MOGPL
Projet : L'optimisation équitable

Dang-Nhu Barthélémy, Delefosse Aymeric

04 Décembre 2022

L'équité est un concept complexe qui se retrouve dans de nombreux domaines : droit, philosophie, économie (théorie des jeux), sciences (*machine learning*), etc. D'après le Larousse, l'équité est la « qualité consistant à attribuer à chacun ce qui lui est dû par référence aux principes de la justice naturelle », expliquant l'association de ce terme avec celui d'égalité.

Le but de ce travail est d'étudier le concept d'équité dans le domaine de la recherche opérationnelle et de l'optimisation. Nous nous intéresserons à résoudre des problèmes de décision et à trouver des solutions efficaces équitables. Pour cela, nous utiliserons la fonction objectif $f(x) = \sum_i w_i z_{(i)}(x)$, où $z_{(i)}(x)$ est la i-ième satisfaction la plus basse et w_i des poids décroissants.

Ce rapport est fourni avec un fichier `README.md` détaillant le fonctionnement du programme ayant servi à la résolution des différents problèmes énoncés au long de ce rapport.

1 Linéarisation de f

En optimisation équitable, on s'intéresse par exemple, à comment répartir des objets entre plusieurs agents de façon à réduire les inégalités.

L'exemple suivant est une illustration de ce problème : comment sélectionner un ensemble de trois objets parmi 5 de manière à satisfaire équitablement deux agents, sachant que chaque agent se fait sa propre idée de l'utilité des objets pour lui ? Ici, les utilités des objets sont respectivement $(5, 6, 4, 8, 1)$ et $(3, 8, 6, 2, 5)$ et le vecteur de pondération est $w = (2, 1)$. Le problème d'optimisation s'écrit alors :

$$\max 2z_{(1)} + z_{(2)}$$

$$s.c. \begin{cases} z_1 = 5x_1 + 6x_2 + 4x_3 + 8x_4 + x_5 \\ z_2 = 3x_1 + 8x_2 + 6x_3 + 2x_4 + 5x_5 \\ x_1 + x_2 + x_3 + x_4 + x_5 = 3 \end{cases}$$

$$x_i \in \{0, 1\},\ i = 1, \ldots, 5$$

Cependant, ce n'est pas un programme linéaire, et il est donc impossible de le résoudre à l'aide de techniques de programmation linéaire. Dans cette partie, nous verrons comment linéariser ce problème.

1.1 PL en variables 0-1

Soit le programme linéaire suivant, n représentant le nombre d'agents :

$$\min \sum_{i=1}^{n} a_{ik} z_i$$

$$s.c. \sum_{i=1}^{n} a_{ik} = k$$

$$a_{ik} \in \{0, 1\},\ i = 1, \ldots, n$$

Ce programme linéaire revient à minimiser la somme de k valeurs parmi les z_i. Comme, par construction, on a $z_{(i)} \leq z_{(i+1)}$, $i = 1, \ldots, n-1$, il faut naturellement choisir les k premières valeurs, c'est-à-dire : $z_{(1)}$, $z_{(2)}$, $\ldots$, $z_{(k)}$. La solution optimale de ce problème est donc a_k^* tel que :

$$\begin{cases} a_{ik}^* = 1 \text{ si } z_i \in \{z_{(1)}, z_{(2)}, \ldots, z_{(k)}\} \\ a_{ik}^* = 0 \text{ sinon} \end{cases}$$

La valeur optimale de ce problème est alors :

$$\sum_{i=1}^{k} z_{(i)} = L_k(z)$$

1.2 PL en variables continues

En admettant que le programme mathématique précédent peut être relaxé en variables continues, qui admet la même solution, il peut alors s'écrire sous la forme :

$$\min \sum_{i=1}^{n} a_{ik} z_i$$

$$s.c. \begin{cases} a_{1k} + a_{2k} + \quad \cdots + a_{nk} & = k \\ -a_{1k} & \geq -1 \\ \quad - a_{2k} & \geq -1 \\ \qquad\qquad \vdots & \\ \qquad\qquad - a_{nk} & \geq -1 \end{cases}$$

$$a_{ik} \geq 0,\ i = 1, \ldots, n$$

En notant r_k la variable duale liée à la première contrainte et b_{ik} les variables duales associées aux contraintes $-a_{ik} \geq -1$, $i = 1, \ldots, n$, son dual D_k s'écrit alors :

$$\max k\, r_k + \sum_{i=1}^{n} -1 \times b_{ik}$$

$$s.c. \begin{cases} r_k - b_{1k} & & \leq z_1 \\ r_k & -b_{2k} & \leq z_2 \\ & \vdots & \\ r_k & - b_{nk} & \leq z_n \end{cases}$$

$$r_k \in \mathbb{R},\ b_{ik} \geq 0,\ i = 1, \ldots, n$$

Ainsi, de manière générale, D_k s'écrit :

$$\max k\, r_k - \sum_{i=1}^{n} b_{ik}$$

$$s.c.\ r_k - b_{ik} \leq z_i$$

$$r_k \in \mathbb{R},\ b_{ik} \geq 0,\ i = 1, \ldots, n$$

Utilisons cette formulation duale pour calculer par programmation linéaire les composantes du vecteur $L(4, 7, 1, 3, 9, 2)$. On trouve bien $(1, 3, 6, 10, 17, 26)$.

