Projet MOGPL

Optimisation équitable

Jules Bouton

Sorbonne Université
Master ANDROIDE
MU4IN200 Groupe 4
Enseignant: Bruno Escoffier
Année universitaire 2022-2023

Table des matières

1 Linéarisation de f

1.1) Soit $\mathscr{P}_k$ le programme linéaire en variables 0-1 suivant :

$$\begin{aligned} &\min \sum_{i=1}^{k} a_{ik} z_i \\ &s.c. \sum_{i=1}^{k} a_{ik} = k \\ &a_{ik} \in \{0,1\}, i = 1, \ldots, n \end{aligned}$$

$L_k(z) = \sum_{i=1}^{k} z_i$ où z_i sont les composantes de $(z_{(1)}, \ldots, z_{(n)})$ le tri par ordre croissant de $(z_1(x), \ldots, z_n(x))$.$L_k(z)$ est donc la somme des k premiers z_i, c'est-à-dire les k plus petits z_i. Or cette somme est minimale pour les sommes de k éléments parmi les z_i d'où à l'optimum on a bien :

$$\sum_{i=1}^{k} a_{ik} z_i = L_k(z)$$

1.2) Soit la relaxation linéaire suivante :

$$\begin{aligned} &\min \sum_{i=1}^{k} a_{ik} z_i \\ &s.c. \begin{cases} \sum_{i=1}^{k} a_{ik} = k \\ a_{ik} \leq 1 \end{cases} \\ &a_{ik} \geq 0, i = 1, \ldots, n \end{aligned}$$

Posons $\mathscr{D}_k$ le dual :

$$\begin{aligned} &\max\ k r_k - \sum_{i=1}^{n} b_{ik} \\ &s.c.\ r_k - b_{ik} \leq z_i,\ i = 1, \ldots, n \\ &r_k \in \mathbb{R}, b_{ik} \geq 0 \end{aligned}$$

Pour le vecteur $L(4, 7, 1, 3, 9, 2)$ on résout les 6 programmes avec Gurobi et on obtient le vecteur des valeurs à l'optimum suivant $(1, 3, 6, 10, 17, 26)$.

1.3) Montrons que $f(x) = \sum_{k=1}^{n} w'_k L_k(z(x))$ avec $w'_k = \begin{cases} (w_k - w_{k+1}) \text{ si } k = 1, \dots, n-1 \\ w_n \text{ sinon} \end{cases}$.

$$\begin{aligned} f(x) - \sum_{k=1}^{n} w'_k L_k(z(x)) &= \sum_{i=1}^{n} w_i z_i - \sum_{i=1}^{n} w'_i L_i(z(x)) \\ &= \sum_{i=1}^{n} [w_i z_i - w'_i L_i] \end{aligned}$$

Or $L_k - L_{k-1} = \sum_{i=1}^{k} z_i - \sum_{i=1}^{k-1} z_i = z_k$ et $L_1 = z_1$ donc :

$$\begin{aligned} f(x) - \sum_{k=1}^{n} w'_k L_k(z(x)) &= \sum_{i=2}^{n-1} [w_i(L_i - L_{i-1}) - (w_i - w_{i+1})L_i] + w_1 z_1 - w'_1 L_1 + w_n z_n - w'_n L_n \\ &= \sum_{i=2}^{n-1} [w_i(L_i - L_{i-1}) - (w_i - w_{i+1})L_i] + w_2 L_1 + w_n z_n - w'_n L_n \\ &= \sum_{i=2}^{n-1} [w_i(L_i - L_{i-1}) - (w_i - w_{i+1})L_i] + w_2 L_1 + w_n z_n - w_n L_n \\ &= \sum_{i=2}^{n-1} [w_i(L_i - L_{i-1}) - (w_i - w_{i+1})L_i] + w_2 L_1 - w_n(L_n - z_n) \\ &= \sum_{i=2}^{n-1} [w_i(L_i - L_{i-1}) - (w_i - w_{i+1})L_i] + w_2 L_1 - w_n L_{n-1} \\ &= \sum_{i=2}^{n-1} [w_i L_i - w_i L_{i-1} - w_i L_i + w_{i+1} L_i] + w_2 L_1 - w_n L_{n-1} \\ &= \sum_{i=2}^{n-1} [w_{i+1} L_i - w_i L_{i-1}] + w_2 L_1 - w_n L_{n-1} \\ &= \sum_{i=2}^{n-1} w_{i+1} L_i - \sum_{i=2}^{n-1} w_i L_{i-1} + w_2 L_1 - w_n L_{n-1} \\ &= \sum_{i=2}^{n-2} w_{i+1} L_i - \sum_{i=3}^{n-1} w_i L_{i-1} \end{aligned}$$

