Rapport de projet

SORBONNE UNIVERSITÉ

Projet MOGPL

Optimisation équitable

Binôme :

- **Amayas SADI**
- **Ghiles OUHENIA**

Groupe : **04**

Chargé de TD : **Bruno Escoffier**

Contents

Introduction

Le but de ce projet est de résoudre différents problèmes de partage de bien entre agents de façon équitable. Pour cela, il suffit de trouver une distribution des biens x maximisant $f(x) = \sum_{i=1}^{n} w_i z_{(i)}(x)$.

Comme $f(x)$ n'est pas une fonction linéaire, il faut donc la linéariser en procédant par plusieurs étapes comme suit :

1 Linéarisation de f

1.1 $L_k(z)$ valeur à l'optimum

Soit $P(k)$ le Programme linéaire ci-dessous :

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

La contrainte $\sum_{i=1}^{n} a_{ik} = k$ avec les a_{ik} des variables booléennes pour $i = 1, \ldots, n$ nous force à avoir k variables a_i active (i.e valent 1), puisque la fonction objective est une somme de pondérations des z_i par leurs a_i respectifs, on va sélectionné k valeurs de z_i pour les sommer, et comme on est dans un problème de *minimisation* on cherchera à sélectionner les k plus petites valeurs de Z. Par définition $L_k(z)$.

1.2 Le dual

Soit $P'(k)$ le programme linéaire relaxé en variables continues de $P(k)$ ci-dessous :

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

On construit le dual $D(k)$ associé à $P'(k)$ comme suit :

- comme les variables $a_{ik} \geq 0$ le sens de l'inégalité des contraintes du dual associées est $\leq$.
- $r_k \in \mathrm{R}$ car la contrainte qui lui ai associé dans le primale est une égalité $=$.
- $b_{ik} \leq 0$ car la contrainte qui lui ai associé dans le primale est $\leq$.

Ce qui donne :

$$max\ k.r_k + \sum_{i=1}^{n} b_{ik}$$
$$s.c. \begin{cases} r_k + b_{ik} \leq z_i \end{cases}$$
$$r_k \in \mathrm{R},\ b_{ik} \leq 0, i = 1, \ldots, n$$

Pour résoudre $D(k)$ nous avons décider de coder une fonction python $resolve\_1\_2(Z, k)$ qui prend en argument Z et k qui retourne la valeur à l'optimum de $D(k)$ qui est $L_k(Z)$.

Pour calculer le vecteur $L(Z)$ il suffit de faire n appels à la fonction $resolve\_1\_2(Z, k)$, ce qui donne pour $L(4, 7, 1, 3, 9, 2)$ le vecteur $(1, 3, 6, 10, 17, 26)$.

1.3 Démonstration

On veut montre que :

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

avec :

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

on a :

$\sum_{k=1}^{n} w'_k L_k(z(x)) = \sum_{k=1}^{n-1}(w_k - w_{k+1})L_k(z(x)) + w_n L_n(z(x))$

$= (w_1 - w_2)L_1(z(x)) + (w_2 - w_3)L_2(z(x)) + (w_3 - w_4)L_3(z(x)) + \cdots + (w_{n-2} - w_{n-1})L_{n-2}(z(x)) + (w_{n-1} - w_n)L_{n-1}(z(x)) + w_n L_n(z(x))$

$= w_1 L_1(z(x)) + w_2(L_2(z(x)) - L_1(z(x))) + \cdots + w_{n-1}(L_{n-1}(z(x)) - L_{n-2}(z(x))) + w_n(L_n(z(x)) - L_{n-1}(z(x)))$

et on a : $L_k(z(x)) - L_{k-1}(z(x)) = z_{(k)}(x)$
car : $L_k(z(x)) - L_{k-1}(z(x)) = \sum_{i=1}^{k} z_{(i)}(x) - \sum_{i=1}^{k-1} z_{(i)}(x) = z_{(k)}(x)$

ce qui nous donne :

$$\sum_{k=1}^{n} w'_k L_k(z(x)) = w_1 z_{(1)}(x) + w_2 z_{(2)}(x) + \cdots + w_n z_{(n)}(x)$$

