Faculté des Sciences et Ingénierie - Sorbonne université

Master Informatique parcours - DAC / ANDROIDE

MOGPL - Modélisations, Optimisation, Graphes et Programmation Linéraire

Rapport de projet

Optimisation équitable

Réalisé par :
BENHADDAD Sabrina - M1 DAC
SLIMANI Nour Ismahane - M1 ANDROIDE

Décembre 2022

Table des matières

Table des figures

Introduction

Les décisions prises au sein des organisations ont souvent un impact qui s'apprécie selon plusieurs individus et donc plusieurs points de vue, qu'il convient de considérer de manière équitable.
Dans les problèmes de décision ou d'optimisation multiagents, l'équité est souvent au cœur des préoccupations, qu'il s'agisse de répartir des ressources, d'allouer des tâches, de partager des bénéfices ou plus généralement de prendre en compte les préférences individuelles [**1**].
L'optimisation équitable consiste donc, à trouver des solutions efficaces en apportant un soin particulier à préserver l'équilibre entre la satisfaction des différents points de vue considérés.
De manière générale, nous nous intéressons dans ce projet, à un problème d'optimisation multi-dimensionnel sur un ensemble X (continu ou discret) défini par un système de contraintes dont chaque solution réalisable x est évaluée par n fonctions représentant n points de vues potentiellement différents sur le coût de la solution x. Ainsi, toute solution x sera évaluée par un vecteur $z(x)$ représentant le coût ou performance de x selon le point de vue i.
A noter que ces points de vus représentent des évaluations ou critères d'évaluation en fonction des problèmes considérés.
C'est ainsi que l'optimisation équitable vise à trouver une solution efficace équitable, que l'on ne peut améliorer sur une composante sans la dégrader sur une autre (d'après la définition de l'équité), et qui conduit à un vecteur $z(x)$ équilibré.[**2**]

Notre travail portera tout d'abord sur la linéarisation de la fonction objectif du problème d'optimisation équitable puis, traitera de ses différentes applications que voici :

— Le partage équitable de biens indivisibles

— La sélection multicritère de projets

— La recherche d'un chemin robuste dans un graphe

L'objectif étant de répondre à toutes les questions de l'énoncé du projet.

Chapitre 1

Linéarisation de f

Comme le suggère l'énoncé ainsi que l'article introductif donné, une façon classique d'obtenir une optimisation équitable est de maximiser une moyenne pondérée ordonnée des composantes $z_i(x)$. On cherche donc une solution $x \in X$ qui maximise la fonction :

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

Où w_i sont des poids positifs et décroissants lorsque i augmente ($w_i \geq w_{i+1}, i = 1, ..., n-1$) qui sont fixés par l'utilisateur. On peut également interprété w_i comme étant l'importance attachée au $i^{ème}$ individu le moins bien doté dans la solution.[**3**].
Le vecteur $(z_1(x), ..., z_n(x))$ représente le résultat d'un tri des composantes de $z(x)$ par ordre croissant car nous traitons un problème de maximisation (ainsi $z_i(x) \leq z_{i+1}(x), i = 1, ..., n-1$).

Or en prêtant attention à l'exemple 1 du sujet de l'énoncé, on peut facilement se rendre compte que la fonction objectif n'est pas linéaire.
Nous devons par conséquent obligatoirement la linéariser pour pouvoir procéder à la résolution du PL.

Pour ce faire, commençons par considérer le PL en variable binaire suivant :

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

Pour tout vecteur $z \in \mathbb{R}$, on note $L(z)$ le vecteur $(L_1(z), ..., L_n(z))$ dont la composante $L_k(z)$ est définie par $L_k(z) = \sum_{i=1}^{k} z_i$.
On peut associer le vecteur $L(z)$ au vecteur de Lorenz associé à z, obtenu en triant les composantes de z par ordre croissant pour un problème de maximisation (décroissant pour un problème de minimisation) puis, en calculant les sommes partielles des premières composantes du vecteur z trié[**3**].

A. Expliquons pourquoi $L_k(z) = \sum_{i=1}^{k} z_i$ est la valeur optimale du problème linéaire ci-dessus.
Nous pouvons remarquer assez naturellement dans un premier temps, que les variables de

décision de ce PL sont les a_{ik} à valeur booléenne. La solution optimale sera par conséquent, un vecteur contenant un ensemble de 0 et de 1 satisfaisant obligatoirement la contrainte d'égalité du PL.
Ce qui signifie donc, que le vecteur de la solution optimale qu'on notera a^* contiendra k variables égales à 1, tandis que le reste prendra la valeur 0.
En remplaçant dans la fonction objectif on obtient donc :

$$\sum_{i=1}^{n} a_{ik} z(i) = \sum_{i=1}^{k} a_{ik} z(i) + \sum_{i=k+1}^{n} a_{ik} z(i)$$

D'après la contrainte d'égalité : $\sum_{i=1}^{k} a_{ik} = 1 + 1 + 1 + 1 + 1 + 1 + ... + 1 = k$ (on somme 1 k fois) donc : $\sum_{i=1}^{k} a_{ik} z(i) = \sum_{i=1}^{k} z(i)$ et $\sum_{i=k+1}^{n} a_{ik} z(i) = 0$ car $\sum_{i=k+1}^{n} a_{ik} = 0$ étant donné que ces variables prendront la valeur 0.

D'où, la valeur de la fonction objectif est :

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

Ainsi, nous avons montré que $L_k(z)$ est forcément la valeur optimale du programme linéaire en variable 0-1 ci-dessus.

B. On considère à présent le PL relaxé en variables continues suivant :

$$(P_k) \begin{cases} Min \sum_{i=1}^{n} a_{ik} z_i \\ \sum_{i=1}^{n} a_{ik} = k \\ a_{ik} \leq 1 \\ a_{ik} \geq 0, i = 1, ...n \end{cases}$$

Étant donné que les solutions optimales du problème relaxé sont naturellement entières, nous pouvons considérer $L_k(z)$ comme étant la valeur à l'optimum de ce programme linéaire également.