Et en décalant les indices de la seconde somme on a :

$$\begin{aligned} f(x) - \sum_{k=1}^{n} w'_k L_k(z(x)) &= \sum_{i=2}^{n-2} w_{i+1} L_i - \sum_{i=3-1}^{(n-1)-1} w_{i+1} L_{(i-1)+1} \\ &= \sum_{i=2}^{n-2} w_{i+1} L_i - \sum_{i=2}^{n-2} w_{i+1} L_i \\ &= 0 \end{aligned}$$

D'où :

$$\boxed{f(x) = \sum_{k=1}^{n} w'_k L_k(z(x))}$$

1.4) Sachant que la maximisation de $f(x)$ sur un ensemble X peut s'écrire sous la forme :

$$\begin{gathered} \max \sum_{i=1}^{n} w'_k (k r_k - \sum_{i=1}^{n} b_{ik}) \\ s.c. \begin{cases} r_k - b_{ik} \leq z_i(x), \ i = 1, \ldots, n \\ x \in X \end{cases} \\ b_{ik} \geq 0, i = 1, \ldots, n \end{gathered}$$

On peut réécrire le problème de l'exemple 1 de la sorte :

$$\begin{gathered} \max \ r_1 + 2r_2 - b_{11} - b_{21} - b_{12} - b_{22} \\ s.c. \begin{cases} r_1 - b_{11} \leq 5x_1 + 6x_2 + 4x_3 + 8x_4 + x_5 \\ r_1 - b_{21} \leq 3x_1 + 8x_2 + 6x_3 + 2x_4 + 5x_5 \\ r_2 - b_{12} \leq 5x_1 + 6x_2 + 4x_3 + 8x_4 + x_5 \\ r_2 - b_{22} \leq 3x_1 + 8x_2 + 6x_3 + 2x_4 + 5x_5 \\ x_1 + x_2 + x_3 + x_4 + x_5 = 3 \end{cases} \\ r_k \in \mathbb{R}, \ b_{ik} \geq 0, \ i = 1, \ldots, n, \ k = 1, \ldots, n \\ x_1, \ldots, x_5 \in \{0, 1\} \end{gathered}$$

La résolution par Gurobi donne la solution $x^\star = (0, 1, 1, 1, 0)$ qui a pour valeur 50 soit un score de 18 pour l'agent 1 et 16 pour l'agent 2.

2 Application au partage équitable de biens indivisibles

2.1) On peut formuler ce problème avec le programme linéaire suivant :

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

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

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

$$x_{ij} \in \{0, 1\}\ i = 1, \ldots, n,\ j = 1, \ldots, p$$

Donc, pour l'exemple de l'article, avec $n = 3$ et $p = 6$ avec U la matrice d'utilité $U = \begin{bmatrix} 325 & 225 & 210 & 115 & 75 & 50 \\ 325 & 225 & 210 & 115 & 75 & 50 \\ 325 & 225 & 210 & 115 & 75 & 50 \end{bmatrix}$ (qui ne dépend pas des individus).
Pour $w = (3, 2, 1)$ la résolution avec Gurobi donne l'objet 1 à A, 2 et 4 à B et enfin 3, 5 et 6 à C.
Pour $w = (10, 3, 1)$ la résolution avec Gurobi donne la même affection que la solution précédente.
Si on pose sur la même instance la maximisation de la moyenne des utilités on peut résoudre le programme linéaire suivant:

