CentraleSupélec

Optimisation de l'organisation du travail chez CompuOpti

Seong Woo Ahn
Rodolphe Nonclercq
Tanguy Blervacque

Sommaire

Le problème

CompuOpti → entreprise de service en optimisation et aide à la décision.

Son **problème** :

Optimiser l'affectation du personnel aux projets qui lui sont proposés

Ses **objectifs** :

Maximiser son profit
Minimiser le nombre de projets différents par employé
Minimiser la longueur des projets

La modélisation

Maximisation du profit

➔ Quels projets choisir ?
job_is_done(p)

➔ Si un projet est choisi, quand démarre-t-il ? se termine-t-il ?
start_date(p)
finish_date(p)
job_has_delay(p)
job_penalty(p)

➔ Quel est alors le profit fait par projet?
job_profit(p)

➔ Maximiser la somme

Minimisation du nombre de projets par employé

➔ Quand travaille chaque employé, sur quel projet et sur quelle qualification ?
work(c , p , j , q)

➔ Quel employé participe à quels projets ?
participate(p , c)

➔ Quel est l'employé qui en a le plus ?
max_job_per_staff

➔ Minimiser ce max

Minimisation de la longueur des projets

➔ Quand un projet est-il en cours ?
job_is_active(c , p)

➔ Combien de temps a-t-il duré ?
start_date(p)
finish_date(p)

➔ Quel est le plus long projet ?
max_len_job_staff

➔ Minimiser ce max

Les contraintes

Contrainte de qualification du personnel

```
for staff_idx, staff in enumerate(data['staff']):
    for qualification in data['qualifications']:
        if qualification not in staff['qualifications']:
            m.addConstr(work[staff_idx, :, :, qualification_to_idx[qualification]].sum() <= 0)
```

➔ Un membre du personnel ne peut être affecté à une qualification d'un projet que s'il possède cette qualification

➔ Fixe en partie *work(c , p , j , q)*

Les contraintes

Contrainte d’unicité de l’affectation quotidienne du personnel

```
for staff_idx, staff in enumerate(data['staff']):
        for day in range(horizon):
                m.addConstr(work[staff_idx, :, day, :].sum() <= 1)
```

➔ A tout instant, un membre du personnel ne peut être affecté qu’à un seul projet et qu’`a une seule qualification intervenant dans ce projet

➔ Fixe en partie *work(c , p , j , q)*

Les contraintes

Contrainte de congé

```
for staff_idx, staff in enumerate(data['staff']):
    for vacation_day in staff['vacations']:
        m.addConstr(work[staff_idx, :, vacation_day - 1, :].sum() <= 0)
```

➔ Un membre du personnel ne peut travailler un jour de vacances

➔ Fixe en partie *work(c , p , j , q)*

Les contraintes

Contrainte de couverture des qualifications du projet

```
for job_idx, job in enumerate(data['jobs']):
        for qualification, quantity in job['working_days_per_qualification'].items():
                m.addConstr(work[:, job_idx, :, qualification_to_idx[qualification]].sum() >= quantity - 1 + epsilon - M *
        (1 - job_is_done[job_idx]))
                m.addConstr(work[:, job_idx, :, qualification_to_idx[qualification]].sum() <= quantity - 1 + M *
        job_is_done[job_idx])
```

➔ 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 membres du personnel

➔ Fixe en partie *work(c , p , j , q)*

Les contraintes

Contrainte d'unicité de la réalisation d'un projet

➔ Un projet ne peut être réalisé qu'une fois sur une période de temps donnée

➔ Contrainte implicitement vérifiée car *job_is_done(p)* est un vecteur binaire

➡ *work(c , p , j , q)* et *job_is_done(p)* sont complètement fixés par ces contraintes

➡ Le reste des variables découlent de *work(c , p , j , q)* et sont définies par des contraintes dites "de caractérisation"