Construisons à présent D_k le dual de ce programme relaxé comme ceci :

$$(D_k)\begin{cases} Max\, k * r_k + \sum_{i=1}^{n} b_{ik} \\ r_k + b_{ik} \leq z(i) \\ b_{ik} \leq 0 \\ r_k \in \mathbb{R}, i = 1, ...n \end{cases}$$

Avec r_k la variable duale associée à la première contrainte du primal et b_{ik} les variables duales associées aux contraintes $a_{ik} \leq 1, i = 1, ...n$.

Utilisons cette formulation duale pour calculer par programmation linéaire les composantes du vecteur $L(4, 7, 1, 3, 9, 2)$.
Nous savons dans un premier temps en utilisant la définition du vecteur de Lorenz mentionnée dans l'annexe[**3**], que $L(x_1, ..., x_n) = (x_1, x_1 + x_2, ..., x_1 + .. + x_n)$ où x_i est la $i^{ème}$ plus petite composante du vecteur x.
Ainsi, pour calculer $L(x)$, on trie les composantes de x par ordre croissant car il s'agit d'un problème de maximisation, puis on calcule les sommes partielles des premières composantes du vecteur trié. On doit par conséquent naturellement, retrouver le vecteur suivant : $L = (1, 3, 6, 10, 17, 26)$.

Procédons maintenant au calul demandé par l'énoncé, à savoir le calcul des composantes du vecteur L à l'aide de la programmation linéaire et du dual construit.

Commençons par calculer la première composante du vecteur à savoir $L_1(z)$ où $z = (4, 7, 1, 3, 9, 2)$ et $k = 1$.
Soit a^* solution optimale telle que :

$$\sum_{i=1}^{n} a_{ik} = 1 \;\; et \;\; a_{ik} \leq 1 \;\; i = 1, ..., n$$

On peut facilement conclure que dans ce cas précis, seule une variable est égale à 1, les autres étant à 0. Ce qui implique qu'une seule contrainte sera saturée dans le dual (d'après le théorème des écarts complémentaires).
De plus, nous remarquons aisément que seules 2 contraintes parmi les $n+1$ au total, sont saturées dans le primal. En appliquant le théorème des écarts complémentaires, les $n-1$ variables correspondantes du dual (b_{ik}) seront nulles.
Ainsi, en appliquant les hypothèses du théorème des écarts complémentaires, on obtient :

$$r_1 + b_{1,1} = z_1$$

z_1 réalisable du dual donc, d'après le TEC, z_1 est également optimale du dual. Avec $OPT(P_k) = OPT(D_k)$ et donc :

$$L_1(z) = z_1$$

On applique le même principe pour les autres composantes, on trouvera par exemple pour $k = 2$:

$$2r_2 + b_{1,2} + b_{2,2} = L_2(z) = z_1 + z_2$$

On retrouve par conséquent :

$$L(z) = (z_1, z_1 + z_2 + ... + \sum_{i=1}^{6} z_i)$$

Étant donné que les z_i doivent être trié par ordre croissant d'après l'énoncé ($z_i \leq z_{i+1}$, z_1 doit être la plus petite valeur de z, donc :

$$z_1 = 1, z_2 = 2, z_3 = 3, z_4 = 4, z_5 = 7, z_6 = 9$$

Enfin, $L(4, 7, 1, 3, 9, 2) = (1, 3, 6, 10, 17, 26)$, CQFD.

C. Montrons la nouvelle écriture de f :
On sait par définition que :

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

et que :

$$w_i = \sum_{k=1}^{n} v_k$$

tel que (d'après le document[**3**]) : $\begin{cases} v_i = w_i - w_{i+1} & i < n \\ v_n = w_n & i = n \end{cases}$
Ce qui correspond exactement à la définition de w'_k. Donc $w'_k = v_i$ et :

$$w_i = \sum_{k=1}^{n} w'_k$$

En l'injectant dans la formule de f on obtient :

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

car : $L_k(z(x)) = \sum_{i=1}^{n} z_i(x)$
On a ainsi montré que $f(x) = \sum_{k=1}^{n} w'_k L_k(z(x))$, CQFD.

D. En utilisant les résultats des questions précédentes, la maximisation de $f(x)$ sur un

ensemble X décrit par des contraintes linéaires peut s'écrire sous la forme du programme linéaire suivant :

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

Cette formulation linéaire utilise n^2 variables réelles positives b_{ik} pour $i, k \in \{1, .., n\}$ et n variables non signées $r_k, k \in \{1, .., n\}$.

Reformulons le problème de l'exemple 1 en utilisant cette linéarisation, comme un programme linéaire. On obtient donc, le PL suivant :
Pour $n = 2$ agents et $p = 5$ objets, l'objectif est de sélectionner trois objets parmi 5 de manière à satisfaire équitables les deux agents en connaissant leur utilité respective des objets.
Commençons tout d'abord par calculer nos w'_k :

— $w'_1 = w_1 + w_2 = 2 - 1 = 1$

— $w'_2 = w_2 = 1$

Il suffit ensuite, de transposer nos données avec ce qu'on a pour obtenir le PL recherché que voici :

$$Max \ r_1 + 2r_2 - (b_{1,1} + b_{2,1} + b_{1,2} + b_{2,2})$$

$$\begin{cases} r_k - b_{ik} \leq z(i) \\ z(i) = \sum_{j=1}^{5} u_{ij} x_j \\ \sum_{j=1}^{5} x_j = 3 \\ x_j \in \{0, 1\} \quad r_k \in \mathbb{R} \ \ b_{ik} \geq 0 \ \ i, k = 1, 2 \end{cases}$$

Nous résolvons ce problème à l'aide du solveur Gurobi et obtenons pour finir les résultats suivants :

— La solution optimale du problème est : $x = [0, 1, 1, 1, 0]$

— La valeur optimale du problème est $z = [18, 16]$

Ainsi, la solution optimale équitable du problème serait de sélectionner les objets 2,3 et 4 afin d'avoir une satisfaction max de 16 et 18 pour l'agent 1 et 2 respectivement.
A noter que pour vérifier la solution obtenue, nous avons personnellement listé les 10 combinaisons d'objets possibles pour trouver la même solution. Elle était en effet la meilleure au sens de Pareto/Dominance de Pareto du vecteur de Lorenz associé.

Chapitre 2

Application au partage équitable de biens indivisibles

On considère un problème où n individus doivent se partager p objets indivisibles tels que $p \geq n$, de valeur connue. Chaque individu ayant sa propre idée de l'utilité des objets pour lui.
On suppose connue la matrice U des utilités à n lignes et p colonnes, où chaque élément donne l'utilité de l'objet j pour l'agent i.
Ces utilités sont supposées additives sur les objets (pour chaque individu i, l'utilité d'un ensemble d'objets qu'il reçoit est égale à la somme des utilités des objets de l'ensemble).
On souhaite répartir les p objets équitablement entre les n individus et pour cela on cherche l'affectation x des objets aux individus qui maximise $f(x)$ tel que : $f(x) = \sum_{i=1}^{n} w_i z_i(x)$.
Les z_i dans cet exercice représentent l'utilité de l'ensemble des objets affectés à l'individu i pour un vecteur poids donnée w à composantes strictement décroissantes.

A. Formulons ce problème comme un programme linéaire en variables mixtes
D'après la partie 1 du projet, nous pouvons aisément écrire le PL comme suit :

$$\begin{cases} max \sum_{k=1}^{n} w'_k(kr_k - \sum_{i=1}^{n} b_{ik}) \\ r_k - b_{ik} \leq z_i \ i = 1, .., n \\ z_i = \sum_{j=1}^{p} u_{ij} x_{ij} \\ \sum_{i=1}^{n} x_{ij} \leq 1 \\ b_{ik} \geq 0, \ i = 1, .., n \\ x_{ij}\{0, 1\} \end{cases}$$

Explication des contraintes du PL :

- $r_k - b_{ik} \leq z_i$: il s'agit tout simplement de la contrainte obtenue à partir de la linéarisation du PL.
- $z_i = \sum_{j=1}^{p} u_{ij} x_{ij}$: fait référence au fait que les $z_i(x)$ représentent l'utilité de l'ensemble des objets affectés à l'individu i, l'utilité étant additive.
- $\sum_{i=1}^{n} x_{ij} \leq 1$: Chaque objet j doit être affecté à au plus un individu (un objet ne peut pas être attribué à plus de 1 individu).

A noter que x_{ij} vaut 1 si l'objet j a été affecté à l'individu i et 0 sinon.

Utilisons à présent cette formulation pour résoudre le problème de partage équitable de $p = 6$ objets sur $n = 3$ individus donné dans l'article [**3**].
Le PL à résoudre est donc le suivant :

$$\begin{cases} max \sum_{k=1}^{3} w'_k(kr_k - \sum_{i=1}^{3} b_{ik}) \\ r_k - b_{ik} \leq z_i \ i = 1, .., 3 \\ z_i = \sum_{j=1}^{6} u_{ij}x_{ij} \\ \sum_{i=1}^{3} x_{ij} \leq 1 \\ b_{ik} \geq 0, \ i, k = 1, .., 3 \\ x_{ij}\{0, 1\} \end{cases}$$

En ce qui concerne le vecteur poids utilisé, nous obtiendrons les vecteurs suivants :

— Pour $w = (3, 2, 1) \longrightarrow w'_k = (1, 1, 1)$

— Pour $w = (10, 3, 1) \longrightarrow w'_k = (7, 2, 1)$

Les solutions obtenues par le solveur Gurobi sont donc les suivantes :
Pour $w = (3, 2, 1) \longrightarrow w'_k = (1, 1, 1)$:
La solution qui maximise f consiste à attribuer l'objet 1 à l'individu 1, les objets 2 et 4 à l'individu 3 et enfin le reste à l'individu 3 et conduit à un vecteur d'utilité $z = (325, 340, 335)$, qui répartit plutôt équitablement les 1000 €

Pour $w = (10, 3, 1) \longrightarrow w'_k = (7, 2, 1)$:
La solution qui maximise f consiste à attribuer l'objet 1 à l'individu 1, les objets 3,5 et 6 à l'individu 2 et le reste à l'individu 3 et conduit à un vecteur d'utilité $z = (325, 335, 340)$, qui répartit également plutôt équitablement les 1000 €

Nous remarquons que quelque soit le vecteur poids, le vecteur de coût associé à la solution optimale reste équitable (il s'agit simplement de permutations de l'affectation des objets vu que l'utilité des individus vis-à-vis des objets est parfaitement égale, les utilités sont en effet indépendantes).

Maximisons à présent la satisfaction moyenne des individus définie comme ceci : $\frac{1}{3}(z_1(x) + z_2(x) + z_3(x))$
Ainsi, en utilisant le solveur Gurobi, la solution qui maximise la satisfaction moyenne des individus consiste à attribuer la moitié des objets à l'individu 2 et l'autre moitié à

l'individu 3 (le premier n'a donc aucun objet attribué) et conduit à un vecteur d'utilité $z = (0, 760, 240)$, avec une répartition très inéquitables des satisfactions !.
Contrairement en effet aux précédentes solutions obtenues en utilisant la fonction f.

A noter que les résultats changent lorsque nous ajoutons la contrainte de l'état dans notre PL (d'après l'exemple dans l'article, l'état doit avoir au moins 10% de la valeur totale du lôt, nous devons rajouter donc la contrainte suivante $\sum_{i=1}^{3} z_i \leq 900$).
Nous obtenons ainsi une solution qui consiste à répartir les objets 2 et 6 au premier individu, l'objet 1 au second et enfin les restants au dernier pour un vecteur coût équitable $z = (275, 325, 285)$.
A noter que la maximisation de la satisfaction moyenne des individu dans ce cas donne une solution qui consiste à donner 5 objets au premier individu pour un vecteur coût $z = (885, 0, 0)$ qui n'est absolument pas équitable !

B. On souhaite à présent étudier l'évolution du temps de résolution en fonction de n et p.
Pour $n = 5, 10, 15, 20$ et $p = 5n$ nous allons tirer aléatoirement 10 matrices U à coefficients entiers positifs de taille (n, p) et 10 vecteurs poids w à composantes positives strictement décroissantes, puis nous allons calculer le temps moyen de résolution des 10 instances pour chaque couple (n, p).

Pour cette partie, nous avons considéré le programme linéaire sans la contrainte de l'état.
Étant donné la taille massive des données, nous étions contraints de rajouter une limitation de temps.

(n,p = 5n)	(5,25)	(10,50)	(15,750)	(20,1000)
Temps moyen (s)	0.02	6	14	30

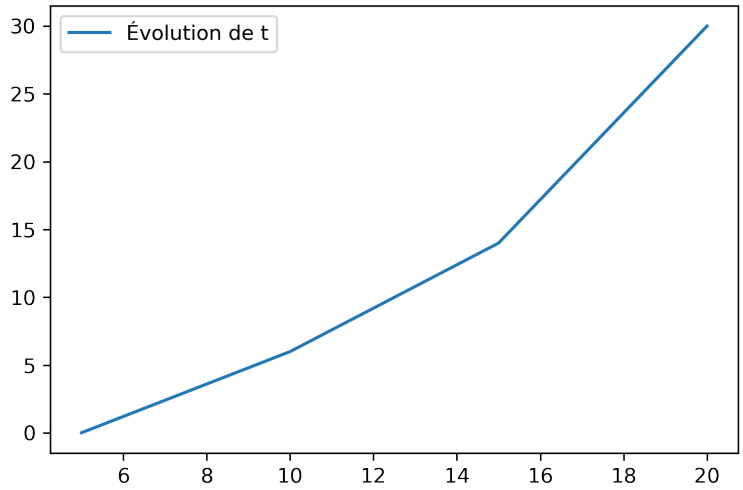

FIGURE 2.1 – Évolution du temps en fonction du couple (n,p) pour b = 100

Commentaire :

Les résultats expérimentaux ci-dessous montrent bien que le temps d'exécution de la résolution des problèmes d'affection équitable dépend fortement de la taille des entrées (n, p).

En effet, plus la taille des entrées augmentent, plus les données sont nombreuses et plus le temps de résolution augmente.

Chapitre 3

Application à la sélection multicritère de projets

Nous cherchons dans cette partie, à sélectionner des projets parmi une liste de p projets soumis.
Chaque projet a un coût connu c_k et le coût total de l'ensemble sélectionné ne doit pas dépasser l'enveloppe budgétaire fixée à $b = \frac{1}{2}(\sum_{k=1}^{p} c_k)$.
Les projets sont évalués quant à leur aptitude à satisfaire plusieurs objectifs de l'entreprise.
On note u_{ij} l'utilité du projet j pour satisfaire l'objectif i (l'utilité est supposée additive) et l'aptitude d'un ensemble de projets x à satisfaire l'objectif i, $z(i)$ est définie comme la somme des utilités u_{ij} des projets j sélectionnés.
L'objectif est donc de trouver une solution équilibrée entre la satisfaction des différents objectifs.
On cherche là aussi, le sous ensemble x qui maximise $f(x)$ pour un vecteur poids donné w à composantes strictement décroissantes.

A. Formulons ce problème comme un programme linéaire en variables mixtes :
Toujours d'après la linéarisation de la fonction objectif vue en Partie 1, nous pouvons aisément écrire le PL comme suit :

$$\begin{cases} max \sum_{k=1}^{n} w'_k(kr_k - \sum_{i=1}^{n} b_{ik}) \\ r_k - b_{ik} \leq z_i \ \ i = 1, .., n \\ z_i = \sum_{j=1}^{p} u_{ij} x_j \\ \sum_{k=1}^{p} c_k x_k \leq b \\ b_{ik} \geq 0, \ \ i = 1, .., n \\ x_k \in \{0, 1\} \end{cases}$$

Explication des contraintes du PL :

- $r_k - b_{ik} \leq z_i$: il s'agit tout simplement de la contrainte obtenue à partir de la linéarisation du PL.
- $z_i = \sum_{j=1}^{p} u_{ij} x_j$: fait référence au fait que les $z_i(x)$ représentent la satisfaction du

chef de service i vis-à-vis de la sélection x, l'utilité étant additive.

— $\sum_{k=1}^{p} c_k x_k \leq b$: il s'agit de la modélisation de la contrainte budgétaire.

À noter que x_k vaut 1 si l'on sélectionne ce projet et 0 sinon.

Utilisons cette formulation pour résoudre l'instance donnée dans l'article [**3**]. Il s'agit d'un problème bi-objectif de sélections de projets parmi $p = 4$ sous une contrainte de budget $b = 100$ et $n = 2$ (car nous avons deux chefs de projets).
Le PL à résoudre est donc le suivant :

$$\begin{cases} max \sum_{k=1}^{2} w'_k(kr_k - \sum_{i=1}^{2} b_{ik}) \\ r_k - b_{ik} \leq z_i \ i = 1, 2 \\ z_i = \sum_{j=1}^{4} u_{ij} x_j \\ \sum_{k=1}^{4} c_k x_k \leq 100 \\ b_{ik} \geq 0, \ i = 1, 2 \\ x_k \in \{0, 1\} \end{cases}$$

Les solutions obtenues par le solveur Gurobi sont les suivantes :
Pour $w = (2, 1) \longrightarrow w'_k = (1, 1)$ **:**
La solution qui maximise f est $x = (1, 0, 0, 1)$. Elle consiste à financer les projets 1 et 4 et conduit à un vecteur d'utilité $(z_1, z_2) = (21, 20)$, qui est bien équilibré.
Pour $w = (10, 1) \longrightarrow w'_k = (9, 1)$ **:**
La solution qui maximise f est $x = (1, 0, 0, 1)$. Elle consiste à financer les projets 1 et 4 et conduit à un vecteur d'utilité $(z_1, z_2) = (21, 20)$, qui est bien équilibré.

Comme nous pouvons le voir, les résultats obtenus sont exactement identiques. En effet, étant donné que le premier chef de projet (agent 1) a tout le temps le plus de poids dans les deux vecteurs w sélectionnés (celle du second restant inchangée), alors la solution restera la même dans les deux cas.

Maximisons à présent la satisfaction moyenne des objectifs définie par $\frac{1}{2}(z_1(x)+z_2(x))$.
En utilisant toujours le solveur Gurobi, nous obtenons les résultats suivants :
La solution qui maximise la satisfaction moyenne est $x = (1, 0, 1, 0)$. Elle consiste à financer les projets 1 et 3 et conduit à un vecteur $(z_1, z_2) = (36, 6)$, avec une répartition très inéquitable des satisfactions !.
Contrairement en effet aux précédentes solutions obtenues en utilisant la fonction f.

B. On souhaite à présent, étudier l'évolution du temps de résolution en fonction de n et p.
Dans cette partie, nous allons tirer aléatoirement 10 matrices U à coefficients entiers positifs de taille (n, p) donnée, des coûts c_k positifs, puis calculer le temps moyen de résolution des 10 instances pour chaque couple (n, p).

Pour cette partie, nous avons considéré deux cas pour éviter toute confusion.
Nous allons dans un premier temps procéder au travail demandé en considérant b fixe tel que b = 100, ainsi nous obtenons les résultats suivants :

(n,p)	(2,5)	(5,10)	(10,15)	(15,20)
Temps moyen (s)	0.01	0.03	0.049	0.06

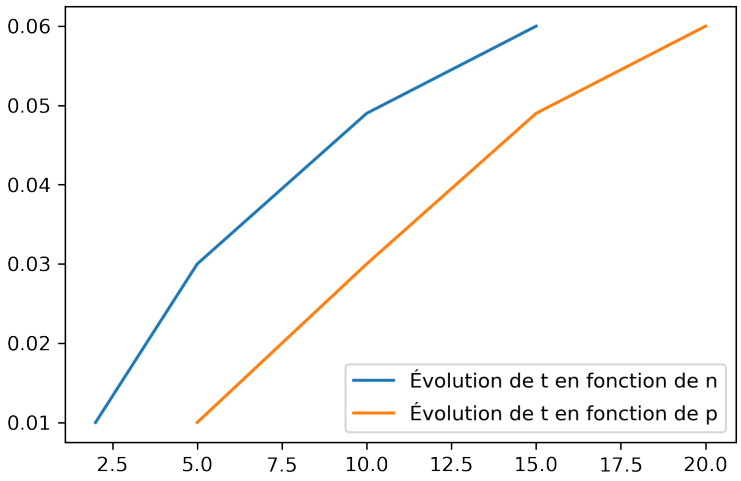

FIGURE 3.1 – Évolution du temps en fonction du couple (n,p) pour b = 100

Dans un second temps nous allons prendre en compte lors de la génération aléatoire de nos vecteurs/matrices, la contrainte annoncée dans l'énoncé du projet suivante : $b = \frac{1}{2}(\sum_{k=1}^{p} c_k)$. Nous obtenons par conséquent les résultats suivants :

(n,p)	(2,5)	(5,10)	(10,15)	(15,20)
Temps moyen (s)	0.018	0.022	0.044	0.075

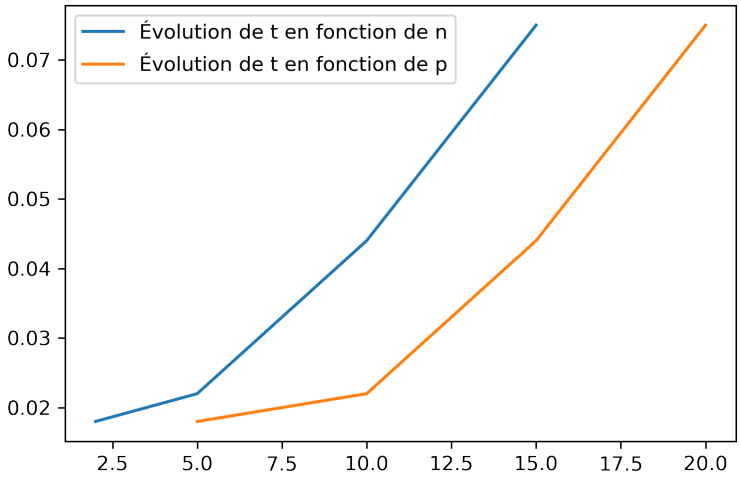

FIGURE 3.2 – Évolution du temps en fonction du couple (n,p) pour b soumis à la contrainte

Commentaire :

Nous pouvons aisément à l'aide de ces résultats expérimentaux, conclure que dans tous les cas, le temps de résolution des problèmes de sélection multicritère de projets dépend intrinsèquement de la taille des paramètres en entrées, à savoir n et p.

Ainsi pour le couple (n, p) est grand, c'est-à-dire plus le nombre d'agents ainsi que le nombre de projet augmente, plus la taille d'exécution de la résolution augmente également. Ce qui est parfaitement représenté par les graphes ci-dessus (fig 3.1 et 3.2).

Chapitre 4

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

---

On se place dans un contexte de planification d'itinéraires dans l'incertain. Dans un réseau modélisé par un graphe orienté on recherche un chemin rapide pour aller d'un sommet initial à un sommet destination.
Le problème est compliqué par le fait que plusieurs scénarios sont envisagés sur les temps de transport dans ce réseau. Plus précisément on considère un ensemble $S = \{1, ..., n\}$ de scénarios possibles et le temps pour parcourir chaque arc (i, j) du graphe dans les différents scénarios est donné par le vecteur $(t_{ij}^1, ..., t_{ij}^n)$ où t_{ij}^s représente le temps nécessaire pour parcourir l'arc (i, j) dans le scénario s pour tout $s \in S$. Les temps sont additifs le long d'un chemin ce qui signifie que le temps pour parcourir un chemin P dans le scénario s est défini par $t^s(P) = \sum_{(i,j)\in P} t_{ij}^s$.

A. Étant donné un graphe G orienté, formulons le programme linéaire qui permet de calculer le chemin le plus rapide dans un scénario donné (on s'inspirera du problème de la recherche d'un flot max à coût minimum).
On commence par définir les variables de décision de notre PL que voici :
Variables d'arcs :

$$x_{ij} = \begin{cases} 1 & si\ on\ prend\ \ l'arc\ \ (i,j) \\ 0 & sinon \end{cases}$$

Pour la fonction objectif, on minimise le temps pour parcourir un chemin P dans le scénario s.
Les contraintes quant à elles, sont inspirées du programme de recherche d'un flot max à coût minimum. Nous obtenons pour finir le PL suivant :

$$\begin{cases} Min\ \ t^s(P)x_{ij} = \sum_{(i,j)} t_{ij}^s x_{ij} \\ \\ \sum_{(i,j)} x_{ij} - \sum_{(j,i)} x_{ji} = \begin{cases} 1\ si\ \ i = s\ (sommet\ source) \\ -1\ \ si\ \ i = t\ (sommet\ puit) \\ 0\ \ si\ \ i \neq s, t \end{cases} \end{cases}$$

En d'autre termes pour chaque sommet, la somme des poids de ses arcs entrants doit être égale à la somme des poids de ses arcs sortants (ce qu'on appelle la contrainte de

conservation du flôt), sauf dans le cas où le sommet est la source ou le puit du graphe (comme mentionné dans le PL).
Application au graphe donné dans l'énoncé :
Pour le scénario 1 - $t^1 = (5, 10, 2, 4, 1, 4, 3, 1, 1, 3, 1, 1)$

$$\begin{cases} Min \sum_{(i,j)} t^1_{ij} x_{ij} \\ x_{1,2} + x_{1,3} + x_{1,4} = 1 \\ x_{2,5} + x_{2,3} + x_{2,4} - x_{1,2} = 0 \\ x_{3,5} + x_{3,6} - x_{2,3} - x_{1,3} - x_{4,3} = 0 \\ x_{4,3} + x_{4,6} - x_{1,4} - x_{2,4} = 0 \\ x_{5,7} - x_{2,5} - x_{3,5} = 0 \\ x_{6,7} - x_{3,6} - x_{4,6} = 0 \\ -x_{5,7} - x_{6,7} = -1 \\ x_{ij} \in \{0, 1\} \end{cases}$$

Pour le scénario 2 - $t^2 = (3, 4, 6, 2, 3, 6, 1, 2, 4, 5, 1, 1)$

$$\begin{cases} Min \sum_{(i,j)} t^2_{ij} x_{ij} \\ x_{1,2} + x_{1,3} + x_{1,4} = 1 \\ x_{2,5} + x_{2,3} + x_{2,4} - x_{1,2} = 0 \\ x_{3,5} + x_{3,6} - x_{2,3} - x_{1,3} - x_{4,3} = 0 \\ x_{4,3} + x_{4,6} - x_{1,4} - x_{2,4} = 0 \\ x_{5,7} - x_{2,5} - x_{3,5} = 0 \\ x_{6,7} - x_{3,6} - x_{4,6} = 0 \\ -x_{5,7} - x_{6,7} = -1 \\ x_{ij} \in \{0, 1\} \end{cases}$$

Où $(1, 2, 3, 4, 5, 6, 7)$ correspond respectivement aux sommets (A,B,C,D,E,F,G)

A noter que la dernière contrainte du PL est redondante (elle peut s'écrire comme étant une combinaison linéaires des autres contraintes), il est donc possible de la supprimer (c'est ce que nous avons fait au cours de nos implémentations).

Résolvons ces deux programmes linéaires à l'aide de Gurobi, l'objectif étant de déterminer les chemins les plus rapides de a à g dans chacun des deux scénarios.

Pour le premier scénario :
La solution qui minimise la fonction objectif consiste à emprunter le chemin $a \longrightarrow d \longrightarrow c \longrightarrow f \longrightarrow g$ de valeur optimale $t^* = 5$.

Pour le second scénario :
La solution qui minimise la fonction objectif consiste à emprunter le chemin $a \longrightarrow c \longrightarrow e \longrightarrow g$ de valeur optimale $t^* = 6$.

B. Ne sachant pas quel scénario va se produire et ne connaissant même pas leur probabilité, nous allons chercher un chemin robuste, qui reste autrement dit rapide dans les différents scénarios. Pour cela on cherche un chemin P qui maximise $v_f(P) = f(-t^1(P), ..., -t^n(P))$. Proposons un programme linéaire pour calculer un tel chemin.

Le programme linéaire que nous souhaitons écrire est un programme robuste inspiré du PL obtenu dans la partie 1 du projet.
En effet en reprenant les résultats obtenus, il suffira tout simplement de les adapter en fonction de ce qui est demandé dans l'énoncé. Autrement dit, nous allons remplacer z_i initialement présent dans le Pl linéarisé de la question 4 - Partie 1 par $-z_i$ (nous souhaitons effectivement maximiser $v_f(P) = f(-t^1(P), -t^2(P))$ pour les deux scénarios).
Ainsi nous obtenons :

$$\begin{cases} Max \sum_{k=1}^{n} w'_k(kr_k - \sum_{i=1}^{n} b_{ik}) \\ r_k - b_{ik} \leq -z_i(x) \\ x \in X \\ r_k \in \mathbb{R} \;\; b_{ik} \geq 0 \;\; i = 1, .., n \end{cases}$$

En l'appliquant au graphe de l'énoncé, nous obtenons pour finir, le programme linéaire robuste suivant ($n = 2$ car nous avons deux scénarios) :

$$\begin{cases} Max \sum_{k=1}^{2} w'_k(kr_k - \sum_{i=1}^{2} b_{ik}) \\ r_k - b_{ik} \leq -z_i(x) \\ z_i(x) = \sum_{(e \in E)} t_e^s x_e \\ x \in X \;\; contraintes\ de\ conservation\ du\ flôt \\ x_e \in \{0, 1\} \;\; r_k \in \mathbb{R} \;\; b_{ik} \geq 0 \;\; i, s, k = 1, 2 \end{cases}$$

tel que $e \in E$ représente un arc appartenant à l'ensemble des arcs du graphe.

En utilisant le solveur Gurobi, nous pouvons constater que la solution qui maximise $v_f(P)$ est $x = (1,0,0,1,0,0,0,0,0,0,1,0)$. Elle consiste à emprunter le chemin $a \longrightarrow b \longrightarrow e \longrightarrow g$ et conduit à un vecteur de coût $(z_1, z_2) = (10, 10)$, qui est bien équilibré.

C. On souhaite étudier l'impact de la pondération w sur la robustesse de la solution proposée.
Pour cela, nous utilisons une famille de vecteurs de pondération $w(\alpha) \in \mathbb{R}^n$ définie pour tout $\alpha \geq 1$ par les poids :

$$w_i(\alpha) = (\frac{n-i+1}{n})^\alpha - (\frac{n-i}{n})^\alpha \quad , \quad i = 1, .., n$$

Pour simplifier, on se placera dans le cas $n = 2$ et on tirera aléatoirement 20 fois des temps de parcours t^1_{ij} et t^2_{ij} pour les arcs (i, j) du graphe de l'énoncé.
Pour chacune des 20 instances ainsi générées on calculera un chemin qui maximise v_f pour le vecteur $w(1)$ autrement dit $\alpha = 1$ et on réprésentera les 20 solutions ainsi obtenues dans le plan (t^1, t^2).
On recommencera le même graphique sur les mêmes instances en optimisant f pour $\alpha = 2, 3, 4, 5$.

Pour $\alpha = 1$ **:** $\begin{cases} w_1(1) = \dfrac{2-1+1}{2} - \dfrac{2-1}{2} \\ w_2(1) = \dfrac{2-2+1}{2} - \dfrac{2-2}{2} \end{cases}$

D'où $w = (\frac{1}{2}, \frac{1}{2}) \longrightarrow w' = (0, \frac{1}{2})$

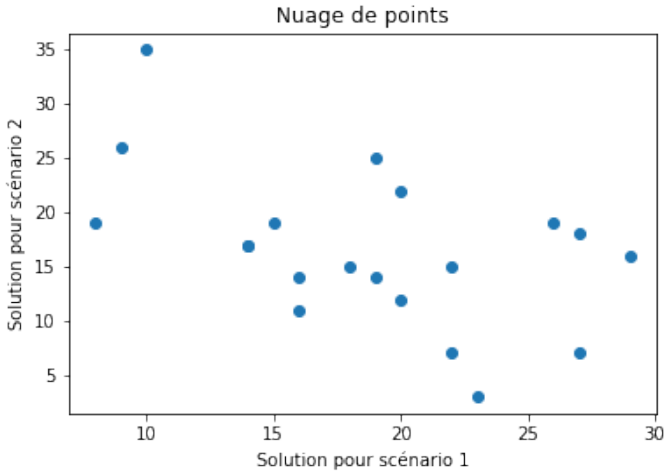

FIGURE 4.1 – Nuage de points pour $\alpha = 1$

Pour $\alpha = 2$ **:** $\begin{cases} w_1(2) = (\frac{2-1+1}{2})^2 - (\frac{2-1}{2})^2 \\ w_2(2) = (\frac{2-2+1}{2})^2 - (\frac{2-2}{2})^2 \end{cases}$

D'où $w = (\frac{3}{4}, \frac{1}{4}) \longrightarrow w' = (\frac{1}{2}, \frac{1}{4})$

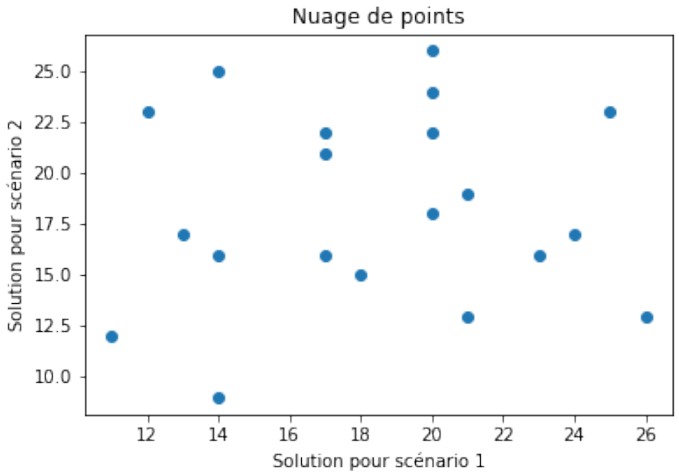

FIGURE 4.2 – Nuage de points pour $\alpha = 2$

Pour $\alpha = 3$ **:** $\begin{cases} w_1(3) = (\frac{2-1+1}{2})^3 - (\frac{2-1}{2})^3 \\ w_2(3) = (\frac{2-2+1}{2})^3 - (\frac{2-2}{2})^3 \end{cases}$

D'où $w = (\frac{7}{8}, \frac{1}{8}) \longrightarrow w' = (\frac{3}{4}, \frac{1}{8})$

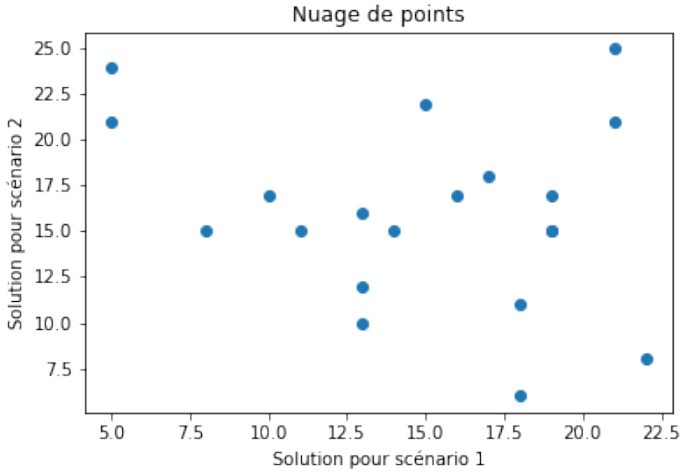

FIGURE 4.3 – Nuage de points pour $\alpha = 3$

Pour $\alpha = 4$ **:** $\begin{cases} w_1(4) = (\frac{2-1+1}{2})^4 - (\frac{2-1}{2})^4 \\ w_2(4) = (\frac{2-2+1}{2})^4 - (\frac{2-2}{2})^4 \end{cases}$

D'où $w = (\frac{15}{16}, \frac{1}{16}) \longrightarrow w' = (\frac{7}{8}, \frac{1}{16})$

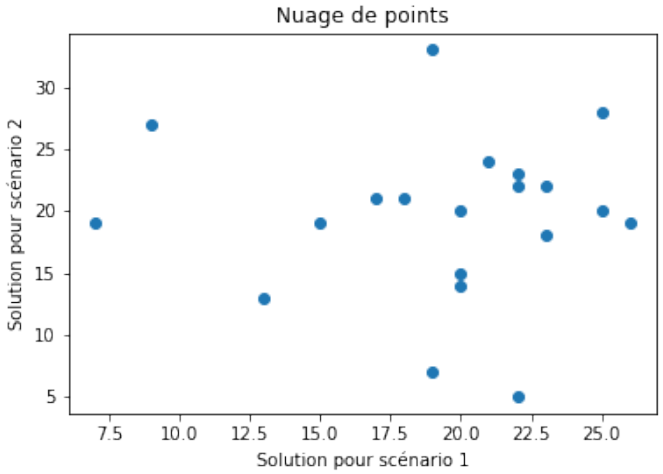

FIGURE 4.4 – Nuage de points pour $\alpha = 4$

Pour $\alpha = 5$ **:** $\begin{cases} w_1(5) = (\frac{2-1+1}{2})^5 - (\frac{2-1}{2})^5 \\ w_2(5) = (\frac{2-2+1}{2})^5 - (\frac{2-2}{2})^5 \end{cases}$

D'où $w = (\frac{31}{32}, \frac{1}{32}) \longrightarrow w' = (\frac{15}{16}, \frac{1}{32})$

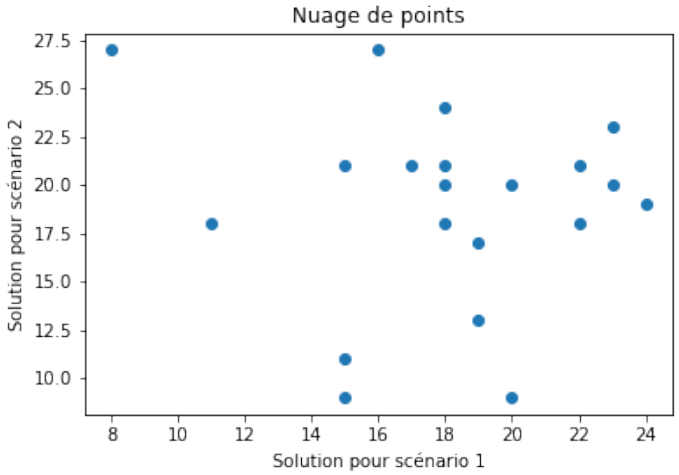

FIGURE 4.5 – Nuage de points pour $\alpha = 5$

En étudiant les différents nuages de points générés par notre étude expérimentale, nous pouvons voir que la modification du vecteur poids n'a pas de réel impact sur les solutions. Les points sont effectivement dispersés et souvent confondus, d'où notre incapacité à déduire une quelconque corrélation entre les pondérations (vecteurs poids associés) et la robustesse des solutions.

Conclusion générale

L'optimisation équitable consiste comme nous avons pu le voir à travers ce projet, à trouver des solutions efficaces en apportant un soin particulier à préserver l'équilibre entre la satisfactions des différents agents considérés.
Les problèmes d'optimisation combinatoire multiagents peuvent ainsi naturellement, se formuler comme des problèmes d'optimisation multicritères dans lesquels chaque fonction considérée représente le point de vue d'un agent [**1**].
Nous avons vu toutefois, que l'optimisation d'une simple combinaison linéaire des critères ne convient pas pour rendre compte de l'équité, étant donnée la nature non-linéaire des fonctions d'agrégations.

C'est pour cette raison que nous avons débuté par linéariser la fonction objectif afin d'obtenir un programme linéaire à variable mixte, à résolution rapide et facile à l'aide du solveur étudié en cours "Gurobi".

Nous avons par la suite, traité le problème d'optimisation équitable sous ses différentes formes à travers l'analyse théorique et expérimentale des problèmes de partage équitable de biens indivisibles, de sélection multicritère de projets et enfin de recherche d'un chemin robuste dans un graphe.

Bibliographie

[1] J.Lesca, P.Perny - "Optimisation multiagent équitable : une approche utilisant la programmation linéaire mixte", LIP6-UPMC.

[2] Énoncé projet MOGPL : optimisation équitable

[3] P.Perny "L'optimisation équitable", Tangente, Hors série n°75