème

On relie maintenant les grandeurs posées précédemment par les contraintes à respecter lors de l'établissement de l'emploi du temps par notre algorithme :

- La contrainte de **qualification du personnel** s'écrit alors :

$$\forall\ (i, j, k, l) \in n \times p \times q \times h), (k \notin Q_p \Longrightarrow X_{i,j,k,l} = 0)$$

  En effet, un collaborateur ne peut travailler sur une compétence qu'il ne possède pas.

- La contrainte **d'unicité d'affectation quotidienne d'un collaborateur** se note :

$$\forall\ j \in p, \forall\ l \in h, \sum_{i \in n} \sum_{k \in q} X_{i,j,k,l} \leq 1$$

  traduisant que chaque collaborateur j en un jour l ne peut être associé au maximum qu'à une compétence k sur le projet i

- La contrainte de **congé** s'exprime comme :

$$\forall\ j \in p, \forall\ l \in V_j, \sum_{i \in n} \sum_{k \in q} X_{i,j,k,l} = 0$$

  signifiant que pour tout collaborateur j, et qu'en tout jour de congé l de ce collaborateur, le collaborateur ne travaille sur aucune compétence d'aucun projet $(X_{i,j,k,l} = 0)$
  La condition $X_{i,j,k,l} = 0$ posée pour tout $(i, j, k, l) \in n \times p \times q \times V_j$ est équivalente à celle-ci puisque $X_{i,j,k,l} \geq 0$

- La contrainte de **fin de projet** s'exprime telle que :

$$\forall (i,k) \in n \times q, (C_{i,k} > 0 \Longrightarrow Y_i \times C_{i,k} \leq \sum_{j \in p} \sum_{l \in h} X_{i,j,k,l})$$

Pour toute compétence k incluse dans un projet i (impliquant donc $C_{i,k} > 0$: le nombre de jours travaillés sur la coméptence k du projet i est strictement positif), le nombre de jours travaillés sur la compétence par les collaborateurs pendant l'horizon temporel sera toujours supérieur à $Y_i \times C_{i,k}$ tant qu'il n'y a pas complétion, avec égalité si le projet est bien terminé à la fin de l'horizon temporel ($Y_i = 1$ dans ce cas)

- La contrainte de **réalisation d'un projet** s'obtient lorsque toutes les compétences ont été traitées par un collaborateur pendant un nombre de jour adéquat :

$$\forall\ (i,k) \in n \times q, \sum_{j \in p} \sum_{l \in h} X_{i,j,k,l} \leq C_{i,k}$$

- La contrainte de **construction du jour de fin de projet** :

$$\forall\ (i,j,k,l) \in n \times p \times q \times h, X_{i,j,k,l} \times l \leq E_i$$

En optimisant, E_i vers la plus petite valeur de l telle qu'il existe i, j, k tels que $X_{i,j,k,l} = 1$

- La contrainte de **retard**, reliant directement la date de fin effective du projet, la date limite de livraison du projet et le nombre de jours de retard :

$$\forall\ i \in n,\ E_i - D_i \leq L_i$$

3.2 Variables multi-contraintes

Pour réduire la durée de calcul, et faciliter l'expression des objectifs et de la fonction d'optimisation, nous avons posé les variables suivantes :

- Le vecteur $(B_i)_{i \in n}$ donne la date de début du projet i.
  Sa contrainte (de construction "logique") est alors

$$\forall\ (i,j,k,l) \in n \times p \times q \times h,\ B_i \leq X_{i,j,k,l} \times l$$

Cette expression est en réalité équivalente à

$$B_i = \min\{\text{argmin}_{l \in h}(X_{i,j,k,l} = 1),\ \forall\ (j,k) \in p \times q\}$$

- La matrice R donne, pour un projet $i \in n$, la participation du collaborateur $j \in p$ (auquel cas $R_{i,j} = 1$), ou non ($R_{i,j} = 0$).
  Sa contrainte de construction donne alors :

$$\forall\ (i,j,k,l) \in n \times p \times q \times h,\ R_{i,j} \geq X_{i,j,k,l}$$

3.3 Fonctions-critère

Afin de construire la fonction objectif, on pose au préalable les fonctions-critère permettant d'obtenir les critères d'intérêt que sont :

- Le profit, obtenu par l'expression :

$$f_p = \sum_{i \in n} Y_i \times G_i - L_i \times P_i$$

autrement dit, la somme sur les projets des gains des projets finis, à laquelle on retranche le produit retard-pénalité par projet.

- La durée des projets terminés est reliée à la grandeur :

$$f_d = \sum_{i \in n} (E_i - B_i) \times Y_i$$

On réalise le calcul uniquement pour les projets finis - la durée du projet n'est pas définie sinon.

