Projet - MADMC
Sélection bi-objectifs avec coefficients intervalles

1. Introduction

On s'intéresse dans ce mini-projet à la version suivante du problème de sélection bi-objectifs (objectifs à minimiser) :

Données :

- n objets,
- un entier k,
- chaque objet i est valué par (c_1^i, c_2^i),
- un intervalle $I = [\alpha_{\min}, \alpha_{\max}]$, avec $\alpha_{\min} \neq \alpha_{\max}$, $\alpha_{\min} < 1$, $\alpha_{\max} > 0$.

Solutions réalisables : tout sous-ensemble de k objets, caractérisé par un vecteur $x = (x_1, \ldots, x_n)$ ($x_i = 1$ si l'objet i est sélectionné, 0 sinon).
But : déterminer une solution réalisable x (dite *minimax* dans la suite) minimisant :

$$f_I(y) = \max \{\alpha y_1 + (1 - \alpha) y_2 : \alpha \in I\}.$$

où $y = c(x) = (\sum_{i=1}^n c_1^i x_i, \sum_{i=1}^n c_2^i x_i)$ est le vecteur-coût de la solution x.

Dans la suite, pour simplifier, on se contentera de déterminer l'image d'une solution minimax dans l'espace des objectifs (cette image est appelée *point minimax*). Le mini-projet consiste à développer deux procédures de résolution du problème, et à comparer leur efficacité.

2. Résultats préliminaires

Pour résoudre le problème, on envisage tout d'abord une procédure de programmation dynamique où l'on déterminerait par récurrence un point minimax pour chacun des sous-problèmes $P(i, j)$ ($1 \leq i \leq n$, $0 \leq j \leq k$). Le sous-problème $P(i, j)$ est la restriction du problème à la sélection de j objets dans $\{1, \ldots, i\}$.

Question 1. A l'aide d'un exemple à trois vecteurs y, y', y'', montrer que le principe d'optimalité n'est pas vérifié, autrement dit qu'on peut avoir simultanément :

$$f_I(y) < f_I(y') \text{ et } f_I(y' + y'') < f_I(y + y'').$$

Comme le principe d'optimalité n'est pas vérifié, on ne peut pas déterminer par récurrence un point minimax. Les sections 3 et 4 décriront des procédures de résolution valides. Au préalable, on va étudier dans cette section deux algorithmes pour déterminer les vecteurs non-dominés (au sens de Pareto) parmi un ensemble de n vecteurs de dimension 2 (ce qui nous sera utile par la suite). On testera ces deux algorithmes sur des ensembles de vecteurs tirés aléatoirement.

Question 2. Implémenter une fonction qui génère un ensemble de n vecteurs où chaque composante est tirée aléatoirement selon une loi normale d'espérance m et d'écart-type $m/4$.

Question 3. Proposer un premier algorithme naïf pour déterminer les vecteurs non-dominés, qui procède avec des comparaisons par paires systématiques. Implémenter cet algorithme.

Question 4. Proposer un second algorithme qui détermine les vecteurs non-dominés en réalisant tout d'abord (1) un tri lexicographique des vecteurs, puis (2) un seul parcours de la liste obtenue pour identifier les vecteurs non-dominés. Implémenter cet algorithme.

Question 5. Comparer les complexités des deux algorithmes proposés, et vérifier expérimentalement votre analyse en traçant les courbes des temps d'exécution respectifs des deux algorithmes en fonction de n, sur des ensembles de vecteurs tirés aléatoirement à l'aide de la fonction de la question 2. Dans les expérimentations, le nombre de vecteurs variera de $n = 200$ à $n = 10000$ (par pas de 200 par exemple), et on prendra $m = 1000$. Pour chaque valeur de n, on fera une moyenne du temps d'exécution sur 50 ensembles tirés aléatoirement.