$$= \sum_{i=1}^{n} w_i z_{(i)}(x) = f(x) \quad c.q.f.d$$

1.4 Reformulation exemple 1

Reformulation de l'exemple 1 comme un programme linéaire :

$$max \; \sum_{k=1}^{2} w'_k (k.r_k + \sum_{i=1}^{2} b_{ik})$$

$$s.c. \begin{cases} r_1 + b_{11} - (5x_1 + 6x_2 + 4x_3 + 8x_4 + x_5) \leq 0 \\ r_1 + b_{21} - (3x_1 + 8x_2 + 6x_3 + 2x_4 + 5x_5) \leq 0 \\ r_2 + b_{12} - (5x_1 + 6x_2 + 4x_3 + 8x_4 + x_5) \leq 0 \\ r_2 + b_{22} - (3x_1 + 8x_2 + 6x_3 + 2x_4 + 5x_5) \leq 0 \\ x_1 + x_2 + x_3 + x_4 + x_5 = 3 \end{cases}$$

$$r_1, r_2 \in \mathrm{R}, \; x_i \in \{0,1\}, i = 1, \ldots, 5, \quad b_{ik} \leq 0, \; i, k \in \{1, \ldots, 5\}$$

On résout ce problème avec la fonction $resolve\_1\_4(coef, W)$.

- Pour W=(2,1) on obtient : $(x_1, x_2, x_3, x_4, x_5) = (0, 1, 1, 1, 0)$, $donc$ $(z_1, z_2) = (18, 16)$ avec une valeur objective de 50.

2 Application au partage équitable de biens indivisibles

2.1 Reformulation du problème en variables mixtes

Notre probléme ce résume à:

$$max \; \sum_{i=1}^{n} w_i * z_{(i)}$$

$$s.c. \begin{cases} \sum_{j=1}^{p} coef_{ij} * x_{ij} = z_i \\ \sum_{i=1}^{n} x_{ij} \leq 1 \end{cases}$$
$$x_{ij} \in \{0,1\}, i \in \{1,\ldots,n\}, j \in \{1,\ldots,p\}$$

Pour le résoudre, on le reformule comme suit :

$$max \sum_{k=1}^{n} w'_k(k.r_k + \sum_{i=1}^{n} b_{ik})$$
$$s.c. \begin{cases} r_k + b_{ik} - \sum_{j=1}^{p} coef_{ij} * x_{ij} \leq 0 \\ \sum_{i=1}^{n} x_{ij} \leq 1 \end{cases}$$
$$r_k \in \mathrm{R},\ b_{ik} \leq 0,\ x_{ij} \in \{0,1\}$$
$$i,k \in \{1,\ldots,n\},\ j \in \{1,\ldots,p\}$$

Tel que :

- $coef_{ij}$: est l'utilité de l'objet j pour l'agent i.
- x_{ij} : est une variable booléenne qui vaut 1 lorsque l'objet j est affecté à l'agent i.
- p : le nombre de produits.
- n : le nombre d'agents.
- La première contrainte sera répéter n^2 fois.
- La deuxième contrainte nous dit que chaque produit posséde au plus un agent (par conséquent répéter p fois).

Pour résoudre ce PL, on utilise la fonction $resolve\_2\_1(coef, W)$.

- Avec w1 = (3, 2, 1) on aura : (335, 325, 340)
  Avec la répartition suivante :

Objets / Agents	x_1	x_2	x_3	x_4	x_5	x_6
1	0	0	1	0	1	1
2	1	0	0	0	0	0
3	0	1	0	1	0	0

- Avec w2 = (10, 3, 1) on aura : (325, 340, 335)
  Avec la répartition suivante :

Objets / Agents	x_1	x_2	x_3	x_4	x_5	x_6
1	1	0	0	0	0	0
2	0	1	0	1	0	0
3	0	0	1	0	1	1

  Remarque : Nous constatons qu'en changeant W la distribution des objets pour chaque agent change, mais nous conservons toujours l'équité du partage.

- En maximisant la satisfaction moyenne des individus définie par (z1(x) + z2(x) + z3(x))/3, on aura : (0, 760, 240)
  Avec la répartition suivante :

Objets / Agents	x_1	x_2	x_3	x_4	x_5	x_6
1	0	0	0	0	0	0
2	1	1	1	0	0	0
3	0	0	0	1	1	1

Remarque : par contre, maximiser la satisfaction moyenne des individus entraîne un partage inéquitable.