1.3 Lien entre $f(x)$ et $L_k(z(x))$

Soit $w'_k = (w_k - w_{k+1})$, pour $k = 1, \ldots, n-1$ et $w'_n = w_n$ on a :

$$\begin{aligned}
\sum_{k=1}^{n} w'_k L_k(z(x)) &= \sum_{k=1}^{n-1} (w_k - w_{k+1}) L_k(z(x)) + w_n L_n(z(x)) \\
&= \sum_{k=1}^{n-1} w_k L_k(z(x)) - \sum_{k=1}^{n-1} w_{k+1} L_k(z(x)) + w_n L_n(z(x)) \\
&= \sum_{k=1}^{n-1} w_k L_k(z(x)) - \sum_{k=2}^{n} w_k L_{k-1}(z(x)) + w_n L_n(z(x)) \\
&= w_1 L_1(z(x)) + \sum_{k=2}^{n-1} w_k [L_k(z(x)) - L_{k-1}(z(x))] + -w_n L_{n-1}(z(x)) + w_n L_n(z(x)) \\
&= w_1 z_{(1)}(x) + \sum_{k=2}^{n-1} w_k [L_k(z(x)) - L_{k-1}(z(x))] + w_n [L_n(z(x)) - L_{n-1}(z(x))]
\end{aligned}$$

Mais, comme $L_k(z(x)) - L_{k-1}(z(x)) = \sum_{i=1}^{k} z_{(i)}(x) - \sum_{i=1}^{k-1} z_{(i)}(x) = z_{(k)}(x)$ par téléscopage :

$$\begin{aligned}
\sum_{k=1}^{n} w'_k L_k(z(x)) &= w_1 z_{(1)}(x) + \sum_{k=2}^{n-1} w_k z_{(k)}(x) + w_n z_{(n)}(x) \\
&= \sum_{k=1}^{n} w_k z_{(k)}(x) \\
&= f(x)
\end{aligned}$$

1.4 Reformulation du problème

On cherche à maximiser $f(x)$ pour $x \in X$. Donc, d'après la question précédente, on cherche à maximiser $\sum_{k=1}^{n} w'_k L_k(z(x))$ pour $x \in X$. Or, d'après les questions 1.1 et 1.2, $L_k(z(x))$ est solution du programme linéaire énoncé à la question 1.2. Donc, d'après le Théorème des Écarts Complémentaires (TEC), $L_k(z(x))$ est également solution du dual D_k qui est le maximum de $k\, r_k - \sum_{i=1}^{n} b_{ik}$, avec $r_k - b_{ik} \leq z_i$, $i = 1, \ldots, n$, $r_k \in \mathbb{R}$ et $b_{ik} \geq 0$. Finalement, maximiser $f(x)$ revient donc à résoudre :

$$\max \sum_{k=1}^{n} w'_k \left(k\, r_k - \sum_{i=1}^{n} b_{ik} \right)$$

$$s.c. \begin{cases} r_k - b_{ik} \leq z_i(x),\ i = 1, \ldots, n \\ x \in X \end{cases}$$

$$r_k \in \mathbb{R},\ b_{ik} \geq 0$$

En effet, pour maximiser la fonction objectif, il faut maximiser les $kr_k - \sum_{i=1}^{n} b_{ik}$ (car $w'_k > 0$). Or, chaque maximisation de ces n éléments se font indépendamment, car les contraintes sur les r_k et b_{ik} sont indépendantes par rapport à k. On retrouve donc nécessairement $L_k(z(x)) = k\, r_k - \sum_{i=1}^{n} b_{ik}$.

Utilisons maintenant ce résultat pour linéariser notre exemple introduit au début de cette section et calculer sa solution optimale. Nous trouvons qu'il faut prendre les objets 2, 3 et 4, cela donne un vecteur de satisfaction $z = (18, 16)$.

2 Application au partage équitable de biens indivisibles

On souhaite répartir p objets (biens, ressources, tâches...) entre n individus de manière équitable. On notera u_{ij} l'utilité de l'objet j pour l'agent i et x_{ij} qui vaut 1 si l'agent i reçoit l'objet j et 0 sinon, pour tout $i = 1, \ldots, n$ et pour tout $j = 1, \ldots, p$.

