Université Paris 6 2014-2015
MASTER ANDROIDE M1 – MOGPL

Projet MOGPL

Algorithmes pour le partage équitable de biens indivisibles

Introduction et objectifs

Le problème du partage équitable est l'abstraction d'un problème réel consistant à diviser équitablement des biens, des ressources ou des tâches entre des agents. On se fixera ici sur l'exemple de biens indivisibles et sur le problème de leur allocation aux agents. On se place du point de vue d'un superviseur (agent, logiciel de contrôle) qui connait l'utilité des biens pour chaque agent et qui doit déterminer une allocation optimale. L'objet du projet est d'implanter et tester des procédures algorithmiques qui, en s'appuyant sur la programmation linéaire et l'algorithmique des graphes, permettent au superviseur de déterminer un partage équitable des biens disponibles (la notion d'équité restant à préciser). Les hypothèses sur l'évaluation des biens ou des ressources à partager sont les suivantes :

- chaque agent a une opinion sur la valeur de chaque partie des biens ou des ressources,
- pour chaque agent, la valeur d'un ensemble de biens est la somme des valeurs qu'il attribue à chaque bien. La valeur que l'agent i accorde au bien j est appelée "utilité du bien j pour l'agent i" et est supposée connue,
- chaque bien est indivisible et ne peut être affecté simultanément à deux agents. En revanche, dans certains cas, on pourra supposer que plusieurs exemplaires de chaque bien sont disponibles, chacun pouvant être affecté à un individu distinct.

Sous ces hypothèses, l'objet du travail est d'étudier l'intérêt relatif de différents critères proposés pour résoudre ce problème, de développer des procédures permettant de calculer une allocation optimale des biens aux agents selon ces critères, enfin de tester ces procédures sur des instances du problème engendrées aléatoirement.

Première modélisation du problème

Soit n agents $a_1, \ldots, a_n$ et m biens $b_1, \ldots, b_m$. Soit $u_{ij} \in [0, M]$ l'utilité du bien j pour l'agent i. On suppose que la matrice des termes u_{ij} est connue. Les stratégies d'allocation des biens sont définies par des variables $x_{ij} \in \{0, 1\}$ telles que $x_{ij} = 1$ si la ressource j est affectée à l'activité i et 0 sinon. La matrice (n, m) de terme général x_{ij} sera notée x et caractérise une solution potentielle. On suppose qu'il existe β_j exemplaires du bien b_j et que chaque agent peut recevoir au plus α_i biens (ceux-ci devant être tous distincts). Dans un premier temps, on va traiter le problème d'affectation simple qui se caractérise par les contraintes additionnelles suivantes : $m = n$ (autant de biens que d'agents), $\alpha_i = 1, i = 1 \ldots n$ (un même agent ne peut recevoir plus d'un bien), $\beta_j = 1, j = 1 \ldots m$ (un bien ne peut être affecté à plusieurs agents).

1°) On souhaite répartir tous les biens sur les agents de manière à maximiser la satisfaction moyenne des agents. Modéliser ce programme comme un programme linéaire que l'on notera $\mathcal{P}_0$.

2°) Ecrire un programme qui engendre aléatoirement une instance de taille (n, m) pour une valeur M donnée et qui calcule une allocation optimale à l'aide de gurobi. On tirera aléatoirement les utilités u_{ij} comme des entiers de l'intervalle [0, M] en utilisant la loi triangulaire centrée sur $M/2$ (voir par exemple http ://fr.wikipedia.org/wiki/Loi_triangulaire).

3°) Effectuer des essais numériques de résolution de $\mathcal{P}_0$ pour $M = 10$, $n = 10, 50, 100, 500, 1000$. Pour chaque valeur de n on tirera 10 instances aléatoirement et on donnera dans un tableau les quantités suivantes :

n	t	moy/M	min/M	max/M
10				
50				
100				
500				
1000				

où t est le temps de résolution moyen (en secondes), moy est la moyenne des satisfactions des individus, min est la satisfaction de l'agent le moins satisfait, max la satisfaction de l'agent le plus satisfait.

Réitérer l'expérience avec $M = 100$, puis avec $M = 1000$. Commenter les résultats et leur éventuelle évolution en fonction de n et M.

Approche égalitariste

Dans l'approche précédente, rien ne garantit que les solutions trouvées partagent équitablement les biens entre les agents. Pour ce faire il faut utiliser d'autres modélisations. Soit $z_i(x) = \sum_{j=1}^{m} u_{ij}x_{ij}$ la satisfaction de l'agent i. On souhaite maintenant trouver une affectation égalitariste, c'est-à-dire qui maximise la satisfaction de l'agent le moins satisfait soit :

$$\max_x \min_i z_i(x)$$

4°) Dans un premier temps on s'intéresse à l'existence d'une affectation dans laquelle les satisfactions des agents seraient toutes supérieures ou égales à λ, pour un λ donné choisi dans $[0, M]$. Montrer que ce problème peut s'écrire comme un problème de recherche de flot maximum dans un graphe. En déduire un algorithme pour trouver l'affectation égalitariste. On implantera cet algorithme en utilisant une bibliothèque de graphe permettant le calcul du flot maximum en temps polynomial (voir avant-dernière section du document).

