Sorbonne Université 26 Octobre 2022
UE LU3IN034

Devoir MOIARO : approche égalitariste de l'affectation

On considère un problème d'affectation qui consiste à attribuer n objets $O = \{o_1, \ldots, o_n\}$ à n agents $A = \{a_1, \ldots, a_n\}$ (un objet par agent et un agent par objet) tout en maximisant la satisfaction des agents. La satisfaction individuelle d'un agent dépend de l'objet qu'il reçoit mais pas des objets que les autres reçoivent. Les préférences des agents sont représentées par des fonctions $u_i : O \to \mathbb{N}$ où $u_i(o_j)$ représente l'utilité de o_j pour a_i pour tout couple (i, j). Une instance de ce problème d'affectation est donc caractérisée par une matrice U de taille n dans laquelle le terme U_{ij} en ligne i et colonne j représente $u_i(o_j)$. Considérons par exemple la matrice suivante composée de notes choisies dans l'intervalle $[0, 20]$:

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

Formellement, une affectation peut être représentée par une bijection $\pi : A \to O$ dans laquelle $\pi(a_i) = o_j$ si o_j est affecté à l'agent a_i. Par exemple l'affectation $\pi(a_1) = o_2, \pi(a_2) = o_1, \pi(a_3) = o_3$, $\pi(a_4) = o_4$ and $\pi(a_5) = o_5$ conduit au vecteur d'utilités $u(\pi) = (20, 5, 11, 11, 7)$ donnant la satisfaction des différents agents. Cette affectation π conduit à une somme de satisfactions de 54 et donc à une satisfaction moyenne de 10.8.

1) Ecrire un programme linéaire qui permette de calculer une affectation qui maximise la satisfaction moyenne et appliquer le pour résoudre l'instance donnée ci-dessus. On développera un code python qui permette de résoudre ce problème avec un solveur de programmation linéaire (par exemple Gurobi) et on observera comment le temps de résolution évolue en fonction de la taille de l'instance considérée (on fera des moyennes de temps sur 10 instances de même taille et on fera augmenter la taille de 5 en 5).

On peut remarquer que la solution π donnée ci-dessus n'est pas très équitable (bien qu'elle conduise à une bonne satisfaction moyenne) car le vecteur de satisfaction est assez déséquilibré (l'agent 1 est 4 fois plus satisfait que l'agent 2). On s'intéresse maintenant à rechercher des affectations qui répartissent de manière plus équilibrée les satisfactions entre agents. Par exemple, une autre affectation réalisable est π' definie par $\pi'(a_i) = o_i$ pour $i = 1, \ldots, 5$. Bien que π' soit sous-optimale au sens du critère précédent (avec un score de 53/5), elle conduit à un vecteur de satisfactions $u(\pi') = (12, 12, 11, 11, 7)$ qui est plus équilibré que celui de π. En particulier, on améliore la valeur de la pire composante. Pour aller dans ce sens on s'intéresse à chercher l'affectation qui maximise la satisfaction de l'agent le moins satisfait (critère du maximin qui caractérise l'approche égalitariste).

2) Pour aborder l'optimisation maximin, dans un premier temps on s'intéresse au problème P_λ (pour une valeur de λ donnée dans l'intervalle $[0, 20]$) qui consiste à rechercher s'il existe une affectation π telle que $u_i(\pi(a_i)) \geq \lambda$ pour tous les $i \in \{1, \ldots, n\}$. Modéliser ce problème comme un problème de flot maximum dans un réseau que l'on précisera. Résoudre le problème à la main avec l'algorithme de Ford-Fulkerson pour l'instance donnée ci-dessus et la valeur $\lambda = 8$.

3) En vous servant de la question 2 expliquer comment l'on peut trouver une affectation optimale au sens du maximin en se fondant sur des calculs de flots et déterminer une telle affectation pour l'instance ci-dessus.

4) On recherche maintenant une affectation π qui maximise la fonction :

$$f(\pi) = \min_i \{u_i(\pi(a_i))\} + \epsilon \sum_i u_i(\pi(a_i))$$

où ϵ est une quantité positive choisie arbitrairement petite. Proposer une formulation du problème de la recherche d'une affectation optimale selon f par la programmation linéaire. Faire un code python utilisant Gurobi pour résoudre ce problème. Donner une instance admettant une affectation optimale au sens du maximin qui ne maximise pas f. Quel est l'avantage d'optimiser la fonction f plutôt que d'utiliser le maximin ?

5) On définit maintenant le regret $r_i(\pi)$ d'un agent i par rapport à une affectation π par l'écart d'utilité entre son objet préféré et celui qui lui est affecté (formellement $r_i(\pi) = \max_j\{u_i(o_j)\} - u_i(\pi(a_i))$. On souhaite alors déterminer l'affectation qui minimise le plus grand regret, c'est-à-dire la fonction :

$$g(\pi) = \max_i r_i(\pi)$$

Proposer une formulation du problème de la recherche d'une affectation optimale selon g par la programmation linéaire. Faire un code python utilisant Gurobi pour résoudre ce problème. Donner un exemple de solution optimale au sens de g qui ne l'est pas au sens de f sur une instance de votre choix.

6) Etudier comment évolue le temps de calcul d'une affectation optimale au sens de f, puis au sens de g, en fonction de la taille de la matrice U. Ici encore, on fera des moyennes de temps sur 10 instances de même taille et on fera augmenter la taille de 5 en 5.

Modalités de travail et délivrables

Le travail est à effectuer en binôme. Le binôme constitué devra être déclaré par mail à patrice.perny@lip6.fr (objet du mail : binome MOIARO-22) avant le mercredi 9 Novembre 2022. Les projets seront déposés au plus tard le mardi 6 décembre 2022 à minuit sur le site moodle de MOIARO. Votre livraison sera constituée d'une archive zip nommée 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 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.

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 en libre-service permanent mais peut aussi être installé sur des machines personnelles en téléchargeant depuis une adresse de Sorbonne Université avec une licence étudiant (gratuite).