Rapport de projet MOGPL
Master d'informatique

Optimisation équitable

Rédigé par :

AROUI Mohammed

PIGEON Nolwenn

Année universitaire 2022/2023 - Semestre 1

TABLE DES MATIÈRES

Partie I

L'objectif de ce projet consiste à trouver une solution efficace équitable c'est-à dire une solution que l'on ne peut améliorer sur une composante sans dégrader l'efficacité d'une autre composante et qui conduit à un vecteur z(x) équilibré. On traitera différents points de vue tels que la décision multicritères, la décision multiagents et la décision dans l'incertain.

Pour obtenir une telle solution efficace équitable, on cherche donc $x \in X$ qui maximise la fonction

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

1 Linéarisation de f

1.1 Programme en variables 0-1 et valeur optimale $L_k(z)$

$\forall(z) \in \mathbb{R}$, on note $L(z)$ le vecteur $(L_1(z), ...L_n(z))$ tel que la composante $L_k(z) = \sum_{i=1}^{k}(z_{(i)})$.
On a le programme linéaire en variables 0-1 suivant:

$$\begin{array}{ll} \text{min} & \displaystyle\sum_{i=1}^{n} a_{ik} z_i \\ \text{s.c.} & \left\{ \begin{array}{c} \sum_{i=1}^{n} a_{ik} = k \\ a_{ik} \in \{0,1\}, i = 1, ...n \end{array} \right. \end{array} \tag{2}$$

Le vecteur Lorenz associé à z défini, $L_k(z_1, z_2, ...z_n) = (z_{(1)}, z_{(1)} + z_{(2)}, ..., z_{(1)} + z_{(2)} + ... + z_{(n)})$ où $z_{(i)}$ est la i^{me} plus petite composante du vecteur z trié par ordre croissant au préalable.

La valeur optimale de ce programme est la somme des k plus petites composantes de z soit $L_k(z)$

1.2 Programme en valeurs continues et dual D_k

Après avoir admis que le programme ci-dessus peut être relaxé en variables continues, on en déduit que $L_k(z)$ est aussi la valeur à l'optimum du programme linéaire suivant:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} a_{ik} z_i \\ \text{s.c.} \quad & \begin{cases} \sum_{i=1}^{n} a_{ik} = k \\ a_{ik} \leq 1 \\ a_{ik} \geq 0, i = 1, ...n \end{cases} \end{aligned} \tag{3}$$

Le dual D_k de ce programme linéaire peut s'écrire comme ci-dessous:

$$\begin{aligned} \max z = \quad & k r_k - \sum_{i=1}^{k} b_{ik} \quad \forall (k) \in \{1, ..., n\} \\ \text{s.c.} \quad & \begin{cases} r_k - b_{ik} \leq z_i \quad \forall (i) \in (1, ..., p) \\ b_{ik} \geq 0 \quad \forall (i) \in (1, ..., p) \end{cases} \end{aligned} \tag{4}$$

Via $Guribi$ on utilise cette formulation duale pour calculer par programmation linéaire les composantes du vecteur $L(4, 7, 1, 3, 9, 2)$. On obtient $\{1.0, 3.0, 6.0, 10.0, 17.0, 26.0\}$

1.3 Critère fondamental

D'après le support, il faut définir $f(x)$ par le produit scalaire $w'_k L_k(z(x))$. Ce critère est fondamental pour comprendre que la maximisation de f capture une idée d'équité, en visant à accroître les dotations de chacun.

On veut montrer que

$$f(x) = \sum_{k=1}^{n} w'_k L_k(z(x)) \tag{5}$$

D'après nos hypothèses de départ, on sait que la fonction f est telle que l'équation 1

Or, $z_1 = L_1(z)$ et $z_n = L_n(z) - L_{n-1}(z)$ pour $n > 1$

Donc

$$\begin{aligned} f(x) &= w_1 L_1(z) + \sum_{i=2}^{n} w_i z_i \\ &= w_1 L_1(z) + w_2(L_2(z) - L_1(z)) + ... + w_{n-1}(L_{n-1}(z) - L_{n-2}(z)) + w_n(L_n(z) - L_{n-1}(z)) \\ &= L_1(z)(w_1 - w_2) + L_2(z)(w_2 - w_3) + ... + L_{n-1}(z)(w_{n-1} - w_n) + L_n(z)w_n \\ &= \sum_{k=1}^{n-1}(w_k - w_{k+1})L_k(z) + w_n L_n(z) \\ &= \sum_{k=1}^{n} w'_k L_k(z) \end{aligned} \tag{6}$$

avec $w'_k = (w_k - w_{k+1})$ pour $k = 1, ..., n$ et $w'n = w_n\ w'$ a toutes ses composantes positives puisque w a ses composantes décroissantes.

