Sorbonne Université le 25 Novembre 2022
MASTER ANDROIDE M2 – MADMC

Projet MADMC : modèles dépendants du rang pour l'optimisation équitable

Introduction

On s'intéresse ici à un problème d'optimisation multi-objectifs sur un ensemble X défini en compréhension (par exemple par un système de contraintes) dont chaque solution réalisable x est évaluée par n fonctions représentant n points de vues potentiellement différents sur la qualité (ou le coût) de la solution x. Ainsi toute solution x sera évaluée par un vecteur $z(x) = (z_1(x), \ldots, z_n(x))$, $z_i(x)$ représentant la performance (ou le coût) de x selon le point de vue i. Selon les problèmes, ces points de vues pourront représenter des critères d'évaluation (décision multicritère), des évaluations individuelles de la qualité d'une solution (décision multi-agents), des évaluations dans un scénario particulier (décision dans l'incertain). Dans ce contexte, l'optimisation équitable vise à produire des solutions Pareto-optimales realisant un bon équilibre entre les performances attenties sur les différents objectifs.

1) Optimisation d'un OWA et d'un WOWA

Une façon classique d'obtenir par optimisation une solution 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 OWA suivante :

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

où w_i sont des poids positifs et décroissants lorsque i augmente ($w_i \geq w_{i+1}, i = 1, \ldots, n-1$) qui sont fixés par l'utilisateur et $(z_{(1)}, \ldots, z_{(n)})$ représente le résultat d'un tri des composantes de $(z_1(x), \ldots, z_n(x))$ par ordre croissant (ainsi $z_{(i)}(x) \leq z_{(i+1)}(x), i = 1, \ldots, n-1$).

Application au partage équitable de biens indivisibles

On considère un problème où n individus doivent se partager $p \geq n$ objets indivisibles de valeur connue, chaque individu ayant sa propre idée de l'utilité des objets pour lui. On suppose connue la matrice U à n lignes et p colonnes dont l'élément u_{ij} donne l'utilité de l'objet j pour 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. 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 $f(x)$ défini par l'équation (1) (les $z_i(x)$ représentant ici l'utilité de l'ensemble des objets affectés à l'individu i) pour un vecteur poids w donné à composantes strictement décroissantes.

1.1) On considère tout d'abord une instance particulière pour laquelle $p = n = 5$ et on cherche une affectation f-optimale attribuant exactement un objet à chaque individu. La matrice des utilités est la suivante :

$$U = \begin{pmatrix} 12 & 20 & 6 & 5 & 8 \\ 5 & 12 & 6 & 8 & 5 \\ 8 & 5 & 11 & 5 & 6 \\ 6 & 8 & 6 & 11 & 5 \\ 5 & 6 & 8 & 7 & 7 \end{pmatrix} \tag{2}$$

Formuler le problème comme un programme linéaire en variables mixtes. On souhaite alors étudier l'impact de la pondération w sur le caractère équilibré de la solution proposée. Pour cela on utilise une famille de vecteurs de pondération $w(\alpha) \in \mathbb{R}^n$ définie pour tout $\alpha \geq 1$ par les poids :

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

. Etudier comment la solution optimale du problème varie en fonction de α (on pourra notamment, pour chaque solution obtenue, représenter les composantes de la solution optimale sous forme d'un histogramme, ainsi que celles du vecteur de Lorenz). Commenter les résultats obtenus.

1.2) On souhaite maintenant étudier l'évolution du temps de résolution par optimisation OWA du problème de partage équitable en fonction de n et p. Pour $n = 5, 10, 15$ et $p = 5n$ tirer aléatoirement 10 matrices U à coefficients entiers positifs de taille (n, p) et 10 vecteurs poids w à composantes positives strictement décroissantes et calculer le temps moyen de résolution des 10 instances pour chaque couple (n, p). Commenter les résultats obtenus.

1.3) On souhaite maintenant pouvoir attribuer une importance différente aux agents. Pour cela on va utiliser un WOWA (OWA pondéré par un vecteur $p \in \mathbb{R}^n_+$) de la forme :

$$g(x) = \sum_{i=1}^{n} [\varphi(\sum_{k=i}^{n} p_{(k)}) - \varphi(\sum_{k=i+1}^{n} p_{(k)})] z_{(i)}(x)$$

avec une fonction $\varphi(p) = p^\alpha$, $\alpha > 1$. On considère alors de nouveau le problème d'affectation équitable de taille 5 défini par la matrice U de l'équation (2) et on recherche une solution g-optimale pour un vecteur de pondération p définissant l'importance des objectifs. Après avoir formulé ce problème comme un programme linéaire, étudier comment la solution optimale du problème évolue en fonction des pondérations (on choisira d'abord $\alpha = 2$ et on fera évoluer progressivement le vecteur p de $(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5})$ vers chacun des points extrêmes de l'espace des poids $((1, 0, 0, 0, 0), (0, 1, 0, 0, 0), \ldots)$. On effectuera ensuite la même analyse avec $\alpha = 5$).