L’optimisation

Un fois les contraintes implémentées on lance l’optimisation des objectifs :

```
# Objectif 1
m.setObjectiveN(-job_profit.sum(), 0, 0)

# Objectif 2
m.setObjectiveN(max_job_per_staff, 1, 0)

# Objectif 3
m.setObjectiveN(max_len_job, 2, 0)
```

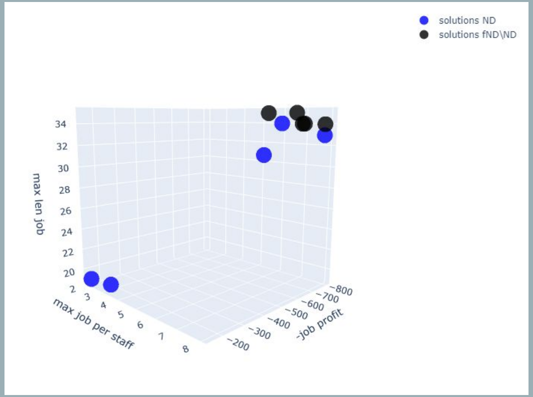
Résultats dans la suite…

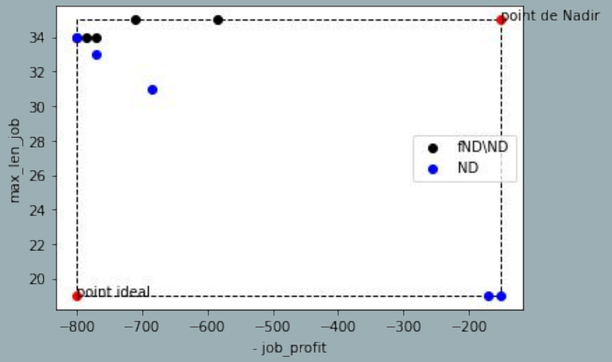
Les résultats

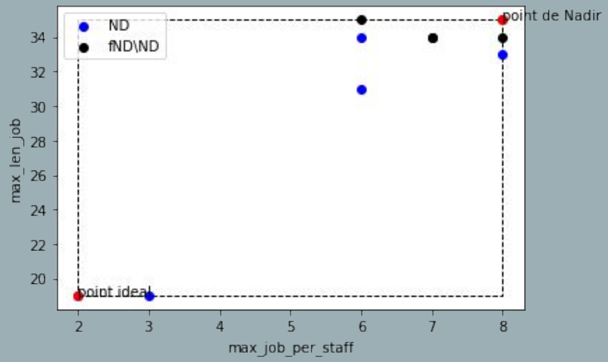
Table 2: Solutions non-dominées

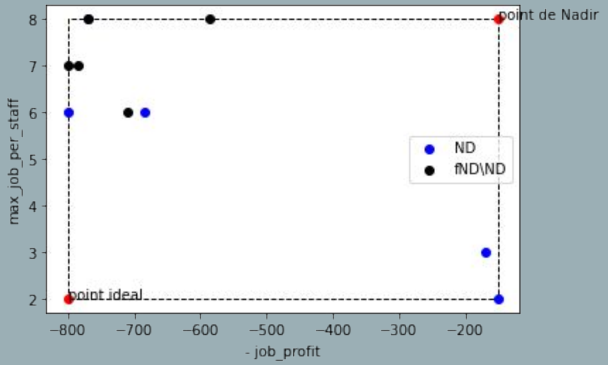
data	job profit	max job per staff	max len job
Toy dataset: solution 1	65	2	4
Toy dataset: solution 2	46	2	3
Medium dataset: solution 1	400	4	19
Medium dataset: solution 2	215	5	17
Large dataset: solution 1	800	6	34
Large dataset: solution 2	770	8	33
Large dataset: solution 3	685	6	31
Large dataset: solution 4	170	3	19
Large dataset: solution 5	150	2	19

Visualisations









Approche 1

Je souhaiterais donner un poids de:

- *0.6 sur le profit*
- *0.3 sur le nombre maximal de projet mené par quiconque collaborateur*
- *0.1 sur la durée consécutive d'un projet*

Fixons par ailleurs un seuil de majorité à 0.6.

$$\sum_{j \in S(a,a')} w_j \geq \lambda$$

Approche 1: résultats

solution 3
VS
solution 4

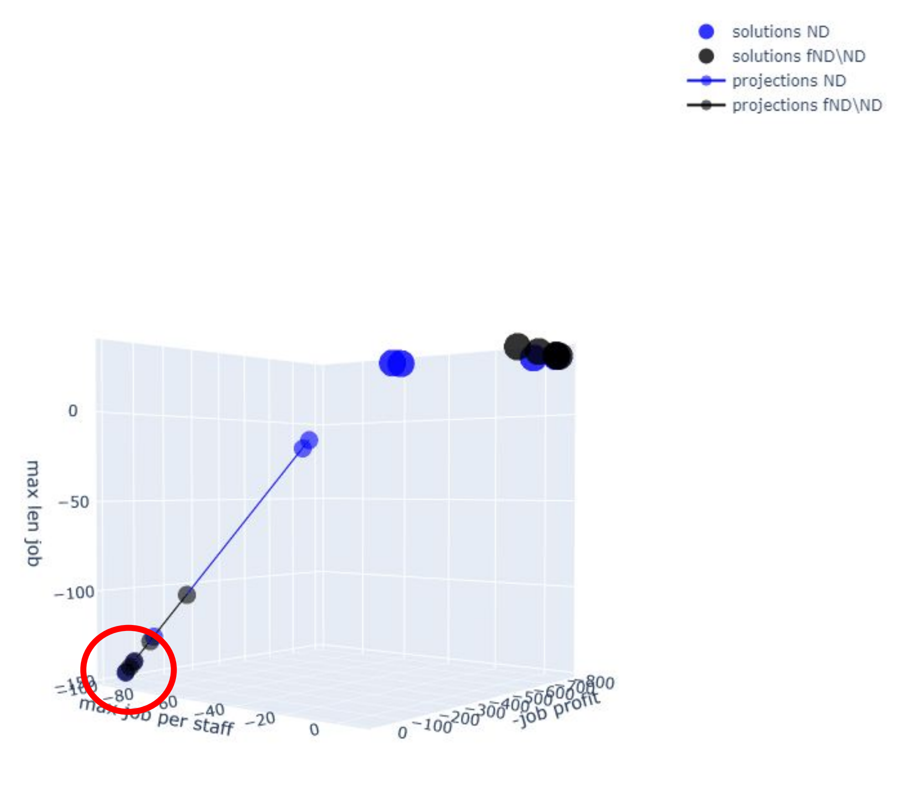
$$
\begin{bmatrix}
1 & 0.9 & 0.9 & 0.6 & 0.6 \\
0.1 & 1 & 0.6 & 0.6 & 0.6 \\
0.4 & 0.4 & 1 & \boxed{0.6} & 0.6 \\
0.4 & 0.4 & 0.4 & 1 & 0.7 \\
0.4 & 0.4 & 0.4 & 0.4 & 1
\end{bmatrix}
\qquad
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
$$

Large dataset: solution 1	800	6	34

Approche 1 bis: projection visuelle

Poids
choisis

w1 = 0.6
w2 = 0.3
w3 = 0.1



Approche 2

Je préfère:

- *la solution 1 à la solution 3*
- *la solution 4 à la solution 5*
- *….*

```
epsilon = 1e-3


m = Model("Solving linear program")

# Création de 2 variables continues v0 et v1
w1 = m.addVar(lb=0.1)
w2 = m.addVar(lb=0.1)
w3 = m.addVar(lb=0.1)
L = m.addVar(lb=0.5,ub=1.0)
# maj du modèle
m.update()
# Ajout de 3 constraintes
m.addConstr(w1 + w2 + w3 == 1)
m.addConstr(w1 + w2 >= L)
m.addConstr(w3 <= L - epsilon)
m.addConstr(w2 + w3 <= L - epsilon)
m.addConstr(w1 + w3 >= L)
# Fonction Objectif
m.setObjective(L, GRB.MAXIMIZE)
# Paramétrage (mode mute)
m.params.outputflag = 0
# Résolution du PL
m.optimize()
print("Optimal solution has weights (w1, w2, w3) = {} and lambda value L = {}".format((w1.x, w2.x, w3.x), L.x))
```

$$\Rightarrow \quad w_1 + w_2 \geq \lambda \;\&\; w_3 < \lambda$$

Approche 2: résultats

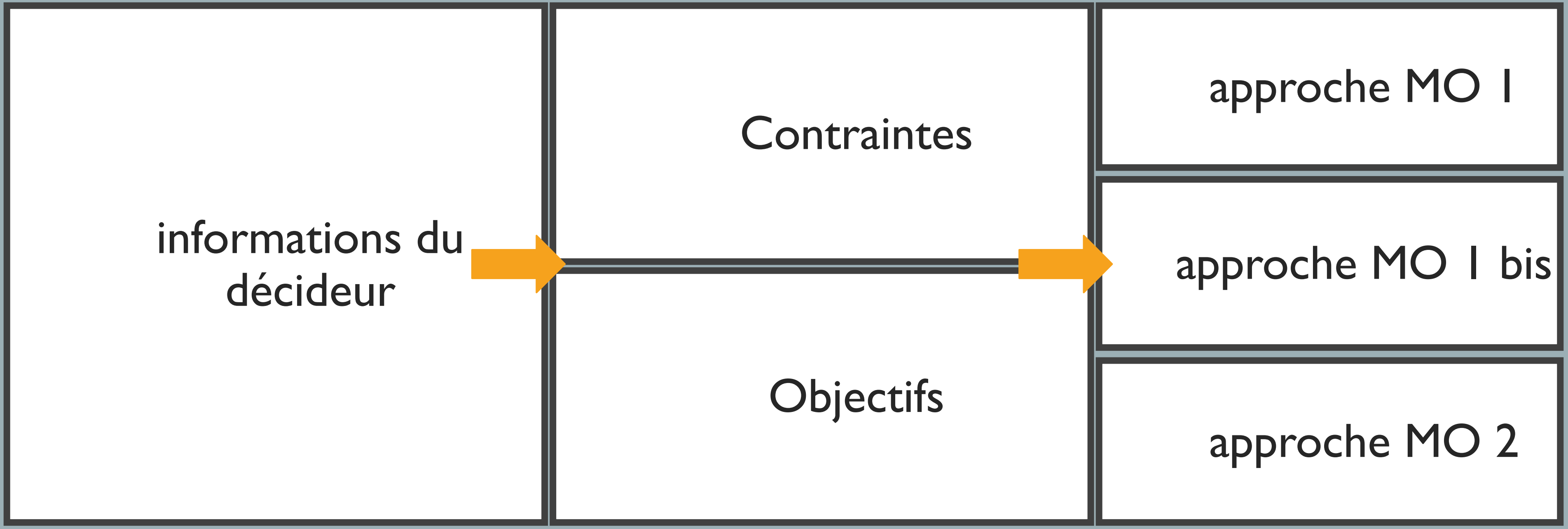
w1 = 0.8
w2 = 0.1
w3 = 0.1

seuil de majorité à 0,5

$$\begin{bmatrix} 1 & 0.9 & 0.9 & 0.8 & 0.8 \\ 0.1 & 1 & 0.8 & 0.8 & 0.8 \\ 0.2 & 0.2 & 1 & 0.8 & 0.8 \\ 0.2 & 0.2 & 0.2 & 1 & 0.9 \\ 0.2 & 0.2 & 0.2 & 0.2 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Large dataset: solution 1	800	6	34

Conclusion


informations du décideur
Contraintes
Objectifs
approche MO 1
approche MO 1 bis
approche MO 2