2.2 L'évolution du temps de résolution

Nous étudierons l'évolution de la durée de résolution selon n et p.

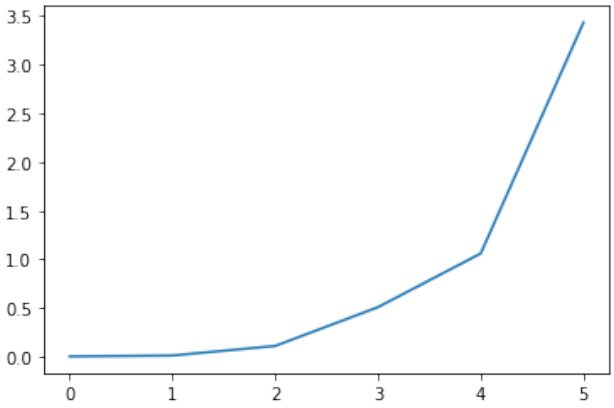

Figure 1: Graphe d'évolution du temps en fonction de n

Remarque : plus la valeur de n est élevé (par conséquence p aussi), plus la durée de résolution est longue. Ce qui est assez normal parce que nous ajoutons plus d'objets et d'agents ainsi notre PL devient plus complexe à résoudre.
Nous notons également que la complexité est au moins quadratique étant donnée la façon dont notre graphique évolue.

3 Application à la sélection multicritère de projets

Notre problème ce résume à:

$$max \sum_{i=1}^{n} w_i * z_{(i)}$$
$$s.c. \begin{cases} \sum_{j=1}^{p} coef_{ij} * x_j = z_i \\ \sum_{j=1}^{p} x_j * C_j \leq b \end{cases}$$
$$x_j \in \{0,1\}, j \in \{1,\ldots,p\}$$

Pour le résoudre, on le reformule ainsi :

$$max \sum_{k=1}^{n} w'_k(k.r_k + \sum_{i=1}^{n} b_{ik})$$
$$s.c. \begin{cases} r_k + b_{ik} - \sum_{j=1}^{p} coef_{ij} * x_j \leq 0 \\ \sum_{j=1}^{p} x_j * C_j \leq b \end{cases}$$
$$r_k \in \mathrm{R},\ b_{ik} \leq 0,\ x_j \in \{0,1\}$$

$$i, k \in \{1, \ldots, n\},\ j \in \{1, \ldots, p\}$$

Tel que :

- $coef_{ij}$: est l'utilité du projet j pour l'agent i.
- x_j : est une variable booléenne qui vaut 1 lorsque le projet j est sélectionné.
- C_j : est le cout associé au projet j.
- p : le nombre de projets.
- n : le nombre d'agents.
- La première contrainte sera répéter n^2 fois.
- La seconde contrainte nous indique que le coût total des projets sélectionnés ne doit pas dépasser un budget déterminé.

Pour b = 100 :

- Pour w = (2, 1) on trouve : $(x_1, x_2, x_3, x_4) = (1, 0, 0, 1),\ avec\ (z_1, z_2) = (21, 20)$.
- Pour w = (10, 1) on trouve : $(x_1, x_2, x_3, x_4) = (1, 0, 0, 1),\ avec\ (z_1, z_2) = (21, 20)$.

Remarque : Nous constatons qu'en changeant W la sélection des projets ne change pas et nous conservons toujours l'équité du partage.

- En maximisant la satisfaction moyenne des individus définie par (z1(x) + z2(x))/2, on trouve : $(x_1, x_2, x_3, x_4) = (1, 0, 1, 0),\ avec\ (z_1, z_2) = (36, 6)$

Remarque : par contre, maximiser la satisfaction moyenne des individus entraîne un partage inéquitable. On remarque aussi que l'utilité moyenne pour chaque agent avec l'utilisation des poids W est de (21+20)/2=20.5 et 21 dans l'autre cas, donc le deuxième maximise mieux l'utilité globale mais les écarts des utilités pour chaque agent sont trop importants.

3.1 L'évolution du temps de résolution

Nous étudierons l'évolution de la durée de résolution selon n et p.