2.1 PL en variables mixtes

On souhaite répartir de manière équitable les p objets, notre problème se modélise avec le programme suivant :

$$\max f(x) = \sum_{i=1}^{n} w_i z_{(i)}(x)$$

$$s.c. \begin{cases} z_i = \sum_{j=1}^{p} u_{ij}\, x_{ij} \\ \sum_{i=1}^{n} x_{ij} \leq 1 \end{cases}$$

$$x_{ij} \in \{0,1\},\ i \in [\![1,n]\!],\ j \in [\![1,p]\!]$$

La première contrainte correspond à l'additivité des utilités sur les objets.
La deuxième contrainte permet d'assurer qu'un objet j ne peut être affecté à plus d'un agent.

On peut réecrire ce programme linéairement à l'aide de la partie 1, on obtient :

$$\max \sum_{k=1}^{n} w'_k \left(k\, r_k - \sum_{i=1}^{n} b_{ik} \right)$$

$$s.c. \begin{cases} z_i = \sum_{j=1}^{p} u_{ij}\, x_{ij} \\ r_k - b_{ik} \leq z_i \\ \sum_{i=1}^{n} x_{ij} \leq 1 \end{cases}$$

$$r_k \in \mathbb{R},\ b_{ik} \geq 0, x_{ij} \in \{0,1\},\ i \in [\![1,n]\!],\ j \in [\![1,p]\!]$$

On a n variables z_i, $n \times p$ variables x_{ij}, n variables r_k et $n \times n$ variables b_{ik}. On a donc au total $n^2 + 2n + np$ variables de décision pour $n^2 + n + p$ contraintes. À cela, il faut compter les $n \times p$ utilités u_{ij}.

On s'intéresse maintenant au problème de partage équitable de l'article, où nous avons 3 individus pour 6 objets. On rappelle que dans l'article, les utilités u_{ij} ne dépendent pas de i et sont les valeurs objectives de l'objet, on a : $u_j = \big(325,\ \ 225,\ \ 210,\ \ 115,\ \ 75,\ \ 50\big)$ les valeurs respectives des objets. Le problème est donc le suivant :

$$\max \sum_{k=1}^{n} w'_k \left(k\, r_k - \sum_{i=1}^{n} b_{ik} \right)$$

$$s.c. \begin{cases} z_1 = \sum_{j=1}^{6} u_j\, x_{1j} \\ z_2 = \sum_{j=1}^{6} u_j\, x_{2j} \\ z_3 = \sum_{j=1}^{6} u_j\, x_{3j} \\ r_k - b_{ik} \leq z_i,\ i = 1,2,3 \\ x_{11} + x_{21} + x_{31} \leq 1 \\ x_{12} + x_{22} + x_{32} \leq 1 \\ x_{13} + x_{23} + x_{33} \leq 1 \\ x_{14} + x_{24} + x_{34} \leq 1 \\ x_{15} + x_{25} + x_{35} \leq 1 \\ x_{16} + x_{26} + x_{36} \leq 1 \end{cases}$$

$$r_k \in \mathbb{R},\ b_{ik} \geq 0, x_{ij} \in \{0,1\},\ i = 1,2,3,\ j = 1,\ldots,6$$

Avec le vecteur poids $w = (3,2,1)$, on a $w' = (1,1,1)$. On trouve comme solution optimale :

— L'agent 1 prend les objets 3, 5 et 6 ;

— L'agent 2 prend l'objet 1 ;

— L'agent 3 prend les objets 2 et 4.

Le vecteur de satisfaction est alors $z = (335, 325, 340)$. On remarque que les objets pris peuvent être interchangés entre les agents, étant donné que nos 3 individus apportent la même utilité à chacun des objets.

Avec le vecteur de poids $w = (10, 3, 1)$, $w' = (7, 2, 1)$ et on trouve le même résultat à notre programme linéaire. Le résultat trouvé étant déjà très équitable, il est difficile d'en trouver un plus équitable. En effet, le problème étant à somme constante, on veut s'approcher le plus possible de l'égalité et nous y sommes déjà très proche.

Regardons ce qu'il se passerait si on voulait maximiser la satisfaction moyenne $\frac{1}{n}\sum_i z_i(x)$. Pour trouver la solution, il suffit d'appliquer le programme précédent à $w = (\frac{1}{n}, \cdots, \frac{1}{n})$. On obtient alors $w' = (0, 0, \cdots, 0, \frac{1}{n})$ et la fonction objectif devient :

$$f(x) = \frac{1}{n} L_n(z) = \frac{1}{n} \sum_{i=1}^{n} z_{(i)}(x) = \frac{1}{n} \sum_{i=1}^{n} z_i(x)$$

Dans notre problème, on obtient le vecteur de satisfaction $(0, 760, 240)$, ce qui n'est effectivement pas très équitable... La répartition est la suivante :

— L'agent 1 ne prend aucun objet ;

— L'agent 2 prend les objets 1, 2 et 3 ;

— L'agent 3 prend les objets 4, 5 et 6.

