Université Sorbonne

Master Androide 2nd Year

MADMC

Rapport du projet MADMC
Sélection bi-objectifs avec coefficients intervalles

Auteurs :
Yuhan WANG

Encadrant :
Olivier spanjaard

Janvier 2021

1 Introduction

Ce projet s'inscrit dans l'étude des méthodes exactes pour le problème de sélection bi-objectif en cas de minimisation, qui est décrit de la façon suivante : parmi un ensemble de n objets valués par deux objectifs c_1^i, c_2^i, on voudrait déterminer parmi toutes les solutions possibles, des solutions minimax qui sont des sous-ensembles de k objets. On va développer deux procédures de résolution possibles, démonstrer leurs utilités et comparer leurs efficacités.

Les codes sont disponible sur : `https://github.com/yuhanWG/MADMC`.

2 Résultats préliminaires

2.1 Question1

On va montrer que le principe d'optimalité n'est pas vérifé en donnant un contre-exemple, c'est à dire dans le calcul du point minimax qu'on peut avoir simultanément :

$$f_I(y) < f_I(y') \text{ ET } f_I(y + y'') > f_I(y' + y'')$$

Supposons qu'on a 3 objets, dont les valuations bi-objectifs dans l'espace des objectifs et les autre données nécessaires sont les suivantes :

$$\begin{cases} \alpha_{min} = 0, \alpha_{max} = 0.9 \\ y1 = (50, 50) \\ y2 = (40, 60) \\ y3 = (20, 0) \end{cases}$$

Donc on a

$$f_I(y_1) = 50, f_I(y_2) = 60, f_I(y_1) < f_I(y_2)$$

Par contre :

$$f_I(y_1 + y_3) = f_I((70, 50)) = 70 * 0.9 + 50 * 0.1 = 68$$
$$f_I(y_2 + y_3) = f_I((60, 60) = 60$$
$$f_I(y_1 + y_3) > f_I(y_2 + y_3)$$

Conclusion : le principe d'optimalité n'est pas vérifié, on ne peut pas donc déterminer par récurrence le point minimax.

2.2 Algorithme1 naïf : comparaison exhaustive

Cette partie répond à la question3 dans l'énoncé.

L'idée de cet algorithme est de comparer les solutions deux à deux et trouver les solutions non-dominées au sens de Pareto. Cette recherche est effectuée par une boucle imbriqué qui comparer une solution avec toutes les autres solutions possibles. Si une solution n'est dominée par aucune autre solution, on l'ajoute dans l'ensemble non dominée. Le problème de la recherche exhaustive est qu'il faut parcourir n fois la liste pour identifier les vecteurs non dominés et sa complexité augmente exponentiellement avec n.

2.3 Algorithme2 : le tri lexicographique

Cette partie répond à la question4 dans l'énoncé.

L'idée de cet algorithme est d'abord faire un tri lexicographique des vecteurs puis effectuer un seul parcours pour identifier les vecteurs non dominés. Notons les vecteurs adjqcents v_i, v_{i+1} dont les images respective sont $(c_1^i, c_2^i), (c_1^{i+1}, c_2^{i+1})$. La relation de dominance entre les deux après le tri lexicographique peut être décrite de manière suivante :

$$\begin{cases} c_1^i = c_1^{i+1}, & \text{alors } c_2^i <= c_2^{i+1}, \ v_{i+1} \text{ forcément dominé} \\ c_1^i < c_1^{i+1}, & v_{i+1} \text{ n'est pas forcémet dominé} \end{cases}$$

Selon les caractéristiques décrits au-dessus, v_{i+1} soit dominé par v_i, soit ils sont incomparables. Donc l'algorithme de tri lex se déroule de la façon suivante :

On va parcourir une et une seule fois la liste des vecteurs, le premier vecteur après le tri est sûrement non-dominé car aucune solution n'a une meilleure performance que lui au premier critère. Après chaque solution est comparée

avec la dernière solutions ajoutée dans l'ensemble des vecteurs non-dominé. Si leur premières valuations sont égaux, pas besoin d'aller et voir le deuxième critère. Sinon on fait la comparaison et on l'ajoute dans la liste s'il est pas dominé.