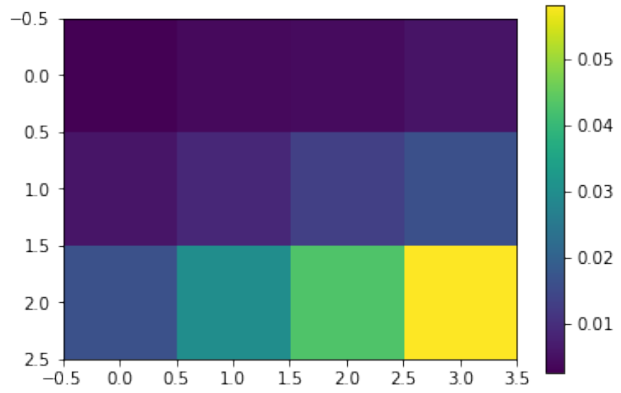

Figure 2: Matrice d'évolution du temps selon n et p

Remarque : plus les valeurs (n, p) sont élevées, plus la durée de résolution est longue. Ce qui est assez normal parce que nous ajoutons plus de projets et d'agents ainsi notre PL devient plus complexe à résoudre.

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

4.1 Re-formulation du programme linéaire

(P) est un programme qui permets de calculer le chemin le plus rapide du sommet initial au sommet destination dans un scénario s, définit comme suite :

$$\begin{gathered} min \sum_{a=(i,j)\in A} T_a * x_a \\ s.c. \sum_{a=(i,j)\in A} x_a - \sum_{a=(j,i)\in A} x_a = d_i \\ x_a \in \{0,1\},\ a \in A \end{gathered}$$

Tel que :

- T_{ij} représente le temps de $l'arc_{ij}$ selon le scénario s.
- A représente l'ensemble des arcs du graphe, tel que chaque arc est identifié par (i,j) avec i désigne le noeud de départ et j le noeud d'arrivée.
- $di \in \{-1,0,1\},\ i \in \{1,\ldots,p\}$, avec $d_a = 1,\ d_g = -1$ et le reste à 0, p le nombre de noeuds du graphe.
- La contrainte sert à définir les noeuds de départ, d'arrivée et de transits.

C'est à dire :
Si di = 1 : cela identifie le noeud de départ car on peut uniquement en sortir.
Si di = -1 : c'est le noeud d'arrivée parce qu'on peut juste y entrer.
Si di = 0 : désigne les noeuds de transits, car si on y entre on est obligé d'en sortir.

la contrainte appliquée pour tous les noeuds permet de construire un chemin partant du noeud de départ jusqu'à l'arrivée.

Pour calculer les plus courts chemins, on utilise la fonction $resolve\_4\_1(T,a,g)$.

- Pour le premier scénario, le chemin optimal est : $(a \to d \to c \to f \to g)$ avec un temps total de 5 unités de temps.
- Pour le deuxième scénario, le chemin optimal est : $(a \to c \to e \to g)$ avec un temps total de 6 unités de temps.

4.2 Chemin robuste

Pour déterminer le chemin le plus robuste pour tous les scénarios, on applique le résultat de la question 1.4 sur le PL précédent, on obtient :

$$\begin{gathered} max \sum_{k=1}^{n} w'_k(k.r_k + \sum_{s=1}^{n} b_{sk}) \\ s.c. \begin{cases} r_k + b_{sk} - \sum_{a=(i,j)\in A} -T_a^s * x_a \le 0 \\ \sum_{a=(i,j)\in A} x_a - \sum_{a=(j,i)\in A} x_a = d_i \end{cases} \\ r_k \in \mathrm{R},\ b_{sk} \le 0,\ x_a \in \{0,1\} \\ s,k \in \{1,\ldots,n\},\ a \in A \end{gathered}$$

Tel que :

- n désigne le nombre d'agents (dans notre cas le nombre de scénarios).
- T_{ij}^s représente le temps de $l'arc_{ij}$ selon le scénario s.

On détermine le chemin le plus robuste partant de a à g avec la fonction $resolve\_4\_2(T,W,a,g)$.

- Pour $W = (2,1)$, le chemin le plus robuste est $(a \to b \to c \to f \to g)$ avec un temps total pour chaque scénario de (11, 8).