1.4 Résolution de l'instance donnée

D'après, les résultats précédents, la maximisation de $f(x)$ sur un ensemble X décrit par des contraintes linéaires peut s'écrire sous la forme du programme linéaire suivant pour notre exemple :

$$\begin{aligned} \text{max} \quad & \sum_{k=1}^{n} w'_k(kr_k - \sum_{i=1}^{n} b_{ik}) \\ \text{s.c.} \quad & \begin{cases} r_k - b_{ik} \leq \sum_{t=1}^{5} x_t u_{i,t} i, = 1, ..., n \\ \sum_{t=1}^{5} x_t = 3 \\ b_{ik} \geq 0, i = 1, ..., n \\ x_t \in \{0, 1\}, t = 1, \ldots, 5 \end{cases} \end{aligned} \tag{7}$$

Nous avons implémenté ce PL en utilisant gurobi avec $w = (2, 1)$ et la matrice d'utilité U.

$$U = \begin{pmatrix} 5, 6, 4, 8, 1 \\ 3, 8, 6, 2, 5 \end{pmatrix}$$

La solution que nous obtenons est $x = (0, 1, 1, 1, 0)$ avec un vecteur d'utilité $(z_1, z_2) = (18, 16)$ qui est bien équilibré.

Partie II

2 Application au partage équitable de biens indivisibles

2.1 PL du problème

Dans cette partie on s'intéresse au problème où n individus doivent partager $p \geq n$ objets indivisibles de valeur connue. On dispose de la matrice U a n lignes et p colonnes, dont l'élément u_{ij} donne l'utilité de l'objet j à l'agent i. Ces utilités sont supposées additives sur les objets c'est-à-dire que pour chaque individu i, l'utilité d'un ensemble d'objets qu'il reçoit est la somme des utilités des objets de l'ensemble. Ainsi, le but est de maximiser une fonction linéaire globale du type $z_i = \sum_{ij} u_{ij}x_{ij}$ où x_{ij} représente une variable de décision booléenne qui vaut 1 (resp. 0) si on affecte (resp. on n'affecte pas) l'objet j à l'individu i.

Pour un partage équitable nous utiliserons le même programme linéaire que nous avons étudié dans la première section, donc , on maximie $f(x) = \sum_{i=1}^{n} w_i z_i$ où $z_i = \sum_{ij} u_{ij}x_{ij}$ pour un vecteur poids w donné, à composantes strictement décroissantes. Nous ajoutons une contrainte pour s'assurer qu'un objet j doit être assigné à un seul individu i : $\sum_{i=1}^{n} x_{ij} \leq 1 \qquad \forall(j) \in (1, ..., p)$
Le programme s'écrit comme:

$$\begin{array}{ll} \max & \displaystyle\sum_{i=1}^{n} w_i z_i(x) \\ \text{s.c.} & \begin{cases} z_i(x) = \sum_{ij} u_{ij}x_{ij} \\ \sum_{i=1}^{n} x_{ij} \leq 1 \qquad \forall(j) \in (1, ..., p) \\ x_{ij} \in \{0, 1\}, i = 1, ...n, j = 1, ...p \end{cases} \end{array} \tag{8}$$

Puis nous le linéarisons de la même manière qu'à la section 1.4 ce qui nous donne:

$$\begin{array}{ll} \max & \displaystyle\sum_{k=1}^{n} w'_k(kr_k - \sum_{i=1}^{n} b_{ik}) \\ \text{s.c.} & \begin{cases} r_k - b_{ik} \leq \sum_{j=1}^{p} u_{ij}x_{ij} \qquad i, k = 1, ..., n \\ \sum_{i=1}^{n} x_{ij} \leq 1 \qquad \forall(j) \in (1, ..., p) \\ b_{ik} \geq 0, i = 1, ..., n \\ x_{ij} \in \{0, 1\}, i = 1, ...n, j = 1, ...p \end{cases} \end{array} \tag{9}$$