En réalité, si on veut la satisfaction moyenne maximale de cet exemple, on a $f(x) = (z_1(x) + z_2(x) + z_3(x))/3 = \sum_{j=1}^{p} u_j/3 = 1000/3$ car chaque individu porte la même valeur aux objets. La fonction objectif est donc en réalité indépendante de x : toute répartition des biens est optimale.

2.2 Évolution du temps de résolution en fonction de n et p

Observons l'influence de n et p sur le temps de résolution du programme. Le modèle a été implémenté à l'aide du solveur Gurobi 10.0 sur un ordinateur de 16 GB de RAM, un processeur Intel(R) Core(TM) i7-8086k @ 4.80GHz. Pour chaque couple (n, p) on tire aléatoirement 10 matrices de satisfaction U et vecteur de poids W, on fait ensuite la moyenne des temps avec $p = 5n$. En prenant $n > 11$, il devient très long de trouver la solution optimale dans un temps "raisonnable". En se contentant de $n \in [\![2, 11]\!]$, on trouve le résultat suivant :

(n,p)	(2,10)	(3,15)	(4,20)	(5,25)	(6,30)	(7,35)	(8,40)	(9,45)	(10,50)	(11,55)
temps (s)	0.0054	0.0110	0.0414	0.0631	0.1274	0.7549	2.6914	6.5806	22.2558	60.3516

TABLE 1 – Temps (s) moyens obtenus par le programme linéaire

En effectuant une régression exponentielle, on trouve un coefficient de corrélation $R = 0.9925$ avec $t = 0.0004 \times 2.898^n$. On peut douter de l'exactitude de cette régression du fait du faible nombre de points, mais cela permet de faire une première approximation pour de faibles valeurs de n. Si l'on considère le modèle exact, il faudrait en moyenne 8 jours avec $n = 20$ pour résoudre le problème et presque 1 heure lorsque $n = 15$. On obtient le graphique suivant :

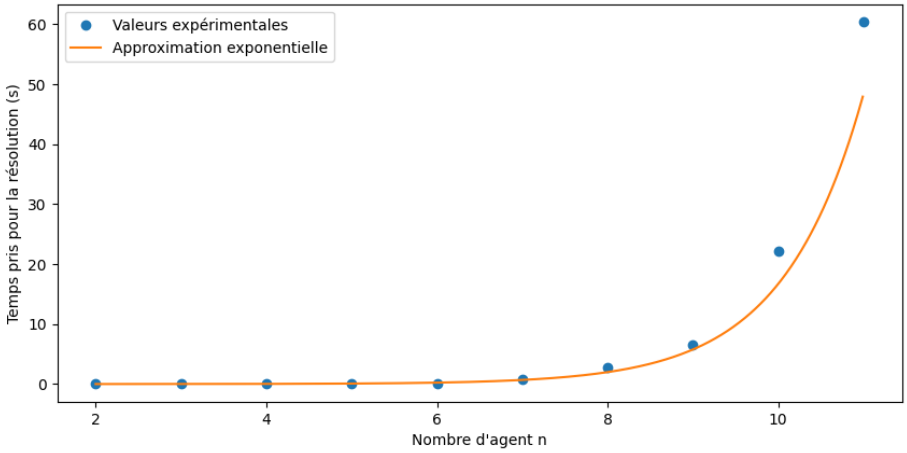


FIGURE 1 – Evolution du temps de résolution du programme linéaire du partage équitable en biens indivisibles en fonction de $n = 2, \ldots, 11$, $p = 5n$

Pour des valeurs plus grandes, le problème est causé par la grande variance du temps de résolution. En effet pour $(n, p) = (20, 100)$ une solution peut être trouvée en une minute comme en plus de 24 heures. Pour remédier à ce problème, nous avons refait l'expérience en fixant un délai d'attente maximal (*time-out*) à 300 secondes (contrairement aux résultats précédents). La moyenne trouvée est donc inférieure à la réalité, car nous ne prenons pas en compte les itérations ayant dépassé le délai d'attente. Cependant, nous pouvons tout de même calculer la vraie médiane en prenant en compte les instances expirées, car moins de la moitié le sont. Les résultats sont présentés ci-dessous en secondes.

(n,p)	(5,25)	(10,50)	(15,75)	(20,100)
temps moyen (avec *time-out*)	0.0765	13.681	26.0034	85.067
temps médian (sans *time-out*)	0.0585	11.491	19.374	238.356

TABLE 2 – Temps (s) obtenus par le programme linéaire

Le problème est d'autant plus complexe lorsque les différents agents accordent tous des utilités différentes aux différents objets, comme c'est le cas ici, les u_{ij} étant tous tirés aléatoirement. Ainsi, le temps de résolution semble être lié au tirage aléatoire du vecteur U : plus les utilités peuvent prendre des valeurs dispersées et grandes, plus le temps de résolution sera long.