4.3 L'impact de la pondération w sur la robustesse de la solution proposée

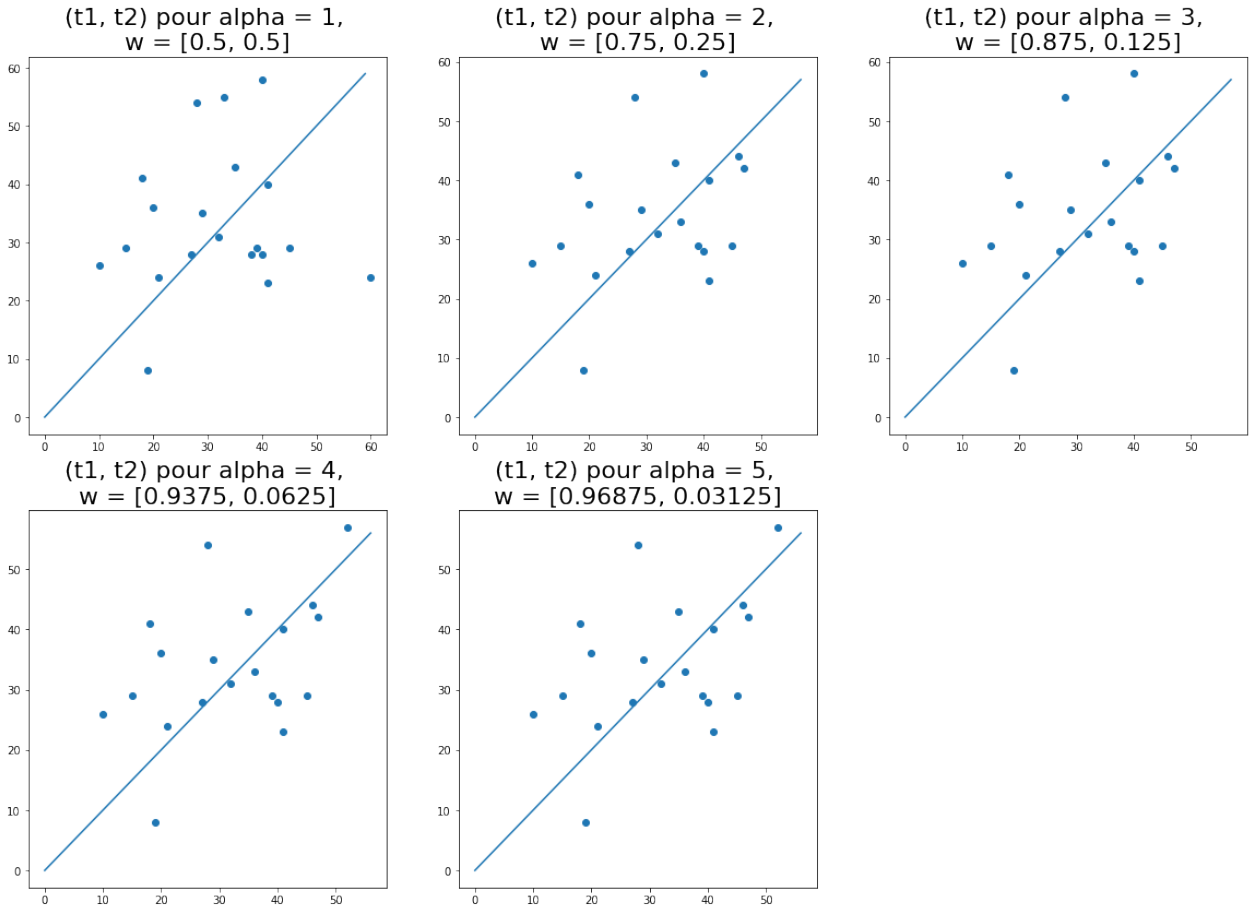

Figure 3: Répartions des points (t^1, t^2) en fonction de la pondération W

Remarque : Plus alpha est élevé, plus l'écart entre les W est grand, plus le chemin devient robuste (temps équitable entre les scénarios).
On le remarque aussi graphiquement (les points se rapprochent de la droite x=y, ce qui veut dire que les écarts entre les temps de parcours de chaque scénario sont petits).