2.4 Complexité

Cette partie répond à la question5 dans l'énoncé.

Dans cette partie, on compare les complexités des deux algorithmes proposés et trace les courbes des temps d'exécutions respectifs. Depuis les figures on peut observer que avec l'augmentation du nombre des vecteurs à comparer, tous les deux prennent plus du temps à dérouler. Mais par rapport à l'algorithme lexicographique, cette augmentation est beaucoup plus importante pour l'algorithme naif. Avec 10000 vecteurs, l'algorithme naif prend environ 2.3 minutes mais l'algorithme lexicograpique ne prend que 0.025 secondes.

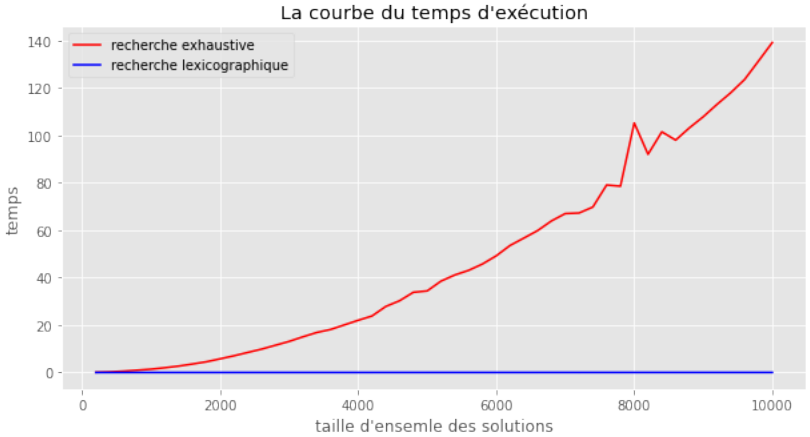

FIGURE 1 – Temps d'exécution des deux algorithmes en fonction de n

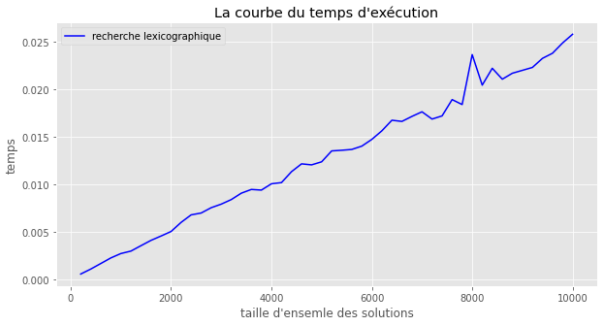

FIGURE 2 – Algorithme lexicographique

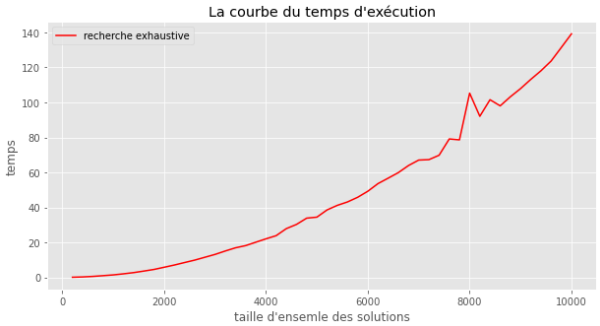

FIGURE 3 – Algorithme naif

3 Une première procédure de résolution : Minimax

3.1 Question6

Rappellons-nous la définition : une solution minimax est une solution qui minimise la formule suivante :

$$f_y(y) = max\{\alpha y_1 + (1 - \alpha y_2)\}$$

On va faire le raisonnement à l'absurde : supposons b est une solution minimax mais elle n'est pas incluse dans l'ensemble des solutions Pareto-optimaux. Selon la définition de l'ensemble Pareto-optimal, cela signifie qu'il existe une solution a qui appartient à l'ensemble Pareto-optimale qui la domine : $a \succ_p b$.