3 Application à la selection multicritère de projets

On souhaite sélectionnner un nombre de projets parmi une liste de p projets soumis en fonction n objectifs à satisfaire. Chaque projet k a un coût c_k et une utilité additive pour l'objectif i, notée u_{ik}. La seule contrainte est de ne pas dépasser le budget total b. On souhaite trouver une solution équilibrée entre la satisfaction des différents objectifs.

3.1 PL en variables mixtes

On se fixe dans le problème une l'enveloppe budgétaire fixée à $b = \frac{1}{2}\sum_{k=1}^{p} c_k$, x_k qui vaut 1 si ce projet est sélectionné et 0 sinon. Alors, notre problème se modélise avec le programme suivant :

$$\max f(x) = \sum_{i=1}^{n} w_i z_{(i)}(x)$$

$$s.c. \begin{cases} z_i = \sum_{k=1}^{p} u_{ik}\, x_k \\ \sum_{k=1}^{p} c_k x_k \leq b \end{cases}$$

$$x_k \in \{0, 1\},\ i = 1, \ldots, n,\ k = 1, \ldots, p$$

On s'intéresse maintenant à l'instance donnée dans l'article concernant un problème bi-objectif de sélections de projets parmi 4 projets sous une contrainte de budget $b = (40 + 50 + 60 + 50)/2 = 100$ k€, soit :

$$\max f(x) = w_1 z_{(1)} + w_2 z_{(2)}$$

$$s.c. \begin{cases} z_1 = 19x_1 + 6x_2 + 17x_3 + 2x_4 \\ z_2 = 2x_1 + 11x_2 + 4x_3 + 18x_4 \\ 40x_1 + 50x_2 + 60x_3 + 50x_4 \leq 100 \end{cases}$$

$$x_k \in \{0, 1\},\ i = 1, 2,\ k = 1, 2, 3, 4$$

Comme précédemment, on se sert de la linéarisation de $f(x)$ pour résoudre notre problème, ce qui donne :

$$\max \sum_{k=1}^{n} w'_k \left(k\, r_k - \sum_{i=1}^{n} b_{ik} \right)$$

$$s.c. \begin{cases} r_k - b_{ik} \leq z_i,\ i = 1, 2 \\ z_1 = 19x_1 + 6x_2 + 17x_3 + 2x_4 \\ z_2 = 2x_1 + 11x_2 + 4x_3 + 18x_4 \\ 40x_1 + 50x_2 + 60x_3 + 50x_4 \leq 100 \end{cases}$$

$$r_k \in \mathbb{R},\ b_{ik} \geq 0, x_j \in \{0, 1\},\ k = 1, 2, 3, 4$$

On a n variables z_i, p variables x_k, n variables r_k et $n \times n$ variables b_{ik}. On a donc au total $n^2 + 2n + p$ variables de décision pour $n^2 + n + 1$ contraintes. À cela, il y a également $n \times p$ utilités u_{ik} et p coûts c_k.

Ainsi, avec le vecteur poids $w = (2, 1)$, la solution optimale est $x = (1, 0, 0, 1)$ et le vecteur d'utilité est $z = (21, 20)$ (ce qui est bien équilibré). Ce résultat correspond concrètement à financer le projet 1 et 4.

Avec le vecteur poids $w = (10, 1)$, la solution optimale est également $x = (1, 0, 0, 1)$ et le vecteur d'utilité est à nouveau $z = (21, 20)$. On obtient le même résultat que précédemment : les poids n'ont pas influencé la solution équitable de notre problème.

Observons le cas où l'objectif est la satisfaction moyenne, c'est-à-dire $(z1 + z2)/2$, pour cela comme précédemment nous allons prendre $w = (1/n, \ldots, 1/n)$. La solution optimale est alors $x = (1, 0, 1, 0)$ (financement du projet 1 et 3) et le vecteur d'utilité devient $z = (36, 6)$, ce qui n'est pas très équilibré... ni équitable, mais la satisfaction globale a en effet augmenté, passant de $20,5$ à 21.

3.2 Évolution du temps de résolution de la sélection multicritère de projets en fonction de n et p

Nous allons maintenant étudier l'évolution du temps de résolution de ce programme linéaire en fonction de $n = (2, 5, 10, 15, 20)$ objectifs et $p = (5, 10, 15, 20, 50, 100)$ projets. On tire aléatoirement 10 matrices U de taille (n, p) à coefficients entiers positifs correspondant aux utilités, ainsi qu'un vecteur C de taille p correspondant aux coûts des différents projets. On rappelle que notre enveloppe budgétaire est fixée par $\frac{1}{2} \sum_{k=1}^{p} c_k$.

