Projet MOGPL : optimisation robuste dans l'incertain total

Jeanne Bonnaventure

Novembre 2024

Table de matière

1 Introduction et notations **3**

2 Linéarisation des critères maxmin et minmax regret **3**

2.1 Critère maxmin . . . 3

2.1.1 Forme générale . . . 3

2.1.2 Application à l'Exemple 1 . . . 4

2.2 Critère minmax regret . . . 4

2.2.1 Forme générale . . . 5

2.2.2 Application à l'Exemple 1 . . . 5

2.3 Visualisation des solutions . . . 6

2.4 Etude de l'évolution du temps en fonction de n et p . . . 6

3 Linéarisation des critères maxOWA et minOWA **7**

3.1 Critère maxOWA . . . 7

3.1.1 Tri implicite dans le calcul des $L_k(z)$ . . . 8

3.1.2 Dual . . . 8

3.1.3 Utilisation du dual pour trouver les composantes du vecteur L . . . 9

3.1.4 Réécriture de l'OWA . . . 9

3.1.5 Application à l'exemple 1 . . . 9

3.2 Critère minOWA des regrets . . . 10

3.2.1 Tri implicite dans le calcul de $R_k(r)$ . . . 10

3.2.2 Dual . . . 11

3.2.3 Réécriture de l'OWA . . . 11

3.2.4 Forme générale . . . 11

3.2.5 Application à l'exemple 1 . . . 12

3.3 Etude de l'évolution du temps de résolution en fonction de n et p . . . 12

4 Application à la recherche d'un chemin robuste dans un graphe **13**

4.1 Notation . . . 13

4.2 Programme linéaire pour calculer le chemin le plus rapide . . . 14

4.2.1 Application à l'exemple 2 . . . 14

4.3 Critères et chemin robuste . . . 14

4.4 Etude de l'évolution du temps de résolution en fonction du nombre de scénarios n et de noeuds p . . . 17

1 Introduction et notations

L'optimisation linéaire classique corresponds à résoudre des programmes linéaires. Cela corresponds à minimiser ou maximiser une fonction objectif f sur un domaine de solution X tel que pour tout $x \in X$, x vérifie des contraintes définies. La fonction objectif f est déterminée de manière certaine. En optimisation robuste on introduit un ensemble S qui corresponds à un ensemble de scénarios possibles, pour chaque scénarios les contraintes restent les mêmes mais la fonction objectif diffère. Chaque $x \in X$ est désormais caractérisé par un vecteur de conséquence $z(x) = (z_1(x), ..., z_n(x))$ qui corresponds aux évaluation de x dans tous les scénarios de S. Si on ne possède aucune information sur les probabilités des différents scénarios on se trouve dans l'incertain total. Dans ce cas on cherche à trouver une solution qui soit robuste c'est à dire une solution qui soit performante dans tous les scénarios possibles.

- X: Domaine combinatoire de solutions.
- f : Fonction objectif attribuant une valeur à toute solution réalisable.
- $S = \{s_1, s_2, \ldots, s_N\}$: Ensemble fini des scénarios possibles.
- $f(x, s)$: Evaluation de $x \in X$ dans le scénario $s \in S$.
- $z(x) = (z_1(x), ..., z_n(x))$: vecteur de conséquences possibles de $x \in X$ où $z_i(x) = f(x, s_i)$.

2 Linéarisation des critères maxmin et minmax regret

2.1 Critère maxmin

On cherche, avec ce critère, à minimiser la fonction

$$g(x) = \min_{i=1,\ldots,n} z_i(x) \tag{1}$$

, c'est à dire la solution dont l'évaluation dans le pire des cas est la meilleure possible. On maximise la valeur minimale de la fonction objectif.

2.1.1 Forme générale

Pour cela on introduit une variable $t = g(x)$ qui corresponds à l'évaluation minimale d'une solution parmis tous les scénarios possibles. On cherche à maximiser t sous les contraintes que c'est un minimum et que x est bien une solution réalisable. Dans un problème à p variables, q contraintes

et n scénarios, on obtient le programme linéaire suivant:

$$\begin{aligned} \text{Maximiser} \quad & t \\ \text{Sous les contraintes} \quad & t \leq \sum_{j=1}^{p} c_{ij} x_j, \quad \forall i \in \{1, \ldots, n\} \\ & \sum_{j=1}^{p} a_{kj} x_j \leq b_k, \quad \forall k \in \{1, \ldots, q\} \\ & x_i \in \mathbb{R}_+ \quad \text{ou} \quad x_i \in \mathbb{Z}_+ \quad \forall i \in \{1, \ldots, p\} \end{aligned} \tag{2}$$

Avec C la matrice des fonctions objectifs des différents scénarios, A la matrice des contraintes du problème d'origine, B le second membre du problème d'origine. La première contrainte permet d'assurer que t est un minimum, la seconde permet de s'assurer que x est réalisable.