$$a \succ_p b \Rightarrow \begin{cases} a_1 = b_1, a_2 < b_2 \\ a_1 < b_1, a_2 \leq b_2 \end{cases}$$

Selon l'ordre interne des deux objectifs, on dispose 2 possiblités :

1. L'ordre interne entre des deux objectifs(c_1, c_2) est cohérent pour les deux solutions a et b. Cela signifie que c'est le même α qui maximise $f_I(a)$ et $f_I(b)$. Comme a domine b, donc il existe au moins une dimension où la performance de la solution a est meilleure que celle de b, ça veut dire sa valuation est plus petite. Dans le cas là, c'est évidemment d'obtenir $f_I(a) < f_I(b)$. Donc la solution b n'est pas un point minimax. Contradiction.
2. Si l'ordre deux deux objetcifs n'est pas le même, supposons-nous c'est α_{min} qui maximise le $f_I(b)$ et α_{max} qui maximise $f_I(a)$: On a :

$$\alpha_{min}b_1 + (1 - \alpha_{min})b_2 > \alpha_{max}b_1 + (1 - \alpha_{max})b_2$$

Selon la définition de la dominance au sens Pareto, on a :

$$\alpha_{max}b_1 + (1 - \alpha_{max})b_2 > \alpha_{max}a_1 + (1 - \alpha_{max})a_2$$

Selon la transitivité de l'inégalité :

$$\alpha_{min}b_1 + (1 - \alpha_{min})b_2 > \alpha_{max}a_1 + (1 - \alpha_{max})a_2$$

Donc la solution b n'est pas un point minimax. Contradiction.

On peut montrer de même façon le cas inverse où c'est α_{max} qui maximise le $f_I(b)$ et α_{mon} qui maximise $f_I(a)$.
Selon les démonstrations, on peut donc déduire que, tous les points minimax sont obligatoirement inclus dans l'ensemble des points non-dominées.

3.2 Programmation dynamique

Dans cette partie, on va utiliser la programmation dynamique pour résoudre le problème suivant :
Etant donné un ensemble de n vecteurs, retourner un ensemble des solutions Pareto-optimaux en sélectionnant k objets parmi les n. Comme ce qu'on a vu en cours, il faut un tableau T de taille $(k+1) * N$ pour enregistrer les (sous)solutions Pareto-optimaux.
Chaque case $T[i, j]$ enregistre les sous-solutions Pareto-optimal en tirant i objets depuis les $(j+1)$ premiers vecteurs(i.e le sous-ensemble $1, ..., j$ des objets). Donc ce qui diffère les colonnes, c'est l'ajout de l'objet $(j+1)$ dans l'ensemble des vecteurs qu'on peut choisir. Il existe deux stratégies : soit on néglige cette objet et prend l'ancienne solution dans la case $T[i, j-1]$, soit on prend cette objet et ajouter son coût dans l'ancienne solution optimale enregistrée dans la case $T[i-1, j-1]$. Pour déterminer les solutions optimales de cette case, il faut calculer les solutions obtenues via les deux stratégies et prendre les solutions Pareto-optimals dedans.

Taille ↓	{1}	...	{1,...,j-1}	{1,...,j}	...	{1,...,n}
0	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)
1						
...						
i-1			F			
i			G	E		
...						
k						

PARETO

FIGURE 4 – Exemple de tableau(Tirée depuis MADMC-MOCO-2020-PremièrePartiepar M.Oliver Spanjaard)

Initialisation :

$$\forall j \in [0, N), T[0, j] = (0, 0)$$

$$T[1, 0] = C(sol_1), \text{C le vecteur bi-objectif qui value un objet}$$

Relation de récurrence :

$$E = OPT(F + C(j) \cup G)$$

Complexité :

$$O(nKV)$$

A partir du point départ $T[0,0]$, on va parcourir et remplir les cases du tableau dans l'ordre ligne-colonne. Supposons que les composants sont bornées par V. Le parcours ligne-colonne réalisé par une boucle imriquée donne une complexité $O(nK)$ et en chaque pas on doit faire les calculs et les comparaison qui coûtent en total $O(nKV)$