- La somme du nombre de projets par collaborateur est :

$$f_n = \sum_{i \in n} \sum_{j \in p} R_{i,j}$$

Justification : *On a fait le choix de minimiser ce critère pour minimiser le nombre de projets par personne, sachant pertinemment que, puisque les résultats sont agrégés donc moyennés, on pourrait obtenir un grand nombre de projets pour quelques exceptions parmi le pool d'individus. Il nous semble en effet* ***plus pertinent au vu de l'objectif commercial du projet de minimiser la moyenne, plutôt que de minimiser le maximum*** *du nombre de projets par collaborateur ; si une solution convient à* 99% *des collaborateurs, nous souhaitons l'adopter plutôt que de dégrader la satisfaction globale pour arranger les* 1% *restants.*

4 Calcul des solutions non-dominées et analyse

4.1 Construction de la fonction multi-objectif

À partir des fonctions-critère, on peut construire une fonction multi-objectif combinant les trois critères à l'aide d'un système de poids :

$$\lambda_1 f_p + \lambda_2 f_d + \lambda_3 f_n \qquad \begin{cases} f_p, f_d, f_n \text{ fonctions critères définies ci-dessus} \\ \lambda_1, \lambda_2, \lambda_3 \in \mathbb{R} \end{cases}$$

Puisque l'on cherchera à maximiser la fonction multi-objectif, et que l'on souhaite maximiser uniquement le profit f_p, et minimiser la durée moyenne d'un projet f_d et le nombre moyen de projet par collaborateurs f_n, cela revient en fait à poser la fonction objectif comme :

$$\lambda_1 f_p - \lambda_2 f_d - \lambda_3 f_n \ , \qquad \lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}_+$$

De plus, nous souhaitons donner **beaucoup de valeur au profit**, car nous estimons que c'est le **critère central de ce problème** ; aussi, on aura

$$\lambda_1 >> \lambda_2, \lambda_3$$

Par ailleurs, pour étudier l'impact de l'importance (donc des poids) des critères de durée de projet et de nombre de projets par collaborateur, on souhaite obtenir une variation inversée de l'importance des poids.

Finalement, on propose la fonction suivante :

$$\beta f_p - \alpha f_d - (1 - \alpha) f_n \qquad \begin{cases} \alpha \in [0; 1] \\ \beta >> \alpha \end{cases}$$

de sorte qu'en faisant varier α, on fasse varier l'importance d'un critère par rapport à l'autre, et donc la solution optimale en conséquence.

Pour analyser la variation des solutions en fonction de l'importance des critères, on se propose de les calculer pour différentes valeurs de α, comprises entre 0 et 1 - en pratique, nous allons réaliser un découpage dichotomique de l'intervalle $[0; 1]$ en un nombre donné de points et traiter les résultats.

Par ailleurs, pour éviter la prépondérance du profit dans le calcul de l'objectif (induite à priori par la condition $\beta >> \alpha$), on réalise le choix arbitraire

$$\beta = 10$$

de sorte que les objectifs secondaires ne soient pas totalement ignorés, leur poids représentant 10% de celui du profit.

4.2 Génération des solutions non-dominées

Avec la fonction multi-objectif implémentée, on calcule les solutions pour chaque valeur de α possible. On pose également certaines grandeurs, directement reliées aux fonctions-critère, pour étudier de manière plus fine les performances des solutions entre elles - on va en fait chercher à obtenir des moyennes :

- Le profit, de nouveau obtenu par l'expression :

$$\sum_{i \in n} Y_i \times G_i - L_i \times P_i = f_p$$

- La durée moyenne d'un projet :

$$1 + \frac{\sum_{i \in n} (E_i - B_i) \times Y_i}{\sum_{i \in n} Y_i} = \frac{f_d}{\sum_{i \in n} Y_i}$$

On réalise le calcul uniquement pour les projets finis - la durée du projet n'est pas définie sinon.

- La somme du nombre moyen de projets par collaborateur est :

$$\frac{\sum_{i \in n} \sum_{j \in p} R_{i,j}}{\text{card}(p)} = \frac{f_n}{\text{card}(p)}$$

On peut alors identifier les solutions non-dominées en les comparant deux à deux suivant les trois critères.