Le modèle a été implémenté à l'aide du solveur Gurobi 10.0 sur un ordinateur de 16 GB de RAM, un processeur Intel(R) Core(TM) i7-8086k @ 4.80GHz. Le temps moyen en millisecondes pour chaque paire (n, p) est indiqué dans le tableau suivant.

p	n=2	n=5	n=10	n=15	n=20
5	1.8	3.6	6.7	11.3	15.6
10	3.4	4.5	10.2	16.0	23.0
15	4.0	6.0	11.7	19.2	28.5
20	4.5	7.6	13.6	21.8	41.4
50	10.8	20.2	31.2	41.6	67.8
100	17.5	33.7	53.6	89.3	109.1

TABLE 3 – Temps (ms) obtenus par le programme linéaire

Ci-dessous, la représentation graphique de cette évolution en fonction de différents couples (n, p).

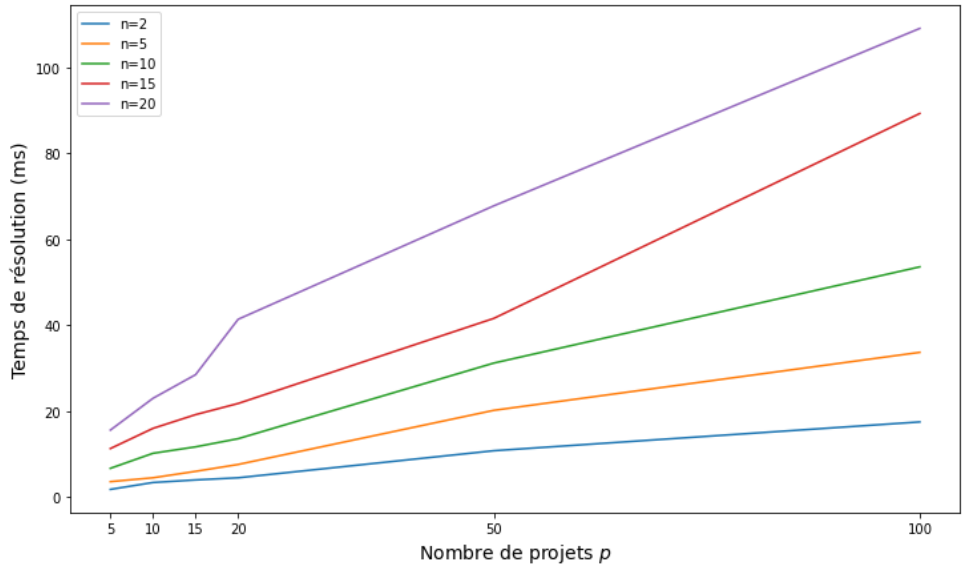
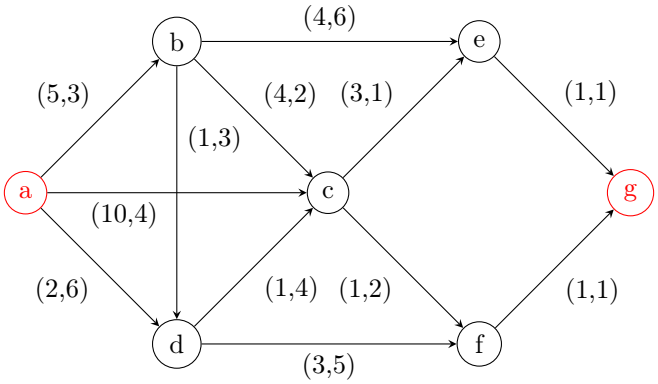


FIGURE 2 – Evolution du temps de résolution du programme linéaire en fonction de (n, p)

Le temps de résolution croit linéairement en fonction de p. De plus, on reste dans des temps de résolution exprimables en millisecondes, ce qui est appréciable, contrairement au cas multi-agents.

4 Application à la recherche d'un chemin robuste dans un graphe

Soit $G = (E, V)$ un graphe, notons $p = |V|$. On s'intéresse dans cette partie aux cas où il y a n scénarios. Le temps entre le sommet i et j dans le scénario s sera noté t_{ij}^s avec la convention $t_{ij}^s = +\infty$ si une telle arête n'existe pas. Dans la suite, on utilisera comme exemple le graphe suivant avec (t_{ij}^1, t_{ij}^2) comme étiquette de l'arc $(i, j) \in E$.



4.1 Calcul du plus cours chemin d'un scénario

Dans un premier temps, on souhaite déterminer le plus court chemin du sommet 1 au sommet p dans le scénario s. Cela revient à résoudre le problème linéaire suivant :

$$\min t^s(P) = \sum_{(i,j)\in E} x_{ij} t_{ij}^s$$

$$s.c. \begin{cases} \sum_{j=2}^{p} x_{1j} = 1 \\ \sum_{i=1}^{p-1} x_{ip} = 1 \\ \sum_{j=1}^{p} x_{ij} = \sum_{j=1}^{p} x_{ji}, \ j = 2, \cdots, p-1 \end{cases}$$

$$x_{ij} \in \{0, 1\}$$

La fonction objectif est le temps pris par le chemin P. P est caractérisé par les variables de décision x_{ij} qui vaut 0 si le chemin ne passe pas par l'arête (i, j) et 1 sinon.

La première contrainte signifie qu'il ne peut y avoir qu'une arête du chemin sortant de 1. La deuxième contrainte signifie qu'il ne peut y avoir qu'une arête du chemin entrant en p. La troisième contrainte signifie que tous les sommets différents de 1 et p ont autant d'arêtes du chemin entrant que sortant.

Étant donné que $t^s_{ij} > 0$, on peut être sûr que si $(i, j) \notin E$ alors $x_{ij} = 0$ à l'optimalité, car la fonction objectif vaudrait $+\infty$ sinon. De même, les sommets autobouclants ($x_{ii} = 1$) isolés ne sont pas interdits par les contraintes mais cela ne ferait que croître strictement la fonction objectif.

Pour l'implémentation, le solveur Gurobi ne peut pas prendre `np.inf` ou `GRB.INFINITY` pour simuler l'infini, il faut donc prendre une valeur $infty$ suffisamment grande. Il suffit de prendre une valeur telle que $infty > \sum_{(i,j)\in E} t^s_{ij}$. Ainsi, passer par une de ces arêtes sera plus long que de passer par des éléments de E, elles ne seront donc jamais utilisées.

En appliquant le programme au scénario 1 de l'exemple, on trouve le chemin de temps 5 suivant :

$$ⓐ \xrightarrow[2]{} ⓓ \xrightarrow[1]{} ⓒ \xrightarrow[1]{} ⓕ \xrightarrow[1]{} ⓖ$$

En appliquant le programme au scénario 2 de l'exemple, on trouve le chemin de temps 6 suivant :

$$ⓐ \xrightarrow[4]{} ⓒ \xrightarrow[1]{} ⓔ \xrightarrow[1]{} ⓖ$$

4.2 Calcul d'un chemin robuste sur plusieurs scénarios

On veut ensuite trouver un chemin qui reste rapide dans les différents scénarios : cela revient à faire de l'optimisation équitable avec comme coût le temps des chemins $t^s(P)$, c'est-à-dire que notre utilité z_i vaut désormais $-t^i$. On souhaite donc résoudre le programme suivant :

$$\max v_f(P) = -\sum_{i=1}^{n} w_i t^{(i)}(P)$$

$$s.c. \begin{cases} \sum_{j=2}^{p} x_{1j} = 1 \\ \sum_{i=1}^{p-1} x_{ip} = 1 \\ \sum_{j=1}^{p} x_{ij} = \sum_{j=1}^{p} x_{ji},\ j = 2, \cdots, p-1 \end{cases}$$

$$x_{ij} \in \{0, 1\}$$

Ce qui revient à résoudre le programme linéaire suivant :

$$\max \sum_{k=1}^{n} w'_k \left(k\, r_k - \sum_{s=1}^{n} b_{sk} \right)$$

$$s.c. \begin{cases} r_k - b_{sk} \leq -t^s,\ s = 1, \ldots, n \\ t^s = \sum_{(i,j)\in E} x_{ij} t^s_{ij} \\ \sum_{j=2}^{p} x_{1j} = 1 \\ \sum_{i=1}^{p-1} x_{ip} = 1 \\ \sum_{j=1}^{p} x_{ij} = \sum_{j=1}^{p} x_{ji},\ j = 2, \cdots, p-1 \end{cases}$$

$$t^s \geq 0, r_k \in \mathbb{R},\ b_{sk} \geq 0$$

On a n variables t^s, $p \times p$ variables x_{ij}, n variables r_k et $n \times n$ variables b_{sk}. On a donc au total $n^2 + p^2 + 2n$ variables de décision pour $n^2 + n + p$ contraintes.

Pour l'implémentaion de l'infini il faut maintenant prendre une valeur telle que $\forall s, infty > \sum_{(i,j)\in E} t^s_{ij}$. Ainsi, une solution utilisant une telle arête sera dominée par n'importe quelle solution passant par des

éléments de E. Par principe de Pareto, ces solutions ne seront jamais choisies.

En appliquant ce programme au graphe exemple on trouve le chemin suivant :

$$\text{ⓐ} \underset{(5,3)}{\longrightarrow} \text{ⓑ} \underset{(4,6)}{\longrightarrow} \text{ⓔ} \underset{(1,1)}{\longrightarrow} \text{ⓖ}$$

Le vecteur temps est alors $t = (10, 10)$.