1.4) Reprendre la question 1.2 pour observer cette fois l'évolution du temps de calcul d'une solution g-optimale (on choisira des vecteurs p tirés aléatoirement).

2) Optimisation d'une intégrale de Choquet

On s'intéresse maintenant à rechercher une solution équilibrée en optimisant une intégrale de Choquet $h(x) = C_v(z(x))$ utilisant une capacité v qui est une fonction de croyance.

2.1) Expliquer pourquoi maximiser $h(x)$ favorise l'obtention de solutions x dont le vecteur $z(x)$ a des composantes équilibrées.

Application à la selection multicritère de projets

Le responsable d'une organisation cherche à sélectionner des projets (parmi une liste de p projets soumis). Chaque projet a un coût connu $c_k, k \in \{1, \ldots, p\}$ 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. Ici encore ces utilités sont supposées additives et l'aptitude d'un ensemble de projets x à satisfaire l'objectif i (notée $z_i(x)$) est définie comme la somme des utilités u_{ij} des projets j selectionnés. Le but étant de trouver une solution équilibrée entre la satisfaction des différents objectifs, on cherche le sous ensemble x qui maximise $C_v(z(x))$ pour une capacité v qui est une fonction de croyance.

2.2) Formuler ce problème comme un programme linéaire en variables mixtes. Utiliser cette formulation pour rechercher une solution équilibrée pour l'instance bi-critère avec $p = 4$ projets caractérisée par le vecteur de coûts $c = (40, 50, 60, 50)$ et la matrice d'utlité :

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

On donnera quelques exemples de solutions trouvées pour des capacités v tirées au hasard parmi les fonctions de croyance. On comparera ces solutions avec celle obtenue en maximisant la satisfaction moyenne des objectifs définie par $(z_1(x) + z_2(x))/2$.

2.3) On souhaite étudier l'évolution du temps de résolution en fonction de n et p. Pour $n = 2, 5, 10$ et $p = 5, 10, 15, 20$ tirer aléatoirement 10 matrices U à coefficients entiers positifs de taille (n, p), des coûts c_k positifs et une fonction de croyance puis calculer le temps moyen de résolution des 10 instances pour chaque couple (n, p). Commenter les résultats obtenus.

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

On se place maintenant 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 $S = \{1, \ldots, n\}$ de scénarios possibles et le temps pour parcourir chaque arc (i, j) du graphe dans les différents scénarios est donné par le vecteur $(t^1_{ij}, \ldots, t^n_{ij})$ où t^s_{ij} 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 $t^s(P) = \sum_{(i,j)\in P} t^s_{ij}$. Ne sachant pas quel scénario va se produire et ne connaissant même pas leur probabilité, on va chercher un chemin robuste, c'est-à-dire qui reste rapide dans les différents scénarios. Pour cela on cherche un chemin P qui maximise $C_v(-t^1(P), \ldots, -t^n(P))$ où v est une fonction de croyance. Proposer un programme linéaire pour calculer un tel chemin. Appliquer le au graphe de la figure 1 pour obtenir un chemin robuste de a à g (on pourra essayer différentes fonctions de croyance et observer l'impact sur le résultat).

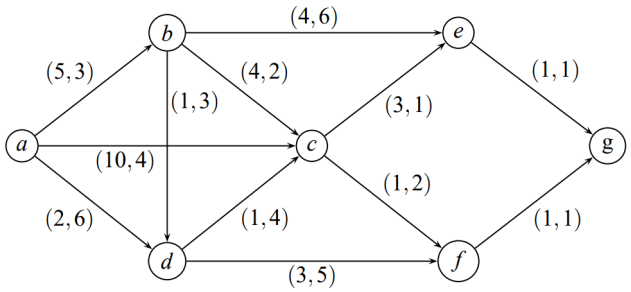

FIGURE 1 – Une instance du problème de chemin robuste à 2 scénarios

Modalités de travail, délivrables et deadlines

Le travail est à effectuer en binôme. Le binôme devra être déclaré par mail à Patrice Perny (objet du mail : binome MADMC-22) au plus tard le 12 décembre à 2022. Les projets seront déposés au plus tard le 22 Janvier 2023 à minuit sur le site moodle de MADMC. Votre livraison sera constituée d'une archive zip nommée numgroupe_nom1_nom2.zip qui comportera les sources du programme, un fichier README détaillant comment exécuter le programme, et un rapport rédigé (un fichier au format pdf nommé numgroupe_nom1_nom2.pdf) qui présentera le travail effectué et les réponses aux différentes questions. Le plan du rapport suivra le plan du sujet. Il est fortement recommandé de rédiger son rapport en LaTeX. Les projets rendus feront l'objet d'une brève soutenance le 24 Janvier 2023.

Accès au solveur Gurobi et implémentation en python

Il est fortement conseillé d'implanter vos programmes de test en python en faisant appel à *Gurobi solver* pour les résolutions de programmes linéaires (http ://www.gurobi.com/). Ce solveur est installé dans les salles de tme mais peut aussi être installé sur des machines personnelles en téléchargeant depuis une adresse de Sorbonne Université avec une licence étudiant (gratuite).