2.1.2 Application à l'Exemple 1

$$\begin{aligned} \text{Maximiser} \quad & t \\ \text{Sous les contraintes} \quad & \bullet \ t \leq 70x_1 + 18x_2 + 16x_3 + 14x_4 + 12x_5 \\ & \quad + 10x_6 + 8x_7 + 6x_8 + 4x_9 + 2x_{10}, \\ & \bullet \ t \leq 2x_1 + 4x_2 + 6x_3 + 8x_4 + 10x_5 \\ & \quad + 12x_6 + 14x_7 + 16x_8 + 18x_9 + 70x_{10}, \\ & \bullet \ 60x_1 + 10x_2 + 15x_3 + 20x_4 + 25x_5 \\ & \quad + 20x_6 + 5x_7 + 15x_8 + 20x_9 + 60x_{10} \leq 100. \\ & \bullet \ x_i \in \{0, 1\} \quad \forall i \in \{1, \ldots, 10\} \end{aligned} \tag{3}$$

On obtient avec ce critère les résultats suivant:

- Solution optimale x^* : (0, 1, 1, 1, 0, 0, 1, 1, 1, 0)
- Vecteur image $z(x^*)$: (66, 66)
- Valeur de la fonction objectif à l'optimum $g(x^*)$: 66

2.2 Critère minmax regret

Pour ce critère on introduit la notion de regret d'avoir choisi une solution $x \in X$, ce regret est défini par $r(x, s_i) = z_i^* - z_i(x)$ avec $z_i^* = \max_{y \in X} z_i(y)$. On va chercher à minimiser le regret d'avoir choisi x dans le scénario s_i c'est à dire minimiser l'amplitude du manque à gagner par rapport à la meilleure décision dans ce scénario. Cela corresponds à minimiser:

$$g(x) = \max_{i=1,\ldots,n} r(x, s_i) \tag{4}$$

2.2.1 Forme générale

On introduit la variable $t = g(x)$, et on précalcul les constantes z_i^* en résolvant le programme linéaire dans le scénario i. On cherche ensuite à minimiser t sous les contraintes que c'est un maximum et que x est une solution réalisable. Dans un problème à p variables, q contraintes et n scénarios on obtient le programme linéaire suivant:

$$\begin{aligned} \text{Minimiser} \quad & t \\ \text{Sous les contraintes} \quad & t \geq z_i^* - z_i(x), \quad \forall i \in \{1, \ldots, n\} \\ & \sum_{j=1}^{p} a_{kj} x_j \leq b_k, \quad \forall k \in \{1, \ldots, q\} \\ & x_i \in \mathbb{R}_+ \quad \text{ou} \quad x_i \in \mathbb{Z}_+ \quad \forall i \in \{1, \ldots, p\} \end{aligned} \tag{5}$$

2.2.2 Application à l'Exemple 1

En résolvant les programmes linéaires associés aux deux scénarios de l'exemple 1 on trouve $z_1^* = 118$ et $z_2^* = 112$.

$$\begin{aligned} \text{Minimiser} \quad & t \\ \text{Sous les contraintes} \quad & \bullet \; t \geq z_1^* - (70x_1 + 18x_2 + 16x_3 + 14x_4 + 12x_5 \\ & \quad + 10x_6 + 8x_7 + 6x_8 + 4x_9 + 2x_{10}), \\ & \bullet \; t \geq z_2^* - (2x_1 + 4x_2 + 6x_3 + 8x_4 + 10x_5 \\ & \quad + 12x_6 + 14x_7 + 16x_8 + 18x_9 + 70x_{10}), \\ & \bullet \; 60x_1 + 10x_2 + 15x_3 + 20x_4 + 25x_5 \\ & \quad + 20x_6 + 5x_7 + 15x_8 + 20x_9 + 60x_{10} \leq 100. \\ & \bullet \; x_i \in \{0, 1\} \quad \forall i \in \{1, \ldots, 10\} \end{aligned} \tag{6}$$

En résolvant ce programme linéaire, obtient les résultats suivants:

- Solution optimale : $x'^* = (0, 1, 1, 0, 0, 1, 1, 1, 1, 0)$
- Vecteur image $z(x'^*) = (62, 70)$
- Valeur de la fonction objectif à l'optimum $g(x'^*) : 50.0$

2.3 Visualisation des solutions

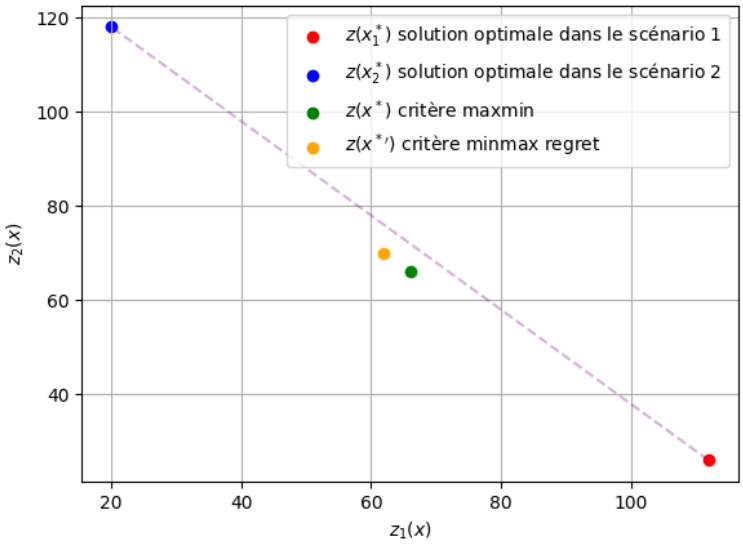

Figure 1: Representation des solutions dans le plan $(z_1(x), z_2(x))$

On remarque sur la figure 1 que les solutions optimales au sens du critère maxmin et minmax regret ne se trouve pas sur la corde qui relie $z(x_1^*)$ et $z(x_2^*)$. Or, si l'on considère une moyenne pondérée de ces deux solutions, cette combinaison doit se trouver sur la corde qui les relie. On en conclut que ni x^*, ni x'^* n'a pu être trouvé en maximisant une moyenne pondérée de $z(x_1^*)$ et $z(x_2^*)$ sinon ils se trouverait sur la droite.