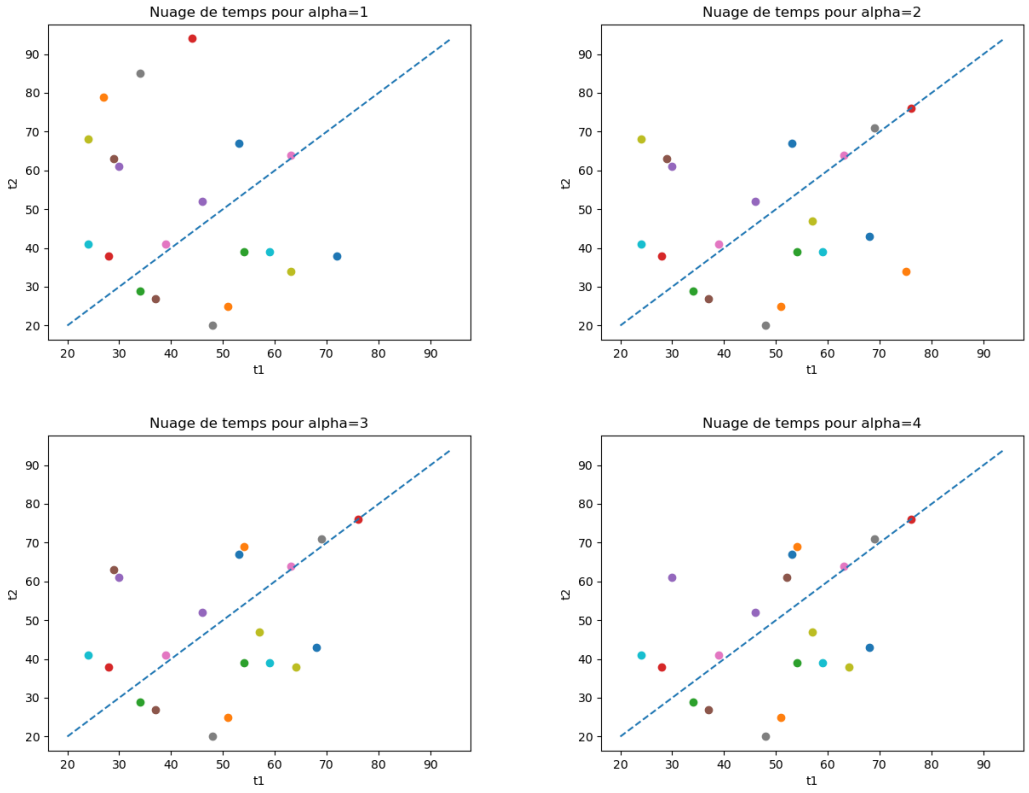
On remarque cependant qu'avec le chemin suivant : ⓐ $\underset{(2,6)}{\longrightarrow}$ ⓓ $\underset{(3,5)}{\longrightarrow}$ ⓕ $\underset{(1,1)}{\longrightarrow}$ ⓖ on obtient le vecteur temps $t = (6, 12)$. La satisfaction est la même, en effet $2 \times 10 + 1 \times 10 = 2 \times 12 + 1 \times 6 = 30$.

4.3 Influence de w sur la robustesse

On souhaite regarder l'influence des poids sur la robustesse. On utilisera les poids :

$$w_i(\alpha) = \left(\frac{n-i+1}{n}\right)^{\alpha} - \left(\frac{n-i}{n}\right)^{\alpha}$$

On va générer aléatoirement 20 scénarios différents sur le graphe exemple puis, pour $\alpha \in [\![1, 5]\!]$ on représentera les temps dans le plan (t^1, t^2). On obtient les nuages de point suivants :



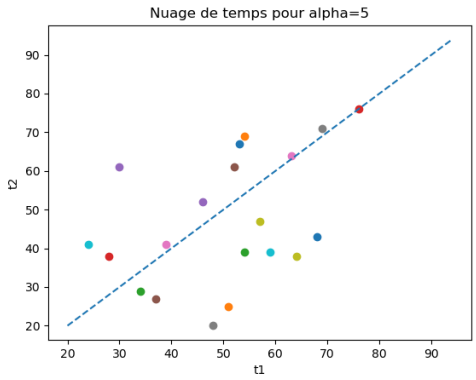


FIGURE 3 – Impact du vecteur de pondération w sur la robustesse de la solution proposée

On remarque que plus α augmente, plus les points se rapprochent de la courbe $t^1 = t^2$ représentée en pointillé. Cela est dû au fait que pour α petit, la différence entre les $w_i(\alpha)$ est faible. On a en particulier $w(1) = (1/n, 1/n, \ldots, 1/n)$: c'est le cas de la minimisation de la moyenne des temps, il n'y a pas de notion de robustesse. Pour α grand, la différence entre les poids augmente, si l'écart entre les temps est trop grand, la fonction objectif sera pénalisée. On a notamment $w(+\infty) = (1, 0, \ldots, 0)$: c'est le cas de la minimisation du temps maximal.

Pour reprendre les termes de l'article, $\alpha = 1$ correspond à une attitude « utilitariste » et $\alpha = +\infty$ correspond à une attitude « égalitariste ». Les α dans $]1, +\infty[$ correspondent au « continuum d'attitude vis-à-vis de l'équité », c'est un compromis entre les deux attitudes extrêmes. Choisir un α revient à se placer sur le spectre unidimensionnel de l'équité.