À titre d'exemple, on peut calculer les solutions non-dominées associées au problème de l'énoncé du sujet - pour lequel on a défini des gains, pénalités et deadlines au préalable. On obtient ainsi les solutions non-dominées suivantes :

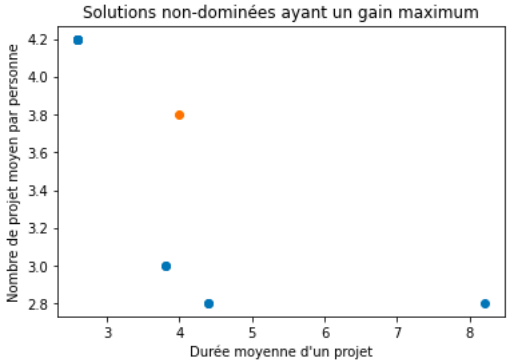

Figure 1: Solutions non-dominées à gain maximal pour l'exemple de l'énoncé du projet (bleu) et l'exemple de solution de l'énoncé (orange)

Ainsi, on réalise que l'exemple de solution donnée dans l'énoncé est strictement dominée par l'une de nos solutions, dont l'emploi du temps est (en numéro de projet/compétence):

	Jour 0	Jour 1	Jour 2	Jour 3	Jour 4	Jour 5	Jour 6	Jour 7	Jour 8	Jour 9	Jour 10	Jour 11
0		1A	1A	1B	1A		3A	3A		3A	4A	4B
1	0C	1C	0C	1C	1C	2C	3C	2C	3C	3C	4C	4C
2	0A	0A	0A	0A		2A	2A	2A	2A			
3		0B	0B	0B				3B	3B	3B		
4	1C	1C	1C	1B	1C	2C	2C	2C	2B		4C	4C

Figure 2: Exemple de solution non-dominée à l'exemple présenté dans l'énoncé

4.3 Discrimination des solutions non-dominées

Afin de privilégier les solutions performantes, nous adoptons la classification suivante parmi les solutions trouvées :

1. Solution non-dominée et gain maximal : **satisfaisante**
2. Solution non-dominée mais gain dominé : **acceptable**
3. Solution dominée : **inacceptable**

Ainsi, en se basant une fois de plus sur le fait que le profit est le critère privilégié dans cette analyse, on peut rechercher uniquement les solutions satisfaisantes d'un problème. On les détermine d'ailleurs pour le jeu d'instances `toy_instance.json` :

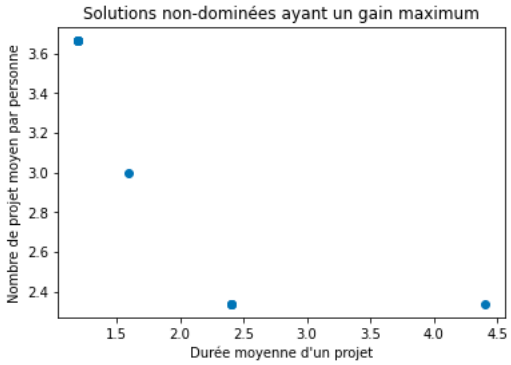

Figure 3: Solutions satisfaisantes pour le jeu de données `toy_instance.json`

Ces solutions seront donc privilégiées dans le cas d'une interaction avec une entreprise ayant des besoins de planification.
Néanmoins, selon le profil de l'entreprise ou des employés (appétance pour les projets courts ou longs, travail sur un grand nombre de projets ou non...), une solution pourra alors être proposée en priorité au client.

Pour le calcul des solutions satisfaisantes de l'instance de taille moyenne `medium_instance.json`, nous n'avons malheureusement pas pu mener le calcul à son terme, comme le montre la figure suivante:

```
model.update()
nb_points = 10
profits = []
avg_durations = []
avg_ppps = []

gain_obj = quicksum(Y[j] * G[j] - L[j] * P[j] for j in range(n))
duration_obj = quicksum((E[j] - B[j])*Y[j] for j in range(n))
project_per_person_obj = quicksum(R[i,j] for i in range(p) for j in range(n))

for alpha in range(nb_points):
    alpha = alpha / (nb_points -1)

    # Nous donnons beaucoup de valeur au gain car nous estimons que c'est le critère le plus important
    # Voyons l'impact des poids attribués aux critères sur la durée et le nombre de personnes par projet

    model.setObjective(10 * gain_obj - alpha * duration_obj - (1-alpha) * project_per_person_obj , GRB.MAXIMIZE)
    model.optimize()

    profits.append(get_profit())
    avg_durations.append(get_avg_project_duration())
    avg_ppps.append(get_avg_projects_per_person())

print("\nOptimal objective value: ", int(model.ObjVal))
```

442m 25.1s Python Python Python Python Python Python Python Python

Figure 4: Le calcul des solutions satisfaisantes de `medium_instance.json` dépasse les 7 heures, sans succès

Nous soupçonnons dès lors l'existence d'une opération fastidieuse, optimisable, mais correcte (puisque fonctionnelle sur les instances de petite taille), augmentant significativement la complexité de la recherche de solutions, et ne nous permettant pas de conclure sur les instances de taille supérieures.