3.3 Question8

Etant donnée une solution, on doit trouver un α qui maximise la fonction f_I.
Notons :

$$\alpha y_1 + (1-\alpha)y_2 = y_1 x + y_2 y, \text{ où } x = \alpha, y = (1-\alpha)$$

On peut donc formuler un problème de maximisation en forme du programme linéaire :

$$\begin{cases} max\ z = y_1 x + y_2 y \\ y = 1 - x \\ x \in [\alpha_{min}, \alpha_{max}] \end{cases}$$

Si on représente le programme linéaire dans l'espace de décisions, la seule région feasable est la droite $y = 1 - x$:

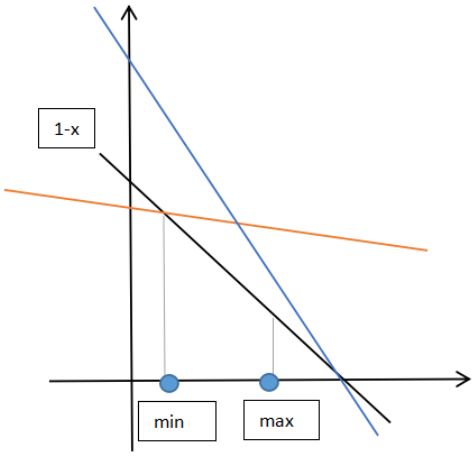

FIGURE 5 – Représentation graphique

Selon la figure, il y a 3 possibilités pour α qui maximise f_I :

$$\begin{cases} y_1 = y_2, & \text{alors } \alpha \text{ ne peut pas influencer la somme} \\ y_1 < y_2, & \text{C'est } \alpha_{min} \text{ qui maximise la somme} \\ y_1 > y_2, & \text{C'est } \alpha_{max} \text{ qui maximise la somme} \end{cases}$$

Donc conlusion : pour un y fixé, c'est toujours α_{min} ou α_{max} qui maximise $\alpha y_1 + (1-\alpha)y_2$

4 Une seconde procédure de résolution

Au lieu de chercher les points au sens Pareto-optimal, dans cette partie on se propose d'utiliser des points non I-dominés. La règle de I-dominance est la suivante :

$$y \text{ I-dominance } y' \text{ si } \begin{cases} \forall \alpha \in I, \alpha y_1 + (1-\alpha)y_2 \leq \alpha y'_1 + (1-\alpha)y'_2 \\ \exists \alpha \in I, \alpha y_1 + (1-\alpha)y_2 < \alpha y'_1 + (1-\alpha)y'_2 \end{cases} \quad (1)$$

4.1 Question10

4.1.1 NI $\subseteq$ ND

On va faire la démonstration par le raisonnement à l'absurde. Soit ND un ensemble des points non-dominés au sens Pareto, NI ensemble des points non I-dominés. Suppsons que $NI \subseteq ND$ n'est pas vérifié. C'est à dire :

$$\exists p \in NI, p \notin ND$$

Selon la définition de l'ensemble des Pareto-optimaux, il existe forcément un vecteur dans l'ensemble ND qui le domine :

$$\exists p' \in ND, p' \succ_p p$$

$$p' \succ_p p \Rightarrow \begin{cases} \forall y \in p, y_{p'} \leq y_p \\ \exists y \in p, y_{p'} < y_p \end{cases}$$

On peut facilement déduire selon la définition d'inégailité que :

$$\forall \alpha \in I(\alpha > 0), \alpha y_1^p + (1-\alpha) y_2^p > \alpha y_1^{p'} + (1-\alpha) y_2^{p'}$$