$$\max \frac{\sum_{i=1}^{n} z_i(x)}{n}$$

$$s.c. \left\{ \sum_{i=1}^{n} x_{ij} = 1,\ j = 1, \ldots, p \right.$$

$$x_{ij} \in \{0, 1\}\ i = 1, \ldots, n,\ j = 1, \ldots, p$$

Et la résolution donne 1, 2 et 3 à A, 4, 5 et 6 à B et rien à C. La moyenne est donc $\frac{1000}{3}$ comme dans les autres solutions (étant donné que l'utilité ne dépend pas des individus dans ce cas) mais la solution n'est pas du tout équitable C n'ayant rien obtenu.

2.2) On obtient les temps de résolutions moyens sur 10 instances aléatoires en faisant varier n et $p = 5n$:

(n, p)	(5,25)	(10,50)	(15,75)	(20,100)	(25,125)
temps (en s.)	0.0593	1.364	3.854	26.547	61.359

On remarque que le temps de résolution augmente rapidement quand la taille du problème augmente. En effet on a n^2 variables binaires ce qui rend la résolution complexe. Il est à noter que sur certaines instances le solveur Gurobi peut passer très longtemps (plus d'une heure et demi) à résoudre le branch and bound, mais cela ne semble pas être lié à un certain type de matrice d'utilité ou de vecteur de pondération.

3 Application à la sélection multicritère de projets

3.1) Ici par rapport à l'exemple 1 du sujet, on change l'espace des solutions réalisables. En effet, là où l'on devait choisir i objets avec i fixé, on doit maintenant choisir j projets tels que la somme de leur coût ne dépasse pas l'enveloppe budgétaire b. On peut donc formuler ce problème comme le programme linéaire suivant:

$$\max \sum_{i=1}^{n} w'_k(kr_k - \sum_{i=1}^{n} b_{ik})$$

$$s.c. \begin{cases} r_k - b_{ik} \leq z_i(x),\ i = 1, \ldots, n \\ \sum_{k=1}^{p} x_k c_k \leq \frac{1}{2} \sum_{k=1}^{p} c_k \end{cases}$$

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

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

Où n est le nombre d'objectifs de l'entreprise et p le nombre de projets.
Sur l'exemple du sujet on a $n = 2,\ p = 4,\ c = (40, 50, 60, 50)$ et $U = \begin{bmatrix} 19 & 6 & 17 & 2 \\ 2 & 11 & 4 & 18 \end{bmatrix}$.
Pour les vecteurs $w = (2, 1)$ et $w = (10, 1)$ la résolution à l'aide de Gurobi donne pour solution optimale la même sélection de projets $x^\star = (1, 0, 0, 1)$ soit $z(x^\star) = (21, 20)$. La solution qui maximise la moyenne des utilités est $x^\star = (1, 0, 1, 0)$ or celle-ci est très inéquitable avec $z(x^\star) = (36, 6)$ pour un gain de seulement 0.5 sur la moyenne.

3.2) On obtient les résultats suivant en terme de performance (n en ligne et p en colonne, toujours des temps en secondes):

	5	10	15	20
2	7e-4	8e-4	1e-3	2e-3
5	1e-3	2e-3	3e-3	3e-3
10	2e-3	4e-3	5e-3	0.011

Ici le nombre de variables binaires est linéaire par rapport à p, ce qui rend la résolution relativement rapide pour des petites instances et raisonnable quand n et p augmentent Pour des tailles plus grandes comme $n = 100$ et $p = 100$, on a observé (sur 10 instances aléatoires) un temps moyen de 8 secondes.