Résolution de problème avec $n = 3$ et $p = 6$: notre matrice d'utilité U est:

$$U = \begin{pmatrix} 325, 225, 210, 115, 75, 50 \\ 325, 225, 210, 115, 75, 50 \\ 325, 225, 210, 115, 75, 50 \end{pmatrix}$$

En maximisant la fonction f avec un vecteur de poids $w = (3, 2, 1)$, cela nous donne la solution suivante: (335,340,325). Avec $w = (10, 3, 1)$ la solution devient (325,335,340) qui reste la même. En maximisant la satisfaction moyenne $(z_1 + z_2 + z_3)/3$ on obtient la solution (0,760,240) qui n'est pas répartie équitablement.

2.2 Tests et résultats

Nous étudions maintenant l'évolution du temps moyen de résolution en fonction de n et p. Pour $n = 2, 5, 10, 15$ et $p = 5n$. Pour cela, nous tirons aléatoirement 10 matrices U à coefficients entiers positifs de taille (n, p) et 10 vecteurs poids w à composantes positives strictement décroissantes, nous calculons le temps moyen de résolution sur 10 instances pour chaque couple (n,p). Nous présentons graphiquement les résultats obtenus:

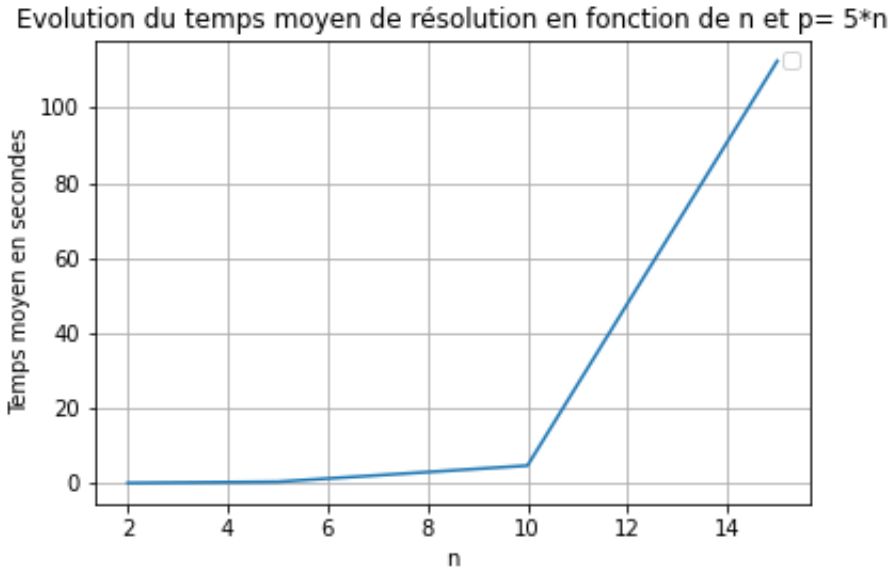


Figure 1: Évolution du temps moyen de résolution sur 10 instances pour n et p =5n

Discussion des résultats obtenus

- D'après la courbe obtenue, nous constatons que le temps moyen de résolution est une fonction croissante du couple (n, p) quand ce dernier augmente le temps moyen augmente nécessairement. Par ailleurs, on observe deux tendances dans cette courbe.

- Les instances de tailles inférieures ou égales à 10, ne dépassent pas les 10 secondes en moyenne sur 10 instances. L'augmentation visible sur la courbe est très légère.
- Tandis qu'avec des instances de tailles supérieures ou égales à 15, cela prend beaucoup de temps à résoudre (100 secondes).