3. Une première procédure de résolution

Dans cette partie, nous allons mettre en oeuvre une procédure en deux temps pour déterminer l'image dans l'espace des objectifs d'une solution minimax : (1) déterminer les points non-dominés (au sens de Pareto) par programmation dynamique bi-objectifs, puis (2) déterminer un point minimax parmi ceux-ci.

Question 6. Cette approche n'est valide que si un point minimax est inclus dans l'ensemble des points non-dominés. Montrer que c'est bien le cas.

Question 7. Rappeler les relations de récurrence et la procédure de programmation dynamique permettant de calculer l'image des sous-ensembles Pareto-optimaux de taille k d'un ensemble de taille n. Rappeler également comment initialiser la procédure, ainsi que la complexité de la procédure obtenue. Implémenter la procédure de programmation dynamique en utilisant la fonction de la question 4 pour déterminer les points non-dominés en chaque case du tableau de programmation dynamique.

Question 8. Montrer que, pour y fixé, $\max\{\alpha y_1 + (1-\alpha)y_2 : \alpha \in I\}$ est réalisé pour $\alpha = \alpha_{\min}$ ou $\alpha = \alpha_{\max}$. En déduire un algorithme pour déterminer un vecteur minimax dans un ensemble de vecteurs (en dimension 2). Implémenter cet algorithme.

Question 9. Utiliser les fonctions implémentées dans les questions 7 et 8 pour implémenter la procédure en deux temps décrite plus haut.

4. Une seconde procédure de résolution

Afin de tirer parti de la nature de la fonction qu'on cherche à optimiser, on se propose d'utiliser la règle de *I-dominance* suivante à la place de la dominance de Pareto :

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

La seconde procédure de résolution consiste donc à (1) déterminer les points non I-dominés, puis (2) déterminer un point minimax parmi ceux-ci.

Question 10. Soit ND l'ensemble des points non-dominés au sens de Pareto, et NI l'ensemble des points non I-dominés. Montrer que $NI \subseteq ND$ et qu'un point minimax est inclus dans NI.

Question 11. Montrer que y I-domine y' si et seulement si :

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

Montrer alors qu'on peut réduire une instance Π du problème de détermination des points non I-dominés en une instance Π' du problème de détermination des points non-dominés (au sens de Pareto), en utilisant une transformation adéquate de Π. Implémenter cette approche pour permettre la détermination des points non I-dominés.

Remarque : la réduction nécessitera de disposer non seulement d'une fonction de transformation de Π en Π', mais également d'une fonction permettant de reconstituer l'image d'une solution de Π à partir de l'image d'une solution de Π'.

Question 12. Comparer expérimentalement la première et la seconde procédure de résolution en traçant les courbes des temps d'exécution respectifs des deux procédures en fonction de $\alpha_{\max} - \alpha_{\min}$, sur des ensembles de vecteurs tirés aléatoirement à l'aide de la fonction de la question 2. Dans les expérimentations, on fixera $n = 50$, $k = 10$ et $m = 1000$. On considérera les intervalles $I_\varepsilon = [0.5 - \varepsilon, 0.5 + \varepsilon]$ en faisant varier ε de 0.025 à 0.5 (par pas de 0.025 par exemple). Pour chaque intervalle I_ε, on fera une moyenne du temps d'exécution sur 50 instances tirées aléatoirement.

5. Organisation et dates

Le travail est à effectuer par binôme. Le choix du langage de programmation est libre. Toutes les possibilités de visualisation sont les bienvenues (bien que facultatives).

Les projets doivent être soumis le **dimanche 24 janvier 2021** au plus tard sur le lien de soumission du site Moodle de l'UE. Votre livraison sera constituée d'une archive `zip` qui comportera les sources du programme, un fichier README détaillant comment compiler et exécuter le programme, et un rapport (un fichier au format `pdf`) avec les réponses aux différentes questions. Le plan du rapport suivra le plan du sujet.

Une **soutenance** avec diapositives (le choix du logiciel est libre, mais prévoir un fichier au format pdf) est prévue lors de la séance du 2 février 2021.