Donc p' $I-domine$ p. Contradiction.
Conclusion : $NI \subseteq ND$

4.1.2 point minimax ∈ NI

Supposons qu'il n'existe pas de point minimax dans l'ensemble de NI. Comme qu'on a montré dans la question6, on est sûre qu'un point minimax est inclus dans l'ensemble de ND, donc p appartient au complémentaire de NI dans ND :

$$p \in ND \backslash NI$$

A partir de la définition de NI :

$$\exists p' \in NI | \forall \alpha \in I = [\alpha_{min}, \alpha_{max}], \alpha p'_1 + (1-\alpha) p'_2 \leq \alpha p_1 + (1-\alpha) p_2$$

Comme ce qu'on a montré dans la question8, pour y fixé, $max\{\alpha y_1 + (1-\alpha) y_2\}$ est toujours résalisé pour $\alpha = \alpha_{min}$ ou $\alpha = \alpha_{max}$. Selon le raisonnement ci-dessus, le point p' est au moins aussi bon que le point p pour toutes les possibilités de valuations de α, y compris $\alpha = \alpha_{min}$ et $\alpha = \alpha_{max}$. Donc si le point p est un point minimax, alors p' est forcément un point minimax. Donc il existe un point minimax dans NI. Contradiction.
Conclusion : Il existe un point minimax inclus dans NI.

4.2 Question11

$$y \text{ I-dominance } y' \text{ si } \begin{cases} \forall \alpha \in \{\alpha_{min}, \alpha_{max}\}, \alpha y_1 + (1-\alpha) y_2 \leq \alpha y'_1 + (1-\alpha) y'_2 \\ \exists \alpha \in \{\alpha_{min}, \alpha_{max}\}, \alpha y_1 + (1-\alpha) y_2 < \alpha y'_1 + (1-\alpha) y'_2 \end{cases} \tag{2}$$

4.2.1 (1)⟺(2)

Dans cette question, on vousdrait montrer que (1)⟺(2).
On montre tout d'abord (1) ⇒ (2). En sachant que : $\{\alpha_{min}, \alpha_{max}\} \in I$:

$$\forall \alpha \in I, \alpha y_1 + (1-\alpha) y_2 \leq \alpha y'_1 + (1-\alpha) y'_2 \tag{1a}$$

$$\Rightarrow$$

$$\forall \alpha \in \{\alpha_{min}, \alpha_{max}\}, \alpha y_1 + (1-\alpha) y_2 \leq \alpha y'_1 + (1-\alpha) y'_2 \tag{2a}$$

Pour $\forall \alpha \in I$, on peut exprimer α en fonction de α_{min} et α_{max}.

$$\alpha = a * \alpha_{min} + b * \alpha_{max} | a, b > 0 \text{ ET } a + b = 1 \tag{3}$$

Remplacer α dans l'equation 1a, on obtient :

$$a * \alpha_{min} y_1 + (1 - a * \alpha_{min} - b * \alpha_{max}) y_2 \leq a * \alpha_{min} y_1 + (1 - a * \alpha_{min} - b * \alpha_{max}) y'_2 \tag{4}$$

$$(1b) \Rightarrow \begin{cases} \text{a=0, } \alpha = \alpha_{max}, \alpha_{max} y_1 + (1-\alpha_{max}) y_2 \leq \alpha_{max} y'_1 + (1-\alpha_{max}) y'_2 \\ \text{b=0, } \alpha = \alpha_{min}, \alpha_{min} y_1 + (1-\alpha_{min}) y_2 \leq \alpha_{min} y'_1 + (1-\alpha_{min}) y'_2 \end{cases}$$

On a bien montré (1)⇒(2).
Maintenant on va montré l'inverse : (2)⇒(1).