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

4.1) On recherche un plus court chemin entre a et g dans un graphe $G = (S, A)$ orienté, dont le coût des arcs est positif. Ce problème peut se reformuler comme un problème de flot maximal coût minimum. On prend a l'unique source

et g l'unique puits, sans contrainte de capacité et avec une fonction de coût inchangée. Ainsi on recherche bien chemin de coût minimum de a à g. On peut le résoudre, pour le scénario i, par le programme linéaire suivant :

$$\min \sum_{(u,v)\in A} t^i_{uv} x_{uv}$$

$$s.c. \begin{cases} \sum_{(a,v)\in A} x_{av} - \sum_{(v,a)\in A} x_{va} = 1 \\ \sum_{(g,v)\in A} x_{gv} - \sum_{(v,g)\in A} x_{vg} = -1 \\ \sum_{(u,v)\in A} x_{uv} - \sum_{(v,u)\in A} x_{vu} = 0, \ \forall u \neq a, g \end{cases}$$

$$x_{uv} \geq 0 \ \forall u \forall v, (u,v) \in A$$

Cette formulation utilise m variables réelles positives x_{uv} pour $(u,v) \in A$.
Sur le graphe de la figure 1 du sujet, la résolution donne le chemin $a \to d \to c \to f \to g$ de durée 5 pour le scénario 1 et $a \to c \to e \to g$ de durée 6 pour le scénario 2.

4.2) Afin de maximiser $v_f(P)$ on pose le programme linéaire en variables mixtes suivant basé sur la linéarisation vu dans la partie 1 :

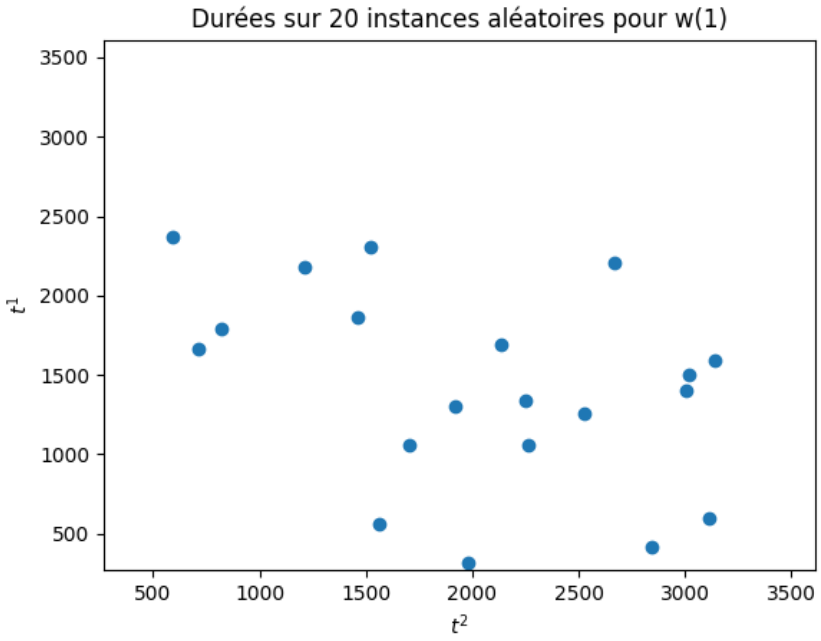
$$\max \sum_{i=1}^{n} w'_k (k r_k - \sum_{i=1}^{n} b_{ik})$$