2.4 Etude de l'évolution du temps en fonction de n et p

Il est possible de généraliser l'exemple 1 en considérant un problème de sac-à-dos robuste en présence de p projets et n scénarios. Un problème de la sorte est définit en tirant aléatoirement dans l'intervalle $[1, ;100]$ les coûts des p projet ainsi que leur utilité dans dans chacun des n scénarios. De plus, on fixe la contrainte de poids à 50% du coût total des projets considérés.

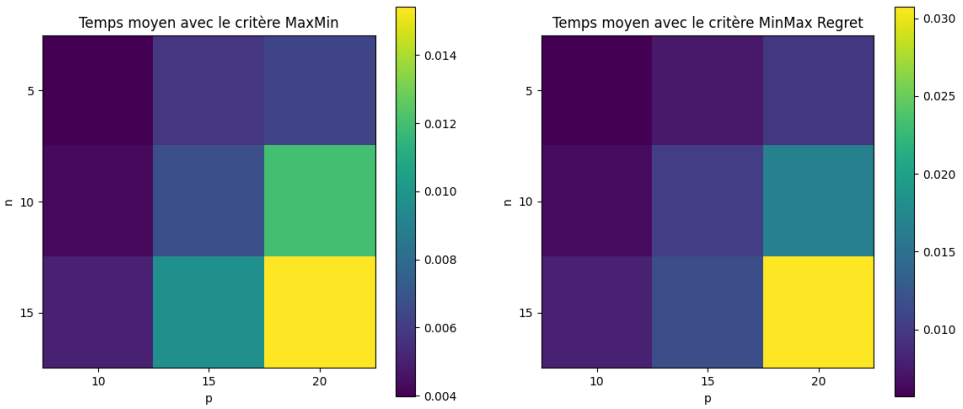

Figure 2: Etude de l'évolution du temps de résolution en fonction de n et p pour le critère maxmin et minmax regret

La figure 2 a été obtenu en tirant aléatoirement 10 instances de problème pour chaque couple (n, p), pour ensuite calculer une moyenne de temps de résolution pour chacun des critères. On remarque que l'évolution du temps de résolution moyen pour chacun des critères en fonction des couples (n, p) est similaire. Cependant on note que le critère maxmin a un temps de résolution moyen moins élévé que le critère minmax regret. Cela s'explique par le fait que le critère minmax regret possède une étape préliminaire de calcul des $z_i^* \quad \forall i \in [1, \dots, n]$, ce qui augmente le temps de résolution moyen.

3 Linéarisation des critères maxOWA et minOWA

3.1 Critère maxOWA

On cherche avec ce critère à maximiser la fonction

$$g(x) = \sum_{i=1}^{n} w_i z_{(i)}(x) \tag{7}$$

où les w_i sont des poids positifs et décroissants lorsque i augmente ($w_i \geq w_{i+}, i = 1, \dots, n-1$), ils sont fixés par l'utilisateur. Le vecteur $(z_{(1)}(x), \dots, z_{(n)}(x))$ représente le résultat d'un tri des composantes de $(z_1(x), \dots, z_n(x))$ par ordre croissant (ainsi $z_{(i)}(x) \leq z_{(i+1)}(x), i = 1, \dots, n-1$). On applique un poids plus important aux composantes faibles du vecteur des $z_i(x)$. On applique donc une "moyenne pondérée ordonnée".

3.1.1 Tri implicite dans le calcul des $L_k(z)$

Soit $z \in \mathbb{R}$, et le vecteur $L(z) = (L_1(z), \ldots, L_n(z))$ avec $L_k(z) = \sum_{i=1}^{n} a_{ik} z_{(i)}$ c'est à dire la somme des k plus petits éléments de z. On observe que $L_k(z)$ est la valeur optimale du programme linéaire en variables 0-1 suivant:

$$\begin{aligned} \text{Minimiser} \quad & \sum_{i=1}^{n} a_{ik} z_i \\ \text{Sous les contraintes} \quad & \sum_{i=1}^{n} a_{ik} = k \\ & a_{ik} \in \{0,1\}, i = 1, \ldots, n \end{aligned} \tag{8}$$

Les a_{ik} appartenant à $\{0,1\}$, agissent comme des sélecteurs si $a_{ik} = 1$, cela signifie que la i-ème composante de z est prise en compte dans la fonction objectif. La contrainte de ce programme linéaire garantit qu'exactement k composantes soient sélectionnées. On cherche donc à minimiser la somme de k composantes de z intuitivement on souhaite choisir les k plus petits éléments de z. Par définition $z_{(1)}(x), \ldots, z_{(k)}(x)$ correspondent aux k plus petits éléments, et $L_k(z)$ corresponds à leur somme.

Ainsi à l'optimum les k a_{ik} qui sont à 1 sélectionnent les k plus petits éléments de z et la valeur optimale du programme linéaire corresponds à $L_k(z)$. En choisissant d'autres a_{ik}, on choisirait d'autres composantes que les k plus petites et la somme serait supérieure car les z_i sélectionnés serait plus grand que certains des k plus petits.

3.1.2 Dual

Soit le programme linéaire suivant:

$$\begin{aligned} \text{Minimiser} \quad & \sum_{i=1}^{n} a_{ik} z_i \\ \text{Sous les contraintes} \quad & \begin{cases} \sum_{i=1}^{n} a_{ik} = k \\ a_{ik} \leq 1 \end{cases} \\ & a_{ik} \in \{0,1\}, i = 1, \ldots, n \end{aligned} \tag{9}$$

Le dual associé au programme linéaire ci-dessus est le suivant:

$$\begin{aligned} \text{Maximiser} \quad & k r_k - \sum_{i=1}^{n} b_{ik} \\ \text{Sous les contraintes} \quad & r_k - b_{ik} \leq z_i \quad \forall i \in [1, \ldots, n] \\ & r_k \in \mathbb{R} \quad \text{et} \quad b_{ik} \geq 0 \quad \forall i \in [1, \ldots, n] \end{aligned} \tag{10}$$

3.1.3 Utilisation du dual pour trouver les composantes du vecteur L

Soit le vecteur $z = [2, 9, 6, 8, 5, 4]$, on chercher à trouver les composantes de $L(z)$, en passant par le dual. On sait qu'à l'optimum les fonctions objectifs du dual (10) et du primal (9) ont la même valeur. De plus on a vu que $L_k(z)$ est la valeur optimale du programme linéaire primal. Ainsi en résolvant le dual on obtient les composantes suivantes: $L(z) = (2, 6, 11, 17, 25, 34)$.

3.1.4 Réécriture de l'OWA

L'OWA défini par l'équation (7) peut se réécrire $g(x) = \sum_{k=1}^{n} w'_k L_k(z(x))$ avec $w'_k = w_k - w_{k+1}$ pour $k = 1, \ldots, n-1$ et $w'_n = w_n$.
Démonstration:

Par définition $L_k(z(x)) = \sum_{i=1}^{k} z_{(i)}(x)$. On peut donc réécrire

$$z_{(i)}(x) = L_i(z(x)) - L_{i-1}(z(x)) \quad \text{pour} \quad i = 1, \ldots, n \tag{11}$$

avec la convention $L_0(z(x)) = 0$. On réécrit (7) avec (11) afin d'obtenir:

$$\begin{aligned} g(x) &= \sum_{i=1}^{n} w_i (L_i(z(x)) - L_{i-1}(z(x))) \\ &= \sum_{i=1}^{n} w_i L_i(z(x)) - \sum_{i=1}^{n} w_i L_{i-1}(z(x)) \\ &= \sum_{i=1}^{n} w_i L_i(z(x)) - \sum_{j=0}^{n-1} w_{j+1} L_j(z(x)) \\ &= \sum_{i=1}^{n-1} (w_i - w_{i+1}) L_i(z(x)) + w_n L_n(z(x)) \end{aligned}$$

En intégrant la notation des w'_k on obtient:

$$\begin{aligned} g(x) &= \sum_{i=1}^{n-1} w'_i L_i(z(x)) + w'_n L_n(z(x)) \\ &= \sum_{i=1}^{n} w'_i L_i(z(x)) \end{aligned} \tag{12}$$

3.1.5 Application à l'exemple 1

En utilisant la linéarisation de l'OWA donnée en 2.4 de l'énoncé on peut formuler le problème de l'optimisation d'un OWA dans l'exemple 1 comme le programme linéaire suivant:

$$
\begin{array}{rl}
\text{Maximiser} & r_1 - b_{11} - b_{12} + r_2 - b_{21} - b_{22} \\
\text{Sous les contraintes} & 60x_1 + 10x_2 + 15x_3 + 20x_4 + 25x_5 \\
& \quad + 20x_6 + 5x_7 + 15x_8 + 20x_9 + 60x_{10} \quad \leq \quad 100. \\
& r_1 - b_{11} \quad \leq z_1(x) \\
& r_1 - b_{12} \quad \leq z_2(x) \\
& r_2 - b_{21} \quad \leq z_1(x) \\
& r_2 - b_{22} \quad \leq z_2(x) \\
& b_{ij} \geq 0, \quad r_i \in \mathbb{R} \quad \forall i, j = 1, 2, \quad x_k \in \{0, 1\}
\end{array}
\tag{13}
$$

Dans ce programme linéaire on a fixé $w_1 = 2$ et $w_2 = 1$ afin d'obtenir $w'_1 = 1$ et $w'_2 = 1$. En résolvant ce programme linéaire, obtient les résultats suivants:

- Solution optimale : $x^* = (0, 1, 1, 1, 0, 0, 1, 1, 1, 0)$
- Vecteur image $z(x^*) = (66{,}66)$
- Valeur de la fonction objectif à l'optimum $g(x^*)$: 198.0

3.2 Critère minOWA des regrets

Avec ce critère on cherche à minimiser la fonction :

$$
g(x) = \sum_{i=1}^{n} w_i r(x, s_{(i)}) \tag{14}
$$

où les w_i sont des poids décroissants lorsque i augmente, fixés par l'utilisateur. Et $(r(x, s_{(1)}), r(x, s_{(2)}), \ldots, r(x, s_{(n)}))$ représente le résultat d'un tri des composantes de $r = (r(x, s_1), r(x, s_2), \ldots, r(x, s_n))$ par ordre décroissant. On rappelle que $r(x, s_i) = z_i^* - z_i(x)$. Afin de linéariser ce critère nous allons nous inspirer de l'approche utilisée pour linéariser le critère maxOWA.

3.2.1 Tri implicite dans le calcul de $R_k(r)$

Pour tout vecteur $r \in \mathbb{R}$ on note $R(r)$ le vecteur $(R_1(r), R_2(r), \ldots, R_n(r))$ dont la composante $R_k(r)$ est définie par $R_k(r) = \sum_{i=1}^{k} r_{(i)}$. C'est à dire que $R_k(r)$ corresponds à la somme des k plus importantes composantes de r. On remarque que $R_k(r)$ est la valeur optimale du programme linéaire en variables 0-1 suivant:

$$
\begin{array}{rl}
\text{Maximiser} & \displaystyle\sum_{i=1}^{n} a_{ik} r_i \\
\text{Sous les contraintes} & \displaystyle\sum_{i=1}^{n} a_{ik} = k \\
& a_{ik} \in \{0, 1\}, i = 1, \ldots, n
\end{array}
\tag{15}
$$

En effet si l'on cherche à maximiser une somme de k élément de r on choisit les k plus importantes. On admet ensuite que ce programme peut être relaxé en variables continues ce qui fait que $R_k(r)$ est aussi la valeur à l'optimum du programme linéaire suivant:

$$\begin{aligned} &\text{Maximiser} && \sum_{i=1}^{n} a_{ik} r_i \\ &\text{Sous les contraintes} && \begin{cases} \sum_{i=1}^{n} a_{ik} = k \\ a_{ik} \leq 1 \end{cases} \\ & && a_{ik} \in \{0,1\}, i = 1, \ldots, n \end{aligned} \tag{16}$$

3.2.2 Dual

En passant au dual on obtient le programme linéaire suivant:

$$\begin{aligned} &\text{Minimiser} && k d_k + \sum_{i=1}^{n} b_{ik} \\ &\text{Sous les contraintes} && d_k - b_{ik} \geq r_i \quad \forall i \in [1, \ldots, n] \\ & && d_k \in \mathbb{R} \quad \text{et} \quad b_{ik} \geq 0 \quad \forall i \in [1, \ldots, n] \end{aligned} \tag{17}$$

on note d_k la variable duale associé à la première contrainte de (16) et b_{ik} les variables duales associées aux contraintes $a_{ik} \leq 1 \quad i = 1, \ldots, n$.

3.2.3 Réécriture de l'OWA

Par définition nous avons $R_k(r(x)) = \sum_{i=1}^{k} r_{(i)}(x)$ donc :

$$\begin{aligned} & r(x, s_{(i)}) = R_i(r(x)) - R_{i-1}(r(x)) \quad \forall i = 1, \ldots, n \\ & \text{avec} \quad R_0(r(x)) = 0 \end{aligned} \tag{18}$$

En suivant le même raisonnement que dans la partie 3.1.4 on obtient la réécriture : $g(x) = \sum_{k=1}^{n} w'_k R_k(r(x))$ avec $w'_k = w_k - w_{k+1} \quad \forall k = 1, \ldots, n-1$ et $w'_n = w_n$.

3.2.4 Forme générale

En utilisant ce qui a été vu précédemment, la minimisation d'un OWA sur un ensemble X décrit par des contraintes linéaires peut s'écrire sous la forme du programme linéaire suivant:

$$
\begin{aligned}
\text{Minimiser} \quad & \sum_{k=1}^{n} w'_k (k d_k + \sum_{i=1}^{n} b_{ki}) \\
\text{Sous les contraintes} \quad & d_k - b_{ki} \geq z_i^* - z_i(x) \quad \forall i \in [1, \ldots, n] \\
& x \in X \\
& d_k \in \mathbb{R} \quad \text{et} \quad b_{ik} \geq 0 \quad \forall i, k \in [1, \ldots, n]
\end{aligned}
\tag{19}
$$

3.2.5 Application à l'exemple 1

En utilisant la linéarisation de l'OWA donnée en 2.4 de l'énoncé on peut formuler le problème de l'optimisation d'un OWA dans l'exemple 1 comme le programme linéaire suivant:

$$
\begin{aligned}
\text{Minimiser} \quad & r_1 + b_{11} + b_{12} + 2r_2 + b_{21} + b_{22} \\
\text{Sous les contraintes} \quad & 60x_1 + 10x_2 + 15x_3 + 20x_4 + 25x_5 \\
& \quad + 20x_6 + 5x_7 + 15x_8 + 20x_9 + 60x_{10} \quad \leq \quad 100. \\
& r_1 - b_{11} \quad \geq z_1^* - z_1(x) \\
& r_1 - b_{12} \quad \geq z_2^* - z_2(x) \\
& r_2 - b_{21} \quad \geq z_1^* - z_1(x) \\
& r_2 - b_{22} \quad \geq z_2^* - z_2(x) \\
& b_{ij} \geq 0, \quad r_i \in \mathbb{R} \quad \forall i, j = 1, 2, \quad x_k \in \{0, 1\}
\end{aligned}
\tag{20}
$$

Dans ce programme linéaire on a fixé $w_1 = 2$ et $w_2 = 1$ afin d'obtenir $w'_1 = 1$ et $w'_2 = 1$. En résolvant ce programme linéaire, obtient les résultats suivants:

- Solution optimale : $x^* = (0, 1, 0, 1, 1, 0, 1, 1, 1, 0)$
- Vecteur image $z(x^*) = (62, 70)$
- Valeur de la fonction objectif à l'optimum $g(x^*)$: 150.0

3.3 Etude de l'évolution du temps de résolution en fonction de n et p

Dans cette section nous avons générer 10 instances de problèmes de sac à dos pour chacun des couples (n, p) avec $n = [15, 10, 15]$ et $p = [10, 15, 20]$.

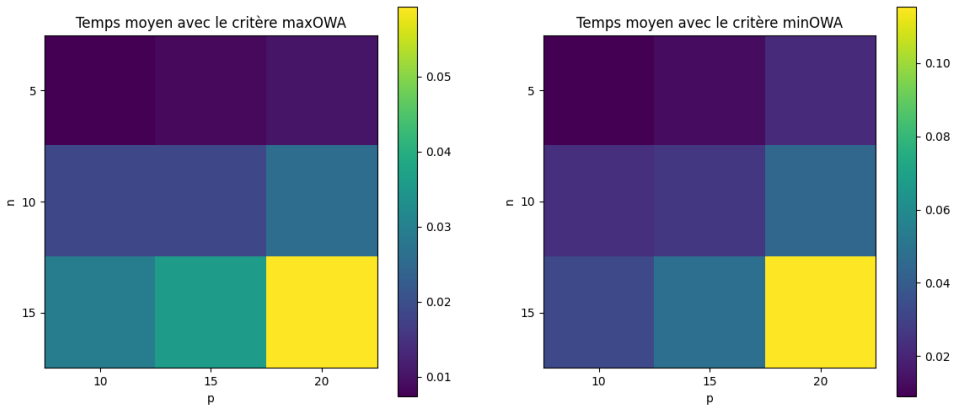

Figure 3: Etude de l'évolution du temps de résolution en fonction de n et $p$pour le critère maxOWA et minOWA regret

On observe que le temps de résolution augmente lorsque p augmente et lorsque n augmente pour les deux critères. De plus les temps de résolution sont similaires pour les problèmes avec une taille élévée mais pour de plus petits problèmes le critrère maxOWA semble être plus rapide.

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

4.1 Notation

- $G(V, A)$: le graphe orienté G tel que que V est l'ensemble des sommets du graphe et $A \subseteq V \times V$ l'ensemble des aretes.
- $S = 1, \ldots, n$: ensemble des scénarios
- t_{ij}^s: représente le temps nécessaire pour parcourir l'arc (i, j) dans le scénario s
- $t^s(P) = \sum_{(i,j)\in P} t_{ij}^s$: temps pour parcourir un chemin P, les temps sont additifs le long d'un chemin
- $t(P) = (t^1(P), \ldots, t^n(P))$: vecteur de conséquences possibles du chemin P
- d: le sommet de départ
- a: le sommet d'arrivée

4.2 Programme linéaire pour calculer le chemin le plus rapide

Etant donné un graphe orienté G, le programme linéaire qui permet de calculer le chemin le plus rapide du sommet initial au sommet destination dans un scénario $s \in S$ donné est le suivant:

$$
\begin{aligned}
\text{Minimiser} \quad & \sum_{(i,j)\in A} t_{ij}^{s} x_{ij} \\
\text{Sous les contraintes} \quad & \sum_{j\in V:(i,j)\in A} x_{ij} - \sum_{j\in V:(j,i)\in A} x_{ji} = 0 \quad \forall i \in V \setminus \{d,a\} \\
& \sum_{j\in V:(d,j)\in A} x_{dj} = 1 \\
& \sum_{j\in V:(j,a)\in A} x_{ja} = 1 \\
& x_{ij} \in \{0,1\} \quad \forall (i,j) \in A
\end{aligned}
\tag{21}
$$

4.2.1 Application à l'exemple 2

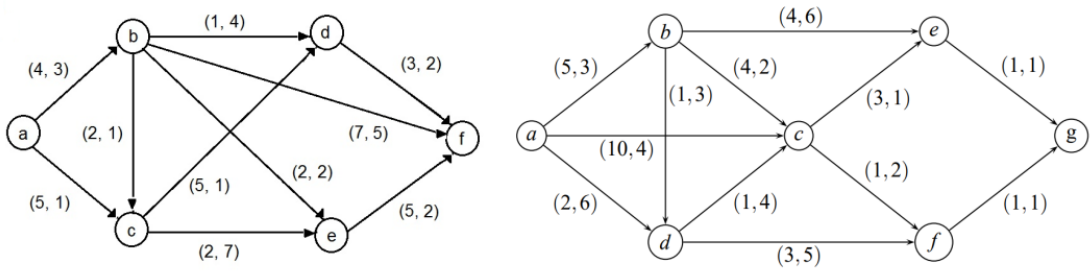

Figure 4: Deux instances du problème de plus court chemin robuste à 2 scénarios

En résolvant le problème du plus court chemin avec le programme linéaire (21) appliqué aux graphes de la figure 4 on obtient les résultats suivants:

- Graphe de gauche:
  - Scénario 1: Chemin optimal: $((a,b),(b,d),(d,f))$, avec un coût de 8
  - Scénario 2: Chemin optimal: $((a,c),(c,d),(d,f))$, avec un coût de 4
- Graphe de droite:
  - Scénario 1: Chemin optimal: $((a,d),(d,c),(c,f),(f,g))$, avec un coût de 5
  - Scénario 2: Chemin optimal: $((a,c),(c,e),(e,g))$, avec un coût de 6

4.3 Critères et chemin robuste

Dans cette partie on formalise un programme linéaire pour calculer un chemin robuste pour chacun des critères vu précédemment. Puis on l'applique à l'exemple 2 4.

- Critère MaxMin:

$$\begin{aligned}
\text{Maximiser} \quad & y \\
\text{Sous les contraintes} \quad & \sum_{j \in V:(i,j) \in A} x_{ij} - \sum_{j \in V:(j,i) \in A} x_{ji} = 0 \quad \forall i \in V \setminus \{d, a\} \\
& \sum_{j \in V:(d,j) \in A} x_{dj} = 1 \\
& \sum_{j \in V:(j,a) \in A} x_{ja} = 1 \\
& y \leq - \sum_{(i,j) \in A} t_{ij}^s x_{ij} \quad \forall s \in S \\
& x_{ij} \in \{0, 1\} \quad \forall (i,j) \in A, y \leq 0
\end{aligned} \tag{22}$$

Les trois premières contraintes permettent d'assurer qu'un chemin considéré P est réalisable, la quatrième sert quant à elle à assurer que $y = \min_{i=1,\ldots,n} -t^i(P)$

  - Résultat pour le graphe de gauche de 4:
    Chemin optimal : $P = ((a,b),(b,d),(d,f))$
    $t(P) = (8, 9)$
    Valeur de la fonction objectif à l'optimum: -9.0
  - Résultat pour le graphe de droite de 4:
    Chemin optimal : $P = ((a,b),(b,e),(e,g))$
    $t(P) = (10, 10)$
    Valeur de la fonction objectif à l'optimum: -10.0

- Critère MinMax regret:

$$\begin{aligned}
\text{Minimiser} \quad & y \\
\text{Sous les contraintes} \quad & \sum_{j \in V:(i,j) \in A} x_{ij} - \sum_{j \in V:(j,i) \in A} x_{ji} = 0 \quad \forall i \in V \setminus \{d, a\} \\
& \sum_{j \in V:(d,j) \in A} x_{dj} = 1 \\
& \sum_{j \in V:(j,a) \in A} x_{ja} = 1 \\
& y \geq -(c_s^* - \sum_{(i,j) \in A} t_{ij}^s x_{ij}) \quad \forall s \in S \\
& x_{ij} \in \{0, 1\} \quad \forall (i,j) \in A, y \in \mathbb{R}
\end{aligned} \tag{23}$$

Avec c_s^* le coût du chemin le plus court dans le scénario s. Les trois premières contraintes permettent d'assurer qu'un chemin considéré P est réalisable, la quatrième sert quant à elle à assurer que $y = \max_{i=1,\ldots,n} -(c_s^* - t^i(P))$

– Résultat pour le graphe de gauche de 4:
Chemin optimal : $P = ((a,b),(b,e),(e,f))$
$t(P) = (11,7)$
Valeur de la fonction objectif à l'optimum: 3.0

– Résultat pour le graphe de droite de 4:
Chemin optimal : $P = ((a,b),(b,e),(e,g))$
$t(P) = (10,10)$
Valeur de la fonction objectif à l'optimum: 5.0

- Critère MaxOWA:

$$\begin{aligned}
\text{Maximiser} \quad & w'_k(kr_k - \sum_{i\in S} b_{ik}) \\
\text{Sous les contraintes} \quad & \sum_{j\in V:(i,j)\in A} x_{ij} - \sum_{j\in V:(j,i)\in A} x_{ji} = 0 \quad \forall i \in V \setminus \{d,a\} \\
& \sum_{j\in V:(d,j)\in A} x_{dj} = 1 \\
& \sum_{j\in V:(j,a)\in A} x_{ja} = 1 \\
& r_k - b_{ik} \leq - \sum_{(i,j)\in A} t^i_{(i,j)} x_{i,j} \\
& r_k \in \mathbb{R}, b_{ik} \geq 0 \quad \forall k, i \in S, \quad x_{ij} \in \{0,1\} \quad \forall (i,j) \in S
\end{aligned} \tag{24}$$

– Résultat pour le graphe de gauche de 4:
Pour tout k dans [2,4,8,16] on obtient les résultats suivants:
Chemin optimal : $P = ((a,b),(b,d),(d,f))$
$t(P) = [8,9]$

– Résultat pour le graphe de droite de 4:

k	P	t(P)	moyenne de $t(P)$	$\min t(P)$
1	$a \to d \to f \to g$	[6, 12]	9	6
2	$a \to b \to c \to f \to g$	[11,8]	9.5	8
4	$a \to b \to e \to g$	[10, 10]	10	10
8	$a \to b \to e \to g$	[10, 10]	10	10
16	$a \to b \to e \to g$	[10, 10]	10	10

On observe que pour lorsque k est petit le modèle va chercher un chemin qui minimise la moyenne des coûts. Lorsque k augmente le modèle devient plus robuste, il choisit des chemins qui minimisent les pires scénarios. Les chemins sont plus coûteux mais plus fiables.

- Critère MinOWA:

$$
\begin{aligned}
\text{Minimiser} \quad & w'_k(kd_k - \sum_{i \in S} b_{ik}) \\
\text{Sous les contraintes} \quad & \sum_{j \in V:(i,j) \in A} x_{ij} - \sum_{j \in V:(j,i) \in A} x_{ji} = 0 \quad \forall i \in V \setminus \{d, a\} \\
& \sum_{j \in V:(d,j) \in A} x_{dj} = 1 \\
& \sum_{j \in V:(j,a) \in A} x_{ja} = 1 \\
& d_k - b_{ik} \geq -(t^{i*} - \sum_{(i,j) \in A} t^i_{(i,j)} x_{i,j}) \\
& d_k \in \mathbb{R}, b_{ik} \geq 0 \quad \forall k, i \in S, \quad x_{ij} \in \{0,1\} \quad \forall (i,j) \in S
\end{aligned}
\tag{25}
$$

Avec t^{i*} le coût du plus court chemin dans le scénario i.

- Résultat pour le graphe de gauche de 4:
  Pour tout k dans [2,4,8,16] on obtient les résultats suivants:
  Chemin optimal : $P = ((a,b),(b,e),(e,f))$
  $t(P) = [11, 7]$

- Résultat pour le graphe de droite de 4:
  Pour tout k dans [2,4,8,16] on obtient les résultats suivants:
  Chemin optimal : $P = ((a,b),(b,e),(e,g))$
  $t(P) = [10, 10]$

4.4 Etude de l'évolution du temps de résolution en fonction du nombre de scénarios n et de noeuds p

Dans cette section nous avons générer 10 instances de graphes pour chacun des couples (n, p) avec $n = [2, 5, 10]$ et $p = [10, 15, 20]$, avec n le nombre de scénarios du graphe et p le nombre de noeuds dans le graphe. Ces graphes ont une densité d'arcs comprise entre 30% et 50% et les coûts des arcs dans les différents scénarios sont aléatoirement tirés dans l'intervalle [1,100].

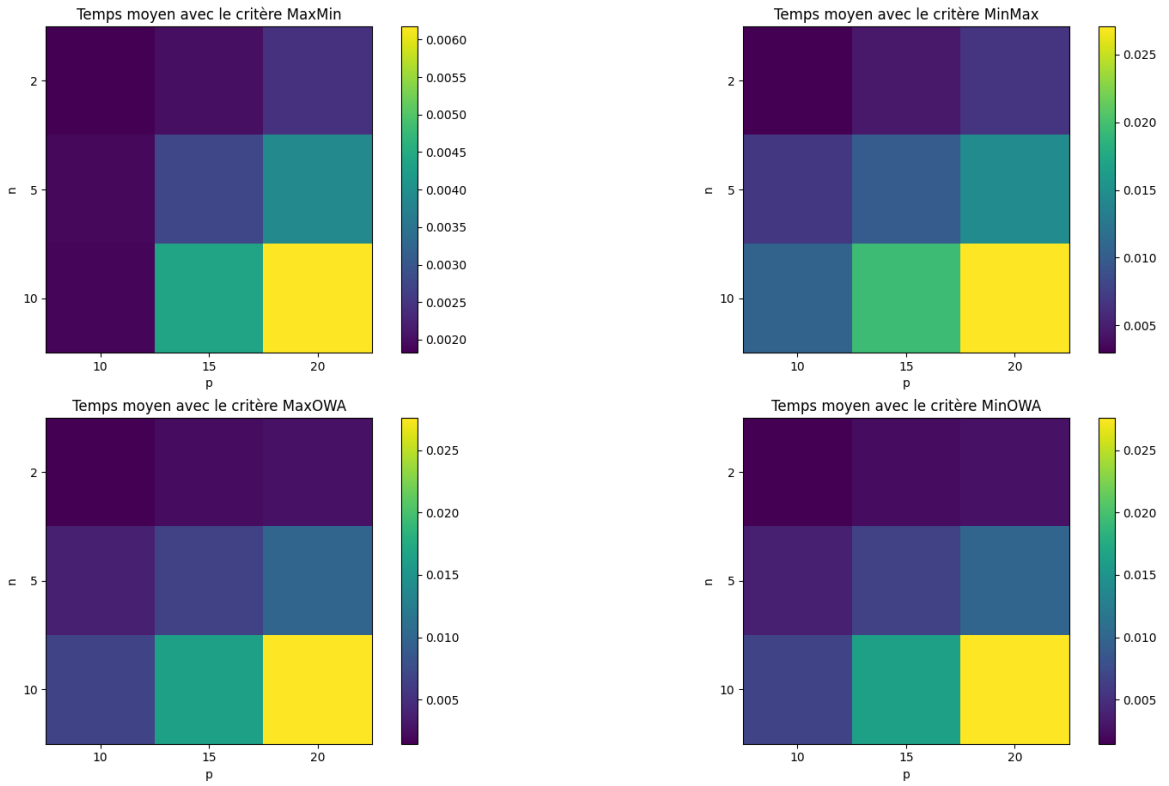

Figure 5: Étude de l'évolution du temps de résolution en fonction de n et p pour les critères Maxmin, Minmax, MaxOWA et MinOWA.

On observe que le temps de résolution augmente lorsque p augmente et lorsque n augmente pour tous les critères, car la complexité du problème augmente donc le problème nécessite plus de temps/calcul pour trouver un chemin robuste. Il apparaît aussi que les deux critères OWA prennent en moyenne plus de temps que les critères plus simples tels que MaxMin ou MinMax. Si dans un certain contexte l'objectif est d'avoir une solution le plus rapidement possible on choisira le critère MaxMin, mais pour des situations où la robustesse est prioritaire on choisira les critères OWA plus longs, mais plus robustes.