5°) Formuler le problème de l'affectation égalitariste comme un programme linéaire $\mathcal{P}_1$ en variables mixtes.

6°) Effectuer des essais numériques de résolution du problème d'affectation égalitariste pour des instances avec $M = 100$, $n = m = 10, 50, 100$. Pour chaque valeur de n on tirera 10 instances aléatoirement et on comparera les temps d'exécution des deux méthodes envisagées aux questions 4°) et 5°).

7°) En étudiant la résolution de quelques instances de taille $n = 5$ ou $n = 10$, comparer les vecteurs de satisfaction des agents obtenus en maximisant la satisfaction moyenne des agents (voir question 3°) à ceux obtenus avec l'affectation maxmin (méthode de la question 5). Expliquer les avantages et inconvénients relatifs des deux méthodes dans un contexte d'allocation de biens.

8°) On souhaite maintenant trouver l'affectation qui maximise le critère suivant :

$$\max_x \min_i z_i(x) + \varepsilon \sum_{i=1}^{n} z_i(x)$$

où $\varepsilon > 0$ est une quantité très petite devant 1. Formuler ce problème comme un programme linéaire $\mathcal{P}_2$ en variables mixtes.

A l'aide des programmes $\mathcal{P}_1$ et $\mathcal{P}_2$, trouver une instance où les solutions optimales x^1 et x^2 des deux programmes vérifient $z_i(x^1) \leq z_i(x^2)$, $i = 1, \ldots, n$, l'une des inégalités étant strictes. En conséquence, selon vous, quel est, de $\mathcal{P}_1$ ou $\mathcal{P}_2$, le programme linéaire qui est susceptible de fournir les solutions les plus intéressantes ?

Approche égalitariste en regrets

Pour chaque agent on note z_i^* le maximum de la fonction $z_i(x)$ sur l'ensemble des affectations admissibles. Cette quantité représente le niveau maximum de satisfaction que l'agent i peut atteindre sur une instance donnée. Dans un problème d'affectation simple, on a $z_i^* = \max\{u_{ij}, j = 1, \ldots, m\}$. Comme $z_i(x)$ mesure la satisfaction de l'agent i pour l'affectation x, la quantité $r_i(x) = z_i^* - z_i(x)$ mesure le regret que la solution x engendre chez l'agent i. On s'intéresse alors à trouver l'affectation qui minimise le

plus grand regret (minimax regret) sur l'ensemble des agents, c'est-à-dire qu'on veut résoudre le problème d'optimisation suivant :

$$\min_x \max_i r_i(x)$$

9°) Formuler le problème de l'affectation minmax regret comme un programme linéaire $\mathcal{P}_3$ en variables mixtes.

10°) En vous inspirant de la question 4°), expliquer comment résoudre le problème de l'affectation minimax regret sans recourir à la programmation linéaire mais en utilisant un algorithme de flot maximum dans un graphe. Implanter cette méthode.

11°) Effectuer des essais numériques de résolution du problème d'affectation minmax regret pour des instances avec $M = 100$, $n = m = 10, 50, 100$. Pour chaque valeur de n on tirera 10 instances aléatoirement et on comparera les temps d'exécution des deux méthodes envisagées aux questions 9°) et 10°).

Extension à l'affectation multiple

On considère maintenant que m n'est pas nécessairement égal à n. On suppose qu'il existe β_j exemplaires du bien b_j et que chaque agent peut recevoir plusieurs biens, au plus α_i (ceux-ci devant être tous distincts). On souhaite alors résoudre le problème d'affectation multiple qui généralise le précédent au cas où plusieurs biens peuvent être affectés à chaque agent, toujours sous l'angle de l'équité.

12°) Expliquer, *en quelques phrases*, quelles sont les approches qui peuvent s'adapter à ce cas plus général parmi celles précédemment décrites en programmation linéaire et en algorithmique des graphes. On précisera notamment comment on doit s'y prendre pour résoudre le problème de l'affectation multiple avec les critères maxmin, minmax regret. On ne fera pas ici d'implémentation ni de test.

Pour les plus rapides...

Nous avons prévu quelques questions plus avancées sur le sujet, destinées aux plus rapides. Ces questions peuvent être obtenues en envoyant un mail à patrice.perny@lip6.fr comportant en fichier attaché le délivrable concernant les autres parties. Ce mail devra être envoyé au plus tard le mercredi 24 novembre.

Accès au solveur Gurