Systèmes de décision - optimisation de l'emploi du temps d'une société de conseil fictive

Membres du projet : Tristan Basler, Vivien Conti, Clément Boulay

Décembre 2022 - Février 2023

1 Description du sujet

Le projet consiste en une optimisation d'allocation de ressources (les travailleurs de l'entreprise CompuOpti) sur des jours de travail. La cliente est la directrice de l'entreprise, Margaux Dourtille. Margaux possède des clients (des entreprises ayant besoin de prestations d'ingénierie) qui lui soumettent des projets à réaliser dans un délai fixé. Margaux peut choisir, pour chaque projet, de prendre le contrat ou bien de le rejeter. Pour chaque projet réalisé, Margaux tire un bénéfice fixé à l'avance (paiement de l'entreprise cliente). Cependant si un projet est réalisé mais rendu avec du retard (c'est-à-dire, au-delà du délai initial fixé), l'entreprise est pénalisée.

Le personnel de CompuOpti est un ensemble d'ingénieur(e)s qui possèdent chacun(e) des compétences (parmi un ensemble tel que $\{A,\ B,\ C,\ D,\ E\}$ par exemple). Les projets sont formulés de la façon suivante :

Projet P1 : $\{4.A,\ 3.B,\ 2.C\}$. Cela se lit : le projet 1 nécessite 4 apports unitaires de compétence A, 3 de la compétence B et 2 de la compétence C. Il faudra alors (pour terminer le projet) allouer le personnel disponible (attention : le personnel peut prendre des congés !) pour valider les besoins du projet. En somme, il s'agit de faire des emplois pour les membres du personnel de façon à maximiser le bénéfice net de l'entreprise sur l'horizon de temps fixé que l'on se donne. Attention : bénéfice ne signifie pas "somme des gains des projets réalisés" car il faut y soustraire les pertes/pénalités éventuelles liées aux projets rendus en retard.

2 Modélisation du problème

2.1 Définition des variables de décision

Dans un premier temps, nous allons définir les variables caractérisantes de notre problème.

1. Les dimensions de notre problème :

   - $NP = \{1,\ \ldots\ ,\ n_p\}$: les différents projets proposés.
   - $NC = \{1,\ \ldots,\ n_c\}$: les différents collaborateurs.
   - $NA = \{1,\ \ldots,\ n_a\}$: les différentes aptitudes des collaborateurs.

- h : l'horizon du problème (en jours, considérés comme des entiers insécables).

2. Les caractéristiques financières :

   - $G = \{q_1, \ldots, q_{n_p}\}$: l'ensemble des gains (en points) des projets disponibles.
   - $P = \{p_{i,j}\}$: les pénalités progressives (en points) des projets choisis (la multiplication entre les pénalités unitaires et le nombre de jours de retard pris est implicite). Ainsi, $p_{i,j}$ correspond à la pénalité progressive en point du rendu du projet i avec un retard j (i.e., elle peut encore augmenter si le projet est rendu à un instant $j' > j$).

   On peut combiner ces deux informations en une seule matrice de la forme :
   $B = b_{i,\,j}$: le bénéfice net engendré par le projet i s'il est rendu au temps j, avec $b_{i,\,j} > 0$: le projet i rendu au temps j aura apporté un bénéfice net à l'entreprise.

3. Les caractéristiques opérationnelles :

   - $R = r_{i,j}$: la matrice des compétences (de taille $n_c * n_a$) avec $r_{i,\,j} = 1$ quand le collaborateur i possède la compétence j.
   - $PC = pc_{i,j}$: la matrice des compétences pour chaque projet avec $pc_{i,j} = k$ si le projet i a besoin de k unités de la compétence j. On en extrait MAX_{COST}, le coût maximal pour tous les projets et toutes les compétences.
   - $C = c_{i,j}$: la matrice des congés des collaborateurs de taille $n_c * h$ avec $c_{i,j} = 1$ si le collaborateur i est en congé au temps j sinon 0.

Dans un second temps, nous allons définir nos variables de décision pour la résolution du problème:

- $A = a_{i,\,j,\,k,\,l}$, la matrice des affections des collaborateurs de taille $n_c * h * n_p * n_a$ avec $a_{i,\,j,\,k,\,l} = 1$ si le collaborateur i est affecté au temps j au projet k avec la compétence l sinon 0.
- $DP = dp_{j,\,k}$, , la matrice de rendu de projets de taille $h * n_p$ avec $dp_{j,\,k} = 1$ si le projet k est rendu au temps i sinon 0.

- $BP\ = bp_{j,\,k},$, la matrice de début de projets de taille $h\ *\ n_p$ avec $bp_{j,\,k} = 1$ si le projet k est commencé au temps i sinon 0.
- d_{max}, la durée maximal entre le début et le rendu des projets.
- cp_{max}, le nombre de projet maximal auquel un collaborateur peut être affecté.

2.2 Définition des objectifs

Notre premier objectif est de maximiser le gain générer par l'entreprise. On peut donc l'exprimer ainsi :

$$\max \sum_{i=0}^{h} \sum_{j=0}^{n_p} dp_{j,\,k} * b_{j,k}, \tag{1}$$

Sous la forme matricielle,

$$\max\ ((DP * B^T) * 1) * 1^T, \text{ avec 1, le vecteur de taille h composé de 1.} \tag{2}$$

Le second objectif est minimiser le nombre de projets différents sur lesquels sont affectés les collaborateurs de l'entreprise. On peut donc l'exprimer ainsi :

$$\min \max_i, \sum_k^h 1_{i,k}$$

avec $1_{i,k}$ qui vaut 1 si le travailleur i a travaillé sur le projet k, 0 sinon.

Nous reformulons sur problème sous la forme :

$$\min cp_{max}, \text{ avec } cp_{max} \text{ qui représente le nombre maximum d'un projet affecté à un travailleur.} \tag{3}$$

La démonstration est en annexe.

Le troisième objectif est minimiser les temps de validation des projets. On peut donc l'exprimer ainsi :

$$\min \max_k, \sum_{j=0}^{h} j * (dp_{j,\,k} - bp_{j,\,k}) + \sum_{j=0}^{h} (dp_{j,\,k})$$

Nous reformulons sur problème sous la forme :

$$\min d_{max}, \text{avec } d_{max} \text{ qui représente le durée maximum d'un projet.} \tag{4}$$

La démonstration est en annexe.

2.3 Définition des contraintes

On peut distinguer deux types de contraintes. Par souci de clarté, nous posons les indices suivant :

- i, l'indice pour les travailleurs.
- j, l'indice temporel.
- k, l'indice pour les projets.
- l, l'indice pour les compétences.

Ainsi $a_{i,j,k,l} = 1$ indique que le travailleur i le jour j est affecté au projet k avec la compétence l.

1. Les contraintes directes :

   - Un travailleur ne peut travailler que sur un seul projet chaque jour. Un travailleur ne peut travailler que s'il n'est pas en congés. Ces deux contraintes peuvent se modéliser ensemble sous la forme. En effet, on a :
     $\forall$ i,j $\sum_{k,l} a_{i,j,k,l} <= 1$
     $\forall$ i,j $\sum_{k,l} a_{i,j,k,l} <= 1 - c_{i,j}$
     Or, $c_{i,j}$ vaut 1 ou 0. Donc la contrainte n°2 entraîne la première. On conservera donc :
     $\forall$ i,j $\sum_{k,l} a_{i,j,k,l} <= 1 - c_{i,j}$
   - Un travailleur ne peut travailler que dans ses compétences. Ainsi,
     $\forall$ i,j,k,l, $a_{i,j,k,l} <= r_{i,l}$

- Un projet ne peut être réalisé qu'une seule fois. Ainsi,

  $\forall$ k, $\sum_j dp_{j,k} <= 1$

- Un projet ne peut être débuté qu'une seule fois. Ainsi,

  $\forall$ k, $\sum_j bp_{j,k} <= 1$

- Un projet réalisé doit débuté et un projet débuté doit être réalisé. Ainsi,

  $\forall$ k, $\sum_j dp_{j,k} == \sum_j bp_{j,k}$

- Les travailleurs ne peuvent pas être affecté à un projet qui n'a pas débuté. Ainsi,

  $\forall$ i,J,k,l $a_{i,J,k,l} <= \sum_j^J bp_{j,k}$

  L'utilisation de J et j transcrit le fait que $\sum_j^J bp_{j,k}$ a débuté ou non au temps J.

- Un projet ne peut être considère réalisé au jour j que si la quantité de travail fournie pour ce projet sur les différentes compétences est supérieur aux attentes pour le projet.

  Exemple : le projet 1 ne peut être considéré comme réalisé que s'il y a eu 2 jours de travail sur la compétence 1 et un jour la compétence 2 dédiés au projet.

  Pour implémenter cette contrainte, deux possibilités existent :

  $\forall$ k, $dp_{J,k} \Leftrightarrow (\forall$ l, $\sum_i \sum_{j=0}^J a_{i,j,k,l} >= pc_{k,l})$

  $\forall$ k,J,l $\sum_i \sum_{j=0}^J a_{i,j,k,l} >= pc_{k,l} * dp_{J,k}$

  L'utilisation d'une valeur J et j transcrit le besoin de calculer le travail cumulé depuis le début et non le travail à l'instant j. Nous pouvons distinguer deux différences majeures, la première formulation nécessite une implication et la seconde l'ignore. En effet, un projet peut avoir suffisamment de travail cumulé mais ne pas être réalisé. Cependant, ce cas ne présente aucun intérêt dans notre étude qui a pour objectif de maximiser le profit de l'entreprise. Nous avons donc choisi la deuxième implémentation de la contrainte, car l'implication est coûteuse en programmation linéaire. Une démonstration complexe de la quasi-équivalence des deux formulations est disponible en annexe.

2. Les contraintes liés aux transformations min max :

   - La durée de tous les projets est inférieur à d_{max} :

     $\forall k$, $\sum_{j=0}^h j * (dp_{j,\,k} - bp_{j,\,k}) + \sum_{j=0}^h (dp_{j,\,k}) <= d_{max}$

- Contrainte liée au nombre de projets affectés aux différents travailleurs (voir annexe) : $\forall m,\ \sum_k \delta_k^m <= cp_{max}$

  $\forall i,k,\ h * (1 - \delta_i^k) <= \sum_{j,l} a_{i,j,k,l}$

  $\forall i,k,\ \sum_{j,l} a_{i,j,k,l} < h * \delta_i^k$

2.4 Remarques

Dans cette section, nous allons rédiger un petit nombre de commentaires expliquant notre modélisation et ses spécificités. Nous avons fait le choix d'implémenter un modèle avec le minimum de contraintes possibles et de les simplifier le plus possibles. Premièrement, nous avons fait le choix de ne pas implémenter le fait que le démarrage d'un projet était lié à la première journée de travail dédiée. En effet, il est possible que la solution nous indique que le projet 1 a débuté le jour 1 mais que personne ne travaille dessus avant le jour 3. La seule condition est que la durée entre le rendu et le début est inférieur à d_{max}. De même, un projet peut être rendu après la complétion de tous ces demandes. Cependant, il nous est possible de calculer les réels dates de début et de rendus des projets après l'optimisation, cette étape est réalisé lors du postprocessing.

L'objectif de cette modélisation est donc de soulager le solveur en rajoutant un peu de postprocessing pour faciliter l'optimisation et son temps de calcul. Le postprocessing étant obligatoire pour présenter les données à l'utilisateur n'est pas problématique. De plus, il ne s'agit que de simples opérations matricielles donc peu coûteuses.

3 Optimisation et résultats - surface de Pareto

L'objectif final de la première partie du projet était de parvenir à représenter la surface de Pareto du problème pour les instances proposées dans les fichiers de configuration au format .json.

La section précédente a consisté à établir un modèle complet pour une instance quelconque du problème. Une remarque importante à ce stade est que pour lancer l'optimisation, il va falloir se concentrer sur une seule fonction-objectif. En effet, on se sait pas optimiser conjointement plusieurs fonctions-objectif, et il nous est donc impossible de gérer ensemble les

3 objectifs définis précédemment.

On va donc, par la suite, fixer la fonction-objectif qui décrit le nombre de projets abordés par le collaborateur le plus diversifié et celle qui décrit le temps mis pour compléter le projet qui aura duré le plus longtemps. On aura donc une seule fonction-objectif : celle qui concerne l'optimisation du bénéfice net de l'entreprise sur l'horizon de temps donné.

De plus, on a la chance que les deux fonctions-objectif que l'on a choisi de fixer prennent des valeurs entières (positives). On va donc pouvoir optimiser le problème du gain sur une grille qui est un sous-ensemble de $\mathcal{N}^{\in}$.

On va ainsi lancer au total $n_p * h$ optimisations, en faisant décroître l'une puis l'autre des valeurs entières que prennent les fonctions-objectif. Nous obtiendrons autant de valeurs optimales de gain, résultantes de l'optimisation du problème décrit dans la section précédente, auquel on a soustrait les contraintes relatives aux deux fonctions-objectif linéarisées (car elles ne sont pas intégrées comme objectifs dans cette partie) et auquel on a ajouté les contraintes suivantes :

$$z_{fo2} = \phi$$

$$z_{fo3} = \psi$$

$$\phi \in [1, n_p]$$

$$\psi \in [1, h]$$

$$\psi, \phi \in N$$

Toutefois, les triplets obtenus sont de la forme : $(gain_{opt}, \phi, \psi)$ et ne sont pas tous des points appartenant à la surface de Pareto ! Il faut encore filtrer les points obtenus pour ne garder que les points non-dominés. Dans les triplets, $gain_{opt}$ est une fonction de ϕ et ψ.

Le filtrage consiste à trouver, à gain fixe, les couples (ϕ, ψ) non-dominés. Par exemple, parmi les couples suivants :

(1, 4)

(2, 4)

(3, 3)

(5, 2)

On voit que selon la coordonnée 1, la valeur de 1 donne le couple $(1, 4)$, la valeur de 2 donnerait le couple $(2, 4)$ mais il dominé par le couple précédent, donc n'est pas retenu. La valeur de 3 donne $(3, 3)$ et la valeur de 5 donne $(5, 2)$. On applique finalement la méthode ϵ-constraint sur la grille ainsi formée (forme restrictive et simplifiée de l'algorithme vu en cours).

On a ainsi une liste de points qui permettent d'échantillonner la surface de Pareto du problème. En annexe, nous affichons les courbes (surfaces vivant dans un cube 3D) obtenues.

4 Modèle de préférences

La section précédente nous a permis de trouver l'ensemble des solutions non dominées du problème grâce à l'étude de la surface de Pareto couplé à l'algorithme d'ϵ- constraint. Par définition, il n'existe aucune solution dominant largement les solutions appartenant à l'ensemble non-dominé. Chacun des éléments de cet ensemble représente donc une solution optimale au problème d'optimisation et il est ainsi impossible, sans hypothèses supplémentaires, de les comparer entre eux. Pour rappel, la problématique du projet est de trouver l'allocation de ressources permettant de maximiser le bénéfice net de l'entreprise sur un horizon de temps fixé. Il est ainsi indispensable d'effectuer une étude de préférences afin de choisir quelle allocation de ressources sera mise en place par l'entreprise.

Dans cette section on propose d'ordonner nos préférences à l'aide du fonction de score S. Pour cela, nous définissons les fonctions s_i et S ainsi:

$$\begin{aligned} s_i \colon R^n &\longrightarrow R \\ x &\longmapsto x_i \\ S \colon R^n &\longrightarrow R \\ x &\longmapsto \sum_{i=1}^{n} w_i x_i \end{aligned}$$

Avec $w \in R^n$ et $\forall i \ w_i \geq 0$ et $\sum_{i=1}^{n} w_i = 1$.

Ainsi pour chacun des triplets x de l'ensemble non-dominé, on obtient le score suivant:

$$S(x) = w_1 s_1(x) + w_2 s_2(x) + w_3 s_3(x)$$

Pour pouvoir comparer les résultats des scores nous avons au préalable normalisé les composantes des triplets solutions. Ainsi, en notant $gain_{opt}$ la somme des gains de tous les projets, h l'horizon du problème et d_{max} la durée du projet le plus long, on obtient pour le vecteur x après normalisation le vecteur suivant:

$$x = \begin{bmatrix} 1 - \frac{x_1}{gain_{opt}} \\ \frac{x_2}{h} \\ \frac{x_3}{d_{max}} \end{bmatrix}$$

Cette normalisation permet aussi d'avoir comme objectif de minimiser la fonction S.
Pour effectuer des préférences sur l'ensemble non-dominé, nous décidons de classer les solutions dans trois ensembles D_i avec $i \in [\![1 \ ; \ 3]\!]$. L'ensemble D_1 représente l'ensemble des solutions qui rapporte un gain d'au moins $0.85 gain_{opt}$, l'ensemble D_2 contient les solutions rapportant au moins $0.5 gain_{opt}$ et l'ensemble D_3 celles rapportant moins de $0.5 gain_{opt}$. Nous considérons alors l'ensemble des solutions de l'ensemble D_1 comme au moins aussi attractives que les solutions de l'ensemble D_2, et celles de l'ensemble D_2 comme au moins aussi attractives que les solutions de l'ensemble D_3. De cette classification, nous pouvons en déduire les contraintes suivant sur le vecteur de poids w:

$$\forall x \in D_1 \text{ et } \forall y \in D_2,$$
$$S(x) \leq S(y)$$
$$\forall y \in D_2 \text{ et } \forall z \in D_3,$$
$$S(y) \leq S(z)$$

Pour la Toy instance, ces contraintes permettent de définir l'espace des poids représenter

ci-dessous:

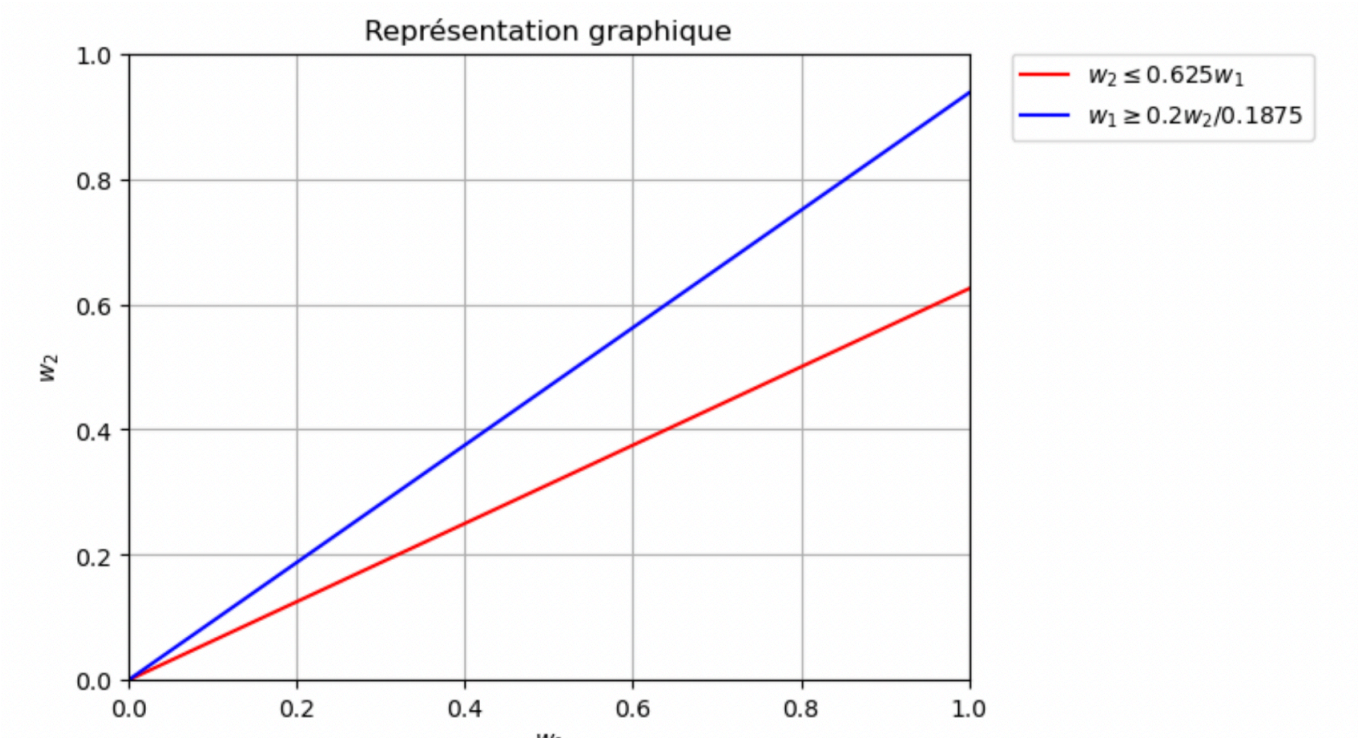

Figure 1: Représentation de l'espace des poids

Pour comparer ensuite deux modèles entre eux, nous avons décidé d'implémenter les deux programmes suivants:

$$Prog1 : \max_{\{w\}}(S(x) - S(y))$$

$$Prog2 : \max_{\{w\}}(S(y) - S(x))$$

Ainsi, pour la Toy instance on obtient les préférences suivantes:

$$Planning_2 \sim Planning_3 \sim Planning_4$$

$$et$$

$$Planning_1 \succeq Planning_4$$

La meilleure allocation de ressources pour la Toy instance est donc représentée par la solution $Planning_1$:

$$Planning_1 = \begin{bmatrix} 0.1875 \\ 0.8 \\ 0.2 \end{bmatrix}$$

Ce qui représente un gain de 65€, un horizon de 4 jours et une durée maximale de projet de 1 jour.

Malgré que le modèle soit relativement simpliste, on se rend compte que la réponse fournie par l'algorithme représente la seule solution appartenant à l'ensemble D_1; ce qui est donc cohérent avec nos attentes et objectifs.

5 Discussion et conclusion

Au cours de ce projet, nous avons modélisé un problème relativement complexe (3 fonctions-objectif et une dizaine de contraintes) à l'aide d'un modèle mathématique de notre conception. Nous avons ensuite utilisé les techniques apprises en cours pour linéariser certaines fonctions-objectif initialement non-linéaires, pour ainsi obtenir un programme linéaire. Il a ensuite été possible de programmer le problème à l'aide de Python et la libraire Gurobi (solveur de programmation linéaire). La résolution du problème nous a permis de tracer la surface de Pareto du problème pour une instance donnée.

Pour choisir parmi les solutions non-dominées du problème (les points de la surface de Pareto tracées précédemment), nous avons établi un modèle de préférence. Nous avons ainsi fixé arbitrairement une préférence (fraction minimale du gain total à obtenir), reflétant la préférence d'une personne fictive (en l'occurence, Margaux). Cette nouvelle contrainte nous a permis d'établir un ordre partiel parmi les triplets solutions, et en particulier d'identifier un triplet meilleur que les autres (selon la relation de préférence large). D'autres modèles de préférence sont possibles, nous avons ici simplement souhaité en illustrer un.

A Annexe

A.1 Critère d'optimisation 2

Le second objectif est minimiser le nombre de projets différents sur lesquels sont affectés les collaborateurs de l'entreprise. On peut donc l'exprimer ainsi :

$$\min \max_i, \sum_k^h 1_{i,k}$$

avec $1_{i,k}$ qui vaut 1 si le travailleur i a travaillé sur le projet k, 0 sinon.

$\sum_{j=1}^{k} \sum_{l=n_a}^{k} a_{i,j,k,l}$ représente le nombre de jours (toutes compétences confondues) que le collaborateur i a passé sur le projet k. Si le collaborateur i a travaillé sur le projet k, alors cette quantité est > 0.

Ainsi, $1_{1,k}$ vaut 1 si et seulement si $\sum_{j=1}^{k} \sum_{l=n_a}^{k} a_{i,j,k,l} > 0$.

D'où la représentation de l'optimisation sous la forme min $\max_i, \sum_k^h 1_{i,k}$. ,

Il faut ensuite linéariser l'objectif, c'est à dire :

$$\text{On cherche à imposer } M^*(1\text{-}\delta) <= (x - x_0) < M^*\delta$$

$$\text{avec } 1_{i,k} = \delta = \delta_{i,k},$$

$$\text{avec } x = x_{i,k} = \sum_{j=1}^{k} \sum_{l=n_a}^{k} a_{i,j,k,l},$$

$$\text{avec } x_0 = 0.$$

Dans notre cas, M = $M_{i,k}$ doit majorer $x_{i,k}$. Il nous suffit donc de prendre M=h, notre horizon. Ainsi dans un premier temps, notre objectif peut s'exprimer ainsi :

$$\min \max_i, \sum_k^h d\delta_{i,k}, \text{ sous la contrainte suivante,}$$

$$h * (1 - \delta) <= \sum_{j=1}^{k} \sum_{l=n_a}^{k} a_{i,j,k,l} < h * \delta \ .$$

Deuxièmement, pour linéariser l'objectif, il faut le mettre sous la forme :

$$\begin{aligned} \min \quad & \max(\sum_{j=1}^{n} c_j^1 * x_j, ..., \sum_{j=1}^{n} c_j^p * x_j) \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{i,j} * x_j >= b_i, \\ & x >= 0 \end{aligned}$$

On identifie donc :

$\sum_{j=1}^{n} c_j^1 * x_j = \sum_{j=1}^{n_p} \delta_{i,k} = \sum_{j=1}^{n_p} \delta_i^k$

D'où :

$$\begin{aligned} \min \quad & cp_{max} \\ \text{s.t.} \quad & \forall m, \sum_k \delta_k^m <= cp_{max}, \\ & \forall i, k, h * (1 - \delta_i^k) <= \sum_{j,l} a_{i,j,k,l}, \\ & \forall i, k, \sum_{j,l} a_{i,j,k,l} < h * \delta_i^k \end{aligned}$$

A.2 Critère d'optimisation 3

Le troisième objectif est minimiser les temps de validation des projets. On peut donc l'exprimer ainsi :

$$\min \max_k, \sum_{j=0}^{h} j * (dp_{j,\,k} - bp_{j,\,k}) + \sum_{j=0}^{h} (dp_{j,\,k})$$

On identifie cette fois :

- $n = h$,
- $c_j^p = (dp_{j,\,k} - bp_{j,\,k})$,
- $x_j = j$

D'où :

$$\begin{aligned} \min \quad & d_{max} \\ \text{s.t.} \quad & \forall k, \sum_{j=0}^{h} j * (dp_{j,\,k} - bp_{j,\,k}) + \sum_{j=0}^{h} (dp_{j,\,k}) <= d_{max} \end{aligned}$$

A.3 Équivalence des contraintes

Nous allons démontrer :

$\forall$ k, $dp_{J,k} \Leftrightarrow (\forall$ l, $\sum_i \sum_{j=0}^{J} a_{i,j,k,l} >= pc_{k,l})$ est pseudo-équivalent à $\forall$ k,J,l $\sum_i \sum_{j=0}^{J} a_{i,j,k,l} >= pc_{k,l} * dp_{J,k}$.

En effet,

- Si $dp_{J,k} = 0$, $\sum_i \sum_{j=0}^{J} a_{i,j,k,l} >= pc_{k,l}$ n'est pas contraint. Or $a_{i,j,k,l}$ vaut 0 ou 1.
  Donc $\sum_i \sum_{j=0}^{J} a_{i,j,k,l} >= 0$
  Donc $\sum_i \sum_{j=0}^{J} a_{i,j,k,l} >= pc_{k,l} * dp_{J,k}$

- Si $dp_{J,k} = 1$, $\sum_i \sum_{j=0}^{J} a_{i,j,k,l} >= pc_{k,l}$. Or $dp_{J,k} = 1$,.
  Donc $\sum_i \sum_{j=0}^{J} a_{i,j,k,l} >= pc_{k,l} * 1$
  Donc $\sum_i \sum_{j=0}^{J} a_{i,j,k,l} >= pc_{k,l} * dp_{J,k}$

Ainsi, on peut dire que $\forall$ k, $dp_{J,k} \Rightarrow (\forall$ l, $\sum_i \sum_{j=0}^{J} a_{i,j,k,l} >= pc_{k,l})$ est équivalent à contraindre $\forall$ k,J,l $\sum_i \sum_{j=0}^{J} a_{i,j,k,l} >= pc_{k,l} * dp_{J,k}$.

Il manque uniquement le sens suivant :

- Si $\sum_i \sum_{j=0}^{J} a_{i,j,k,l} >= pc_{k,l} * dp_{J,k}$, alors $dp_{J,k} = 1$.

Or, notre problème s'intéresse à une optimisation du gain. Ainsi, si $\sum_i \sum_{j=0}^{J} a_{i,j,k,l} >= pc_{k,l} * dp_{J,k}$, la solution $dp_{J,k} = 1$ est meilleur que $dp_{J,k} = 0$. Ce sens est donc entraîné par l'optimisation, d'où la pseudo équivalence.

A.4 Plots des surfaces de Pareto

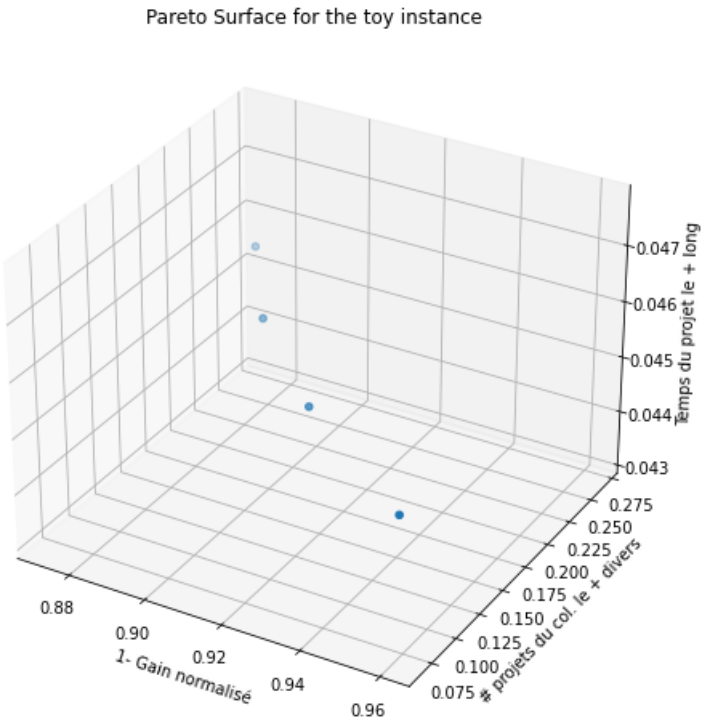

Figure 2: Points de la surface de Pareto sur l'instance Toy

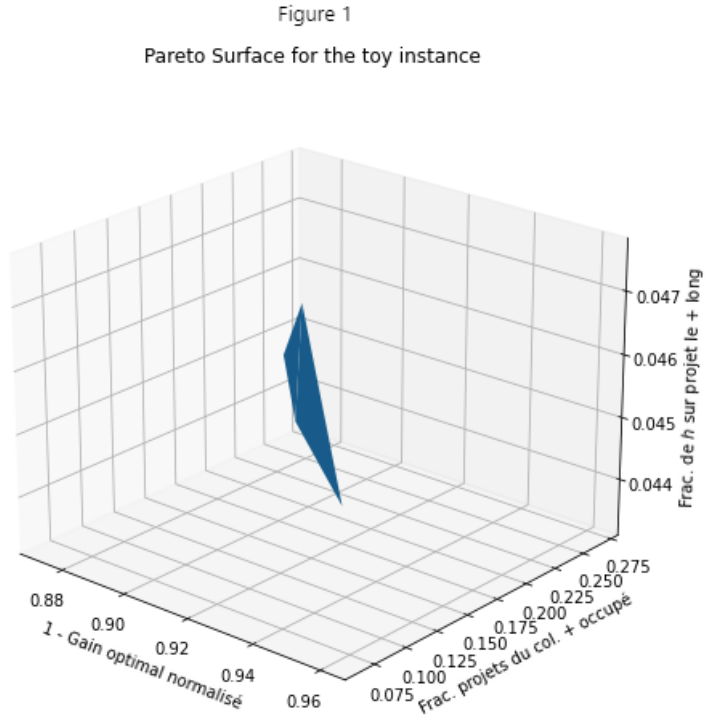

Figure 3: Surface de Pareto sur l'instance Toy

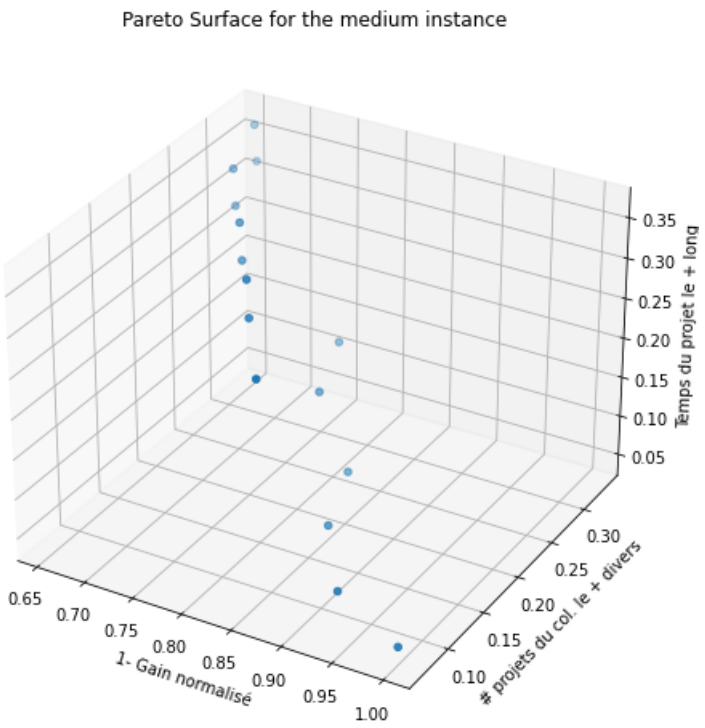

Figure 4: Points de la surface de Pareto sur l'instance Medium