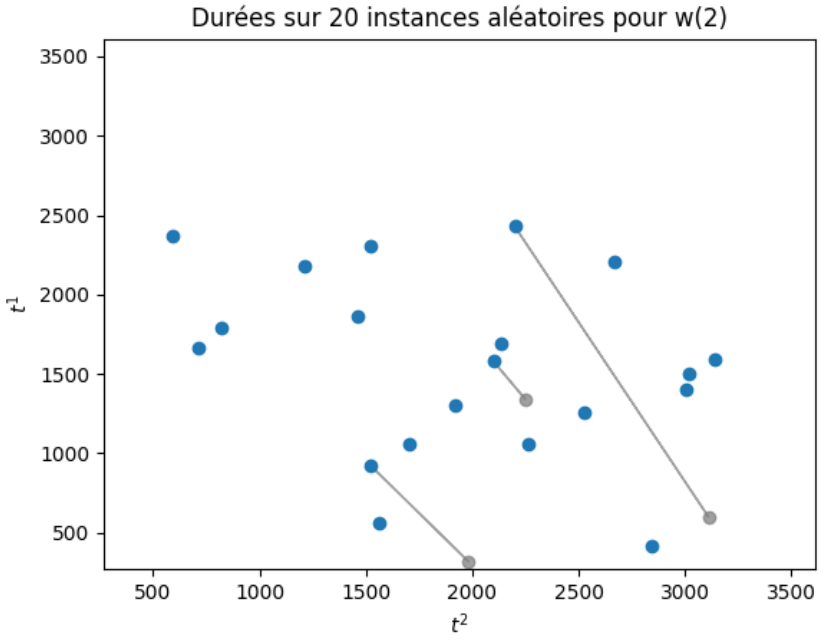
$$s.c. \begin{cases} r_k - b_{ik} \leq - \sum_{(u,v)\in A} x_{uv} t^i_{uv}, \ i = 1, \ldots, n \\ \sum_{(a,v)\in A} x_{av} - \sum_{(v,a)\in A} x_{va} = 1 \\ \sum_{(g,v)\in A} x_{gv} - \sum_{(v,g)\in A} x_{vg} = -1 \\ \sum_{(u,v)\in A} x_{uv} - \sum_{(v,u)\in A} x_{vu} = 0, \ \forall u \neq a, g \end{cases}$$

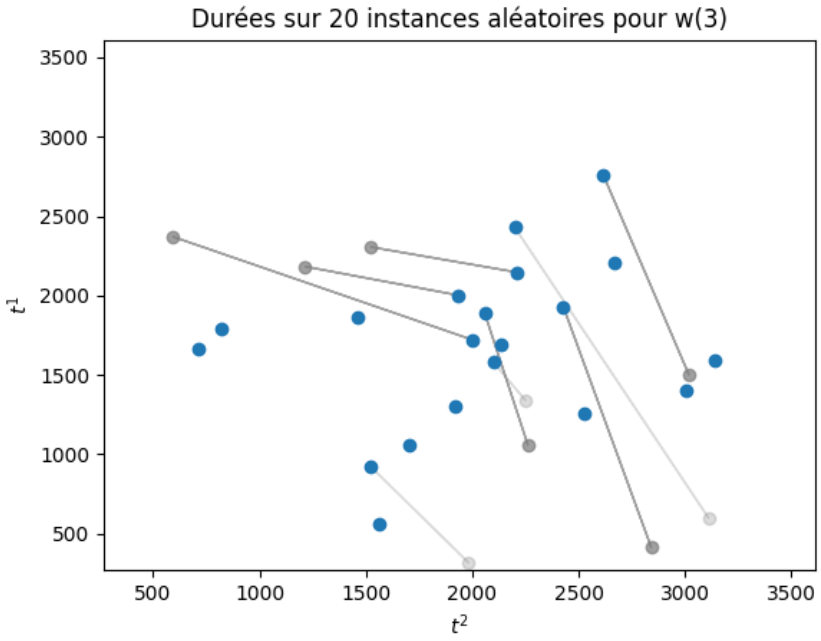
$$b_{ik} \geq 0, i = 1, \ldots, n$$

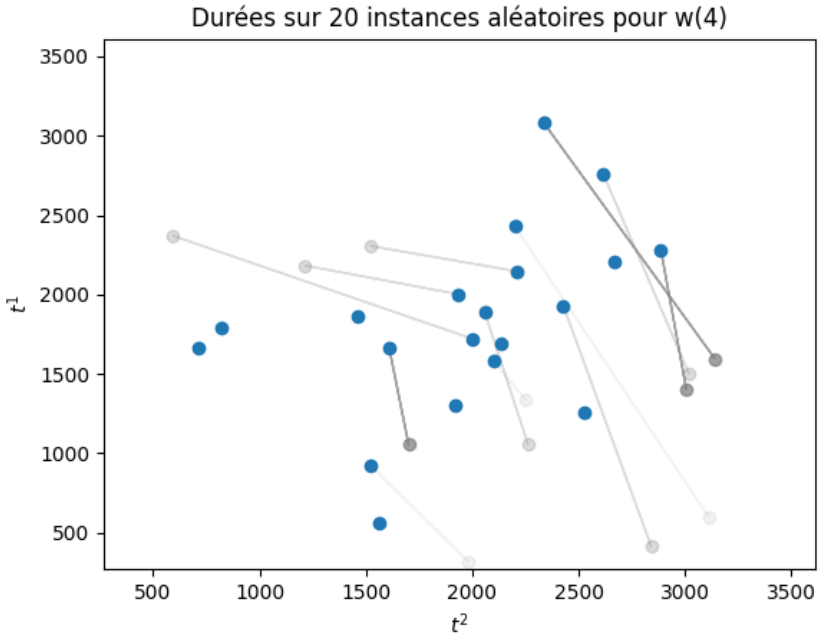
$$x_{uv} \in \{0,1\} \ \forall u \forall v, (u,v) \in A$$