Partie III

3 Application à la sélection multicritères de projets

3.1 PL du problème

Le problème consiste 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 $b = \frac{1}{2}\sum_{k=1}^{p} c_k$. On suppose la matrice U connue à n lignes et p colonnes, dont l'élément u_{ij} donne l'utilité de l'objet j à l'individu i. Ces utilités sont supposées additives, donc la fonction linéaire à maximiser est $z_i(x) = \sum_{j=1}^{p} u_{ij}x_j$ où x_j représente une variable de décision booléenne qui vaut 1 (resp. 0) si on sélectionne (resp. on ne sélectionne pas) le projet j. Pour une solution équitable on maximise la fonction $f(z_1 \dots z_n)$ sous contrainte budgétaire de la forme $\sum_{k=1}^{p} c_k x_k \leq b$. Le programme est le suivant:

$$\begin{array}{ll} \max & \displaystyle\sum_{i=1}^{n} w_i z_i(x) \\ \text{s.c.} & \begin{cases} z_i(x) = \sum_{j=1}^{p} u_{ij}x_j \\ \sum_{k=1}^{p} c_k x_k \leq b \\ x_j \in \{0,1\}, j = 1, ...p \end{cases} \end{array} \tag{10}$$

On linéarise:

$$\begin{array}{ll} \max & \displaystyle\sum_{k=1}^{n} w'_k(kr_k - \sum_{i=1}^{n} b_{ik}) \\ \text{s.c.} & \begin{cases} r_k - b_{ik} \leq \sum_{j=1}^{p} u_{ij}x_j \qquad i,k = 1, ..., n \\ \sum_{k=1}^{p} c_k x_k \leq b \\ b_{ik} \geq 0, i = 1, ..., n \\ x_j \in \{0,1\}, j = 1, ...p \end{cases} \end{array} \tag{11}$$

Résolution de problème avec $n = 2$ et $p = 4$: notre matrice d'utilité U est:

$$U = \begin{pmatrix} 19, 6, 17, 2 \\ 2, 11, 4, 18 \end{pmatrix}$$

et les coûts des projets sont $C = [40, 50, 60, 50]$ et $b = 100$. La maximisation de la fonction f avec un vecteur de poids $w = (2, 1)$ nous donne la solution x(1,0,0,1) avec un vecteur d'utilité $(z_1, z_2) = (21, 20)$ qui est bien équilibrée. Avec $w = (10, 1)$ la solution reste la même. En maximisant la satisfaction moyenne $(z_1 + z_2)/2$ on obtient la solution $x = (1, 0, 1, 0)$ et $(z_1, z_2) = (36, 6)$ qui n'est pas répartie équitablement.

3.2 Tests et résultats

Nous étudions maintenant l'évolution du temps moyen de résolution en fonction de n et p. Pour $n = 2, 5, 10$ et $p = 5, 10, 15, 20$. Pour cela, nous tirons aléatoirement 10 matrices U à coefficients entiers positifs de taille (n, p) ,10 vecteurs poids w et les coûts c_k positifs, nous calculons le temps moyen de résolution, sur 10 instances pour chaque couple (n,p) (on ne fait pas la résolution quand $p < n$). Nous présentons graphiquement les résultats obtenus:

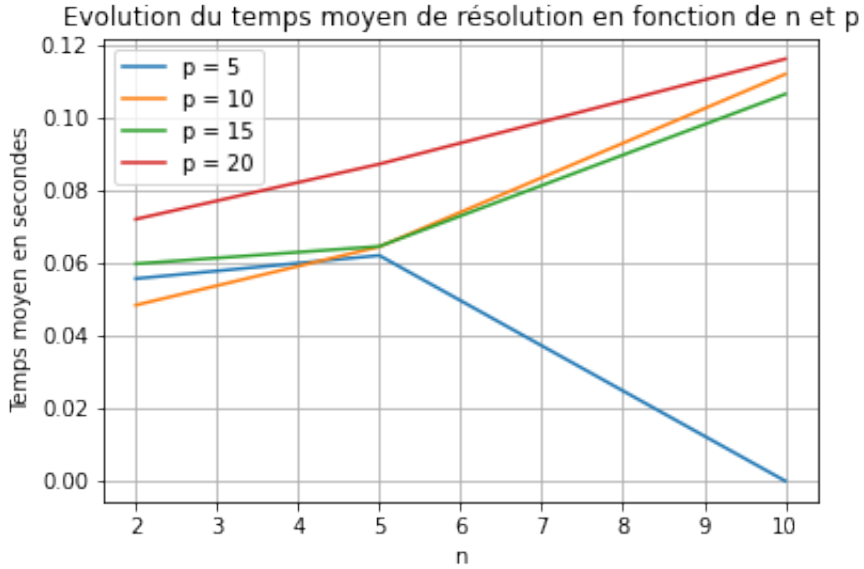


Figure 2: Évolution du temps moyen de résolution sur 10 instances pour n et p

Discussion des résultats obtenus

- Nous constatons que le temps moyen de résolution est une fonction croissante du couple (n, p)quand ce dernier augmente le temps moyen augmente nécessairement

- On constate qu'il n'y a pas de différence apparente dans le temps de résolution pour différentes valeurs de n choisies quand p augmente.
- La résolution reste dans un délai raisonnable lorsque p et n augmentent.