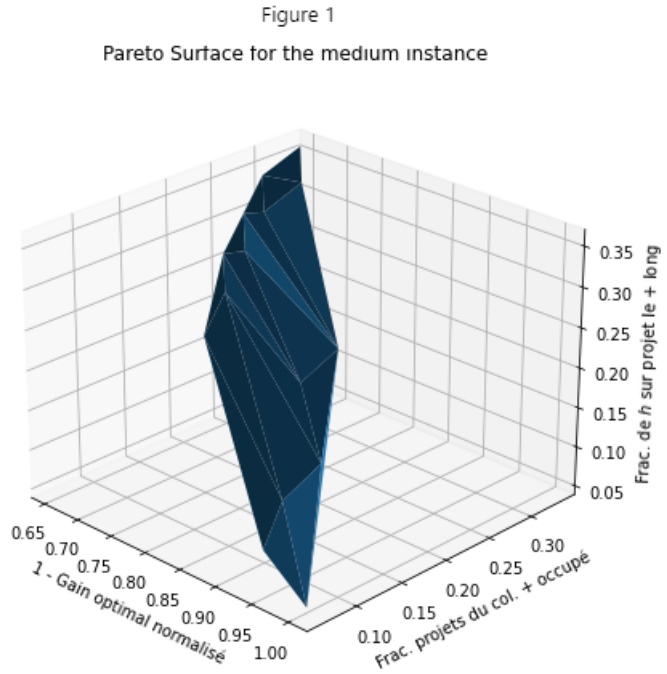

Figure 5: Surface de Pareto sur l'instance Medium

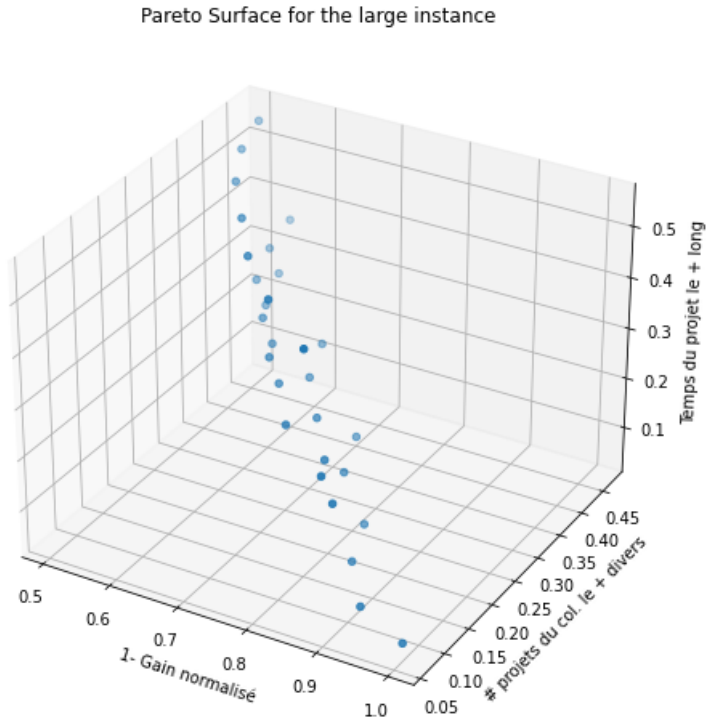

Figure 6: Points de la surface de Pareto sur l'instance Large

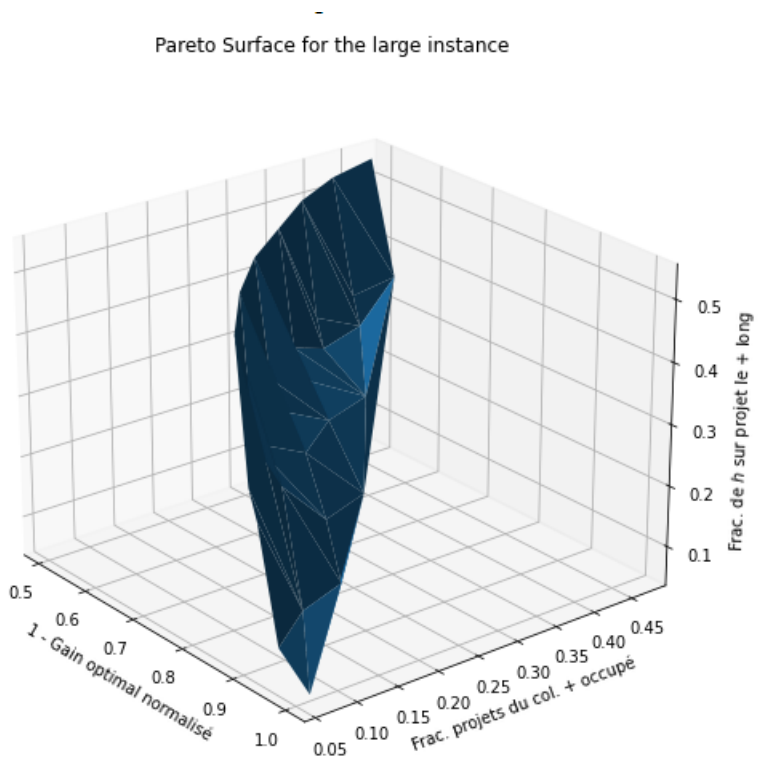

Figure 7: Surface de Pareto sur l'instance Large