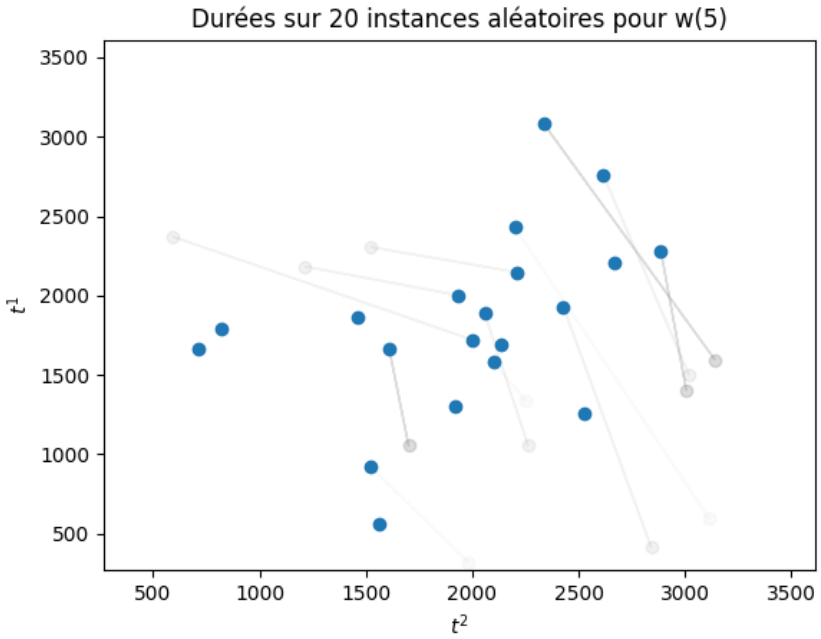
Le chemin alors renvoyé est $a \to b \to c \to f \to g$. Il a une durée de 11 dans le scénario 1 et 8 dans le scénario 2. Sa valeur est donc -30 pour la fonction objectif.
4.3) On obtient les 5 nuages de points suivants, avec en gris l'évolution:


Durées sur 20 instances aléatoires pour w(1)
3500
3000
2500
2000
1500
1000
500
t^1
500
1000
1500
2000
2500
3000
3500
t^2


Durées sur 20 instances aléatoires pour w(2)
3500
3000
2500
2000
1500
1000
500
t^1
500
1000
1500
2000
2500
3000
3500
t^2


Durées sur 20 instances aléatoires pour w(3)
3500
3000
2500
2000
1500
1000
500
t^1
500
1000
1500
2000
2500
3000
3500
t^2


Durées sur 20 instances aléatoires pour w(4)
3500
3000
2500
2000
1500
1000
500
t^1
500
1000
1500
2000
2500
3000
3500
t^2



On peut voir α comme un curseur entre une solution qui va d'une pondération uniforme entre les scénarios (donc le chemin le plus rapide en moyenne) et plus on le place haut plus l'exigence de robustesse est grande. Graphiquement, on remarque que les solutions se rapprochent de la droite des solutions parfaitement équitables (mais pas nécessairement réalisables) $t^1 = t^2$ quand α augmente.