Selon la définition de l'inéquation, on peut multiplier par même nombre strictement positif et l'inéquation reste correcte. Ici $a, b > 0$ et $a + b = 1$.

$$\begin{cases} \alpha_{min}y_1 + (1-\alpha_{min})y_2 \leq \alpha_{min}y'_1 + (1-\alpha_{min})y'_2 \Rightarrow a * \alpha_{min}y_1 + a * (1-\alpha_{min})y_2 \leq a * \alpha_{min}y'_1 + a * (1-\alpha_{min})y'_2 \\ \alpha_{max}y_1 + (1-\alpha_{max})y_2 \leq \alpha_{max}y'_1 + (1-\alpha_{max})y'_2 \Rightarrow b * \alpha_{max}y_1 + b * (1-\alpha_{max})y_2 \leq b * \alpha_{max}y'_1 + b * (1-\alpha_{max})y'_2 \end{cases}$$

L'inégalité est compatible avec l'addition :

$$(a\alpha_{min} + b\alpha_{max})y_1 + (1 - (a\alpha_{min} + b\alpha_{max}))y_2 \leq (a\alpha_{min} + b\alpha_{max})y'_1 + (1 - (a\alpha_{min} + b\alpha_{max}))y'_2$$

$$\Rightarrow$$

$$\alpha y_1 + (1-\alpha)y_2 \leq \alpha y'_1 + (1-\alpha)y'_2 \tag{1a}$$

Comme on a montré dans l'équation (3), tout peut être exprimé en fonction de α_{min} et α_{max}, donc on a bien montré que : (2a) $\Rightarrow$ (1a) et (2b) $\Rightarrow$ (1b).
Fin de démonstration. Conclusion : (1)$\Longleftrightarrow$(2).

4.2.2 Réduction

Dans les questions précédentes, on a montré les idées suivantes :

1. α qui maximise la fonction $f_I(y)$ et soit α_{min}, soit α_{max}.
2. Les points non I-dominées sont forcément non dominés au sens Pareto.
3. Il y a un point minimax dans l'ensemble des points non I-dominées.

On fait donc la réduction suivante : au lieu de rechercher les points non dominées au sens Pareto avec le vecteur-coût de la solution, on prend la nouvelle fonction d'objectif définie comme suivante :

$$f_{I'}(y) = (\alpha_{min}y_1 + (1-\alpha_{min})y_2, \alpha_{max}y_1 + (1-\alpha_{max})y_2)$$

L'ensemble des points Pareto-optimaux trouvés avec cette fonction d'objectif par la programmation dynamique comprend bien les points non I-dominance. La vraie valeur de $max\{\alpha y_1 + (1-\alpha)y_2\}$ est décidée par l'ordre des deux objectifs.

4.3 Question12

Dans cette partie, on compare expérimentalement la première et la seconde procédure. On trace la courbes des temps d'exécution de résolution en fonction de $\alpha_{max} - \alpha_{min}$. Selon la figure, on peut déduire que quand la valeur $\alpha_{max} - \alpha_{min}$ est moins importante, c'est la procédure2 qui procède une meilleure efficacité. En augmentatant cette valeur, la différence entre les deux procédures est de plus en plus négligeable. Quand $\epsilon = 0.5$ et $\alpha_{min} = 0$, il n'y a pas de différence entre les deux procédures.

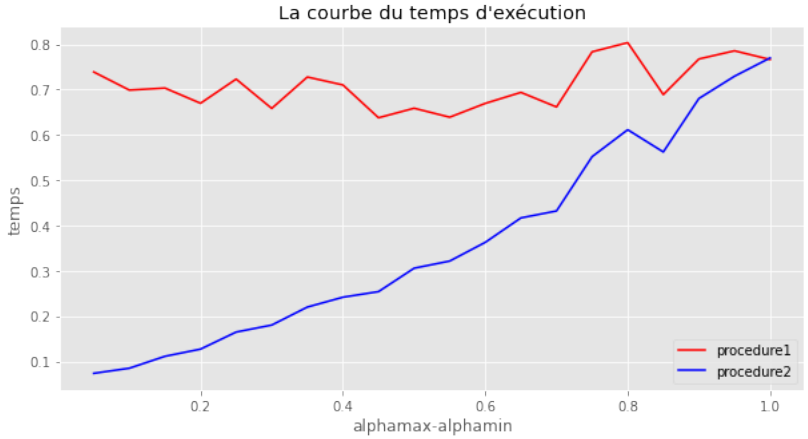

FIGURE 6 – Temps d'exécution des deux procédures en fonction de $\alpha_{max} - \alpha_{min}$