Partie IV

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

4.1 PL du problème du plus court chemin

Etant donné un graphe orienté $G(V, A)$, le problème du plus court chemin peut être formulé par le programme linéaire suivant:

$$\begin{aligned} &\min \quad z^s = \sum_{ij \in A} u^s_{i,j} x_{ij} \text{ avec } x_{ij} \in \{0,1\} \\ &\text{s.c. } \sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = \begin{cases} 1 \text{ si } i = s; \\ -1 \text{ si } i = t; \\ 0 \text{ sinon .} \end{cases} \end{aligned} \tag{12}$$

avec $u^s_{i,j}$ le coût de l'arc $\{i, j\}$ pour le scénario s, $G(V, A)$ le graphe orienté, V l'ensemble des sommets, A l'ensemble des couples de sommets $\{i, j\} \in V^2$, x_{ij} les variables qui indiquent si l'arc (i, j) fait partie ou non du plus court chemin trouvé.

On cherche à choisir l'ensemble des arcs de poids minimal pourvu que les arcs composent un chemin de s à t.

Application:
On doit crée une matrice U^s de taille $(|V|, |V|)$ pour représenter notre graphe avec les les coût d'arcs pour chaque scénario s. nous mettons la diagonale à 0 car elle représente une boucle automatique pour un sommet ,et lorsqu'il n'y a pas de connexion entre deux sommets, on fixe le coût à un grand nombre M.

Voici par exemple matrice U^1 pour le scénario $s = 1$ dans notre exemple:

$$U^1 = \begin{pmatrix} 0,5,10,2,M,M,M \\ M,0,4,1,4,M,M \\ M,M,0,M,3,1,M \\ M,M,1,0,M,3,M \\ M,M,M,M,0,M,1 \\ M,M,M,M,M,0,1 \\ M,M,M,M,M,M,0 \end{pmatrix}$$

Après résolution des programmes linéaires implémentés via $Gurobi$ le plus court chemin du scénario 1 est $\{a,d,f,g\}$ de longueur 5 et le plus court chemin du scénario 2 est $\{a,c,e,g\}$ de longueur 6.

4.2 Recherche du chemin robuste, scénario incertain

Dans cette question on cherche à maximiser $v_f(P) = f(-t^1(P),...,-t^n(P))$
On pose $z = -t^1(P),...,-t^n(P)$ ce qui revient à maximiser $f(x) = \sum_{i=1}^{s} w_i z_i(x)$. On va d'abord linéariser cette fonction comme ce qu'on a fait dans la partie I.

$$\begin{array}{ll} \max & \displaystyle\sum_{i=1}^{s} w_i z_i(x) \\ \text{s.c.} & \left\{ \begin{array}{c} z_i(x) = -\sum_{(m,n)\in A} u^i_{mn} x_{mn} \\ \displaystyle\sum_{n\in V} x_{mn} - \sum_{n\in V} x_{nm} = \left\{ \begin{array}{l} 1 \text{ si } m = s; \\ -1 \text{ si } m = t; \\ 0 \text{ sinon} \end{array} \right. \\ x_{mn} \in \{0,1\}, (m,n) \in A \end{array} \right. \end{array} \tag{13}$$

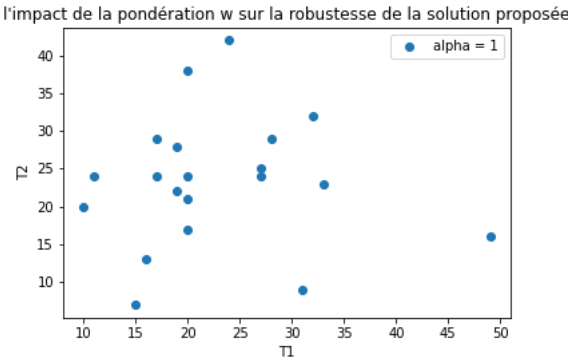
cela devient :

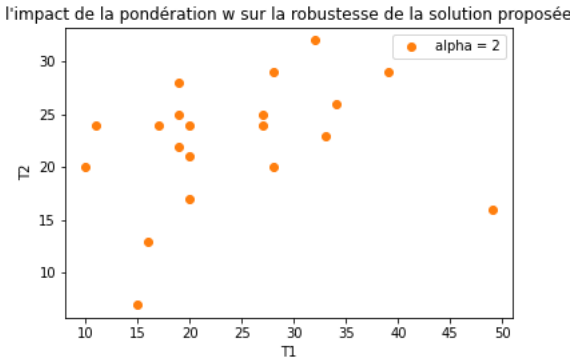
$$\begin{array}{ll} \max & \displaystyle\sum_{k=1}^{s} w'_k (k r_k - \sum_{i=1}^{s} b_{ik}) \\ \text{s.c.} & \left\{ \begin{array}{c} r_k - b_{ik} \leq -\sum_{(m,n)\in A} u^i_{mn} x_{mn} \qquad i,k = 1,...,s \\ \displaystyle\sum_{n\in V} x_{mn} - \sum_{n\in V} x_{nm} = \left\{ \begin{array}{l} 1 \text{ si } m = s; \\ -1 \text{ si } m = t; \\ 0 \text{ sinon} \end{array} \right. \\ b_{ik} \geq 0, i = 1,...,s \\ x_{mn} \in \{0,1\}, (m,n) \in A \end{array} \right. \end{array} \tag{14}$$

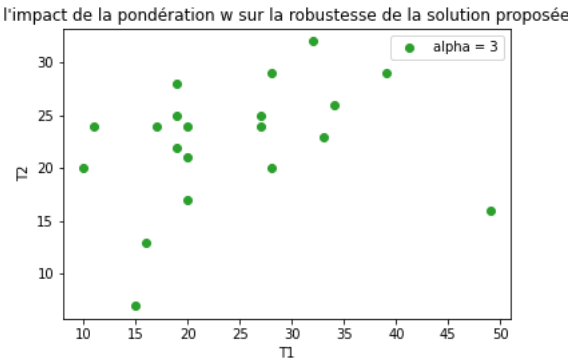
Résolution de l'instance donnée: La résolution de programme linéaire implémenté via $Gurobi$ avec le vecteur $w = (2, 1)$, nous donne le chemin robuste $\{a, b, e, g\}$ avec un coût (10,10) qui est bien équilibrée sur les deux scénarios.

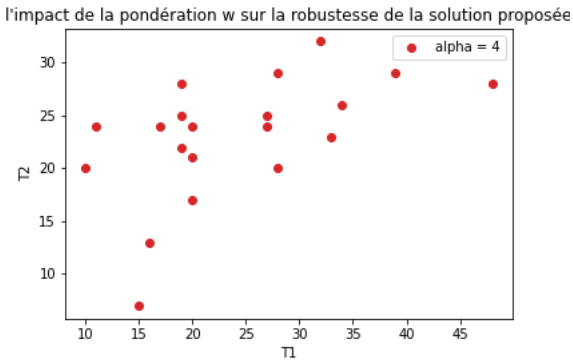
4.3 Tests et résultats

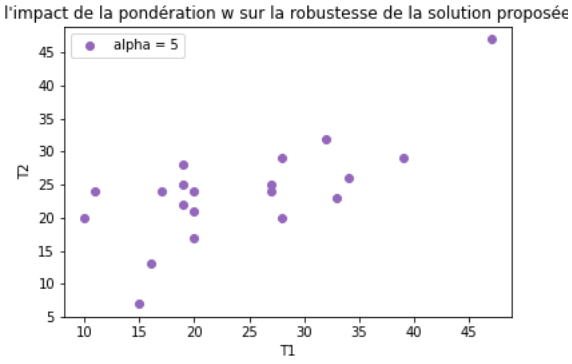
Nous étudions maintenant l'impact de la pondération w sur la robustesse de la solution proposée sur 20 instances. Nous présentons graphiquement les résultats obtenus:











Discussion des résultats obtenus

- On constate que les nuages de points sont largement identiques pour $alpha \in [2, 3, 4]$
- Pour certaines instances la solution devient plus équitable en augmentant alpha. En effet les points se rapprochent du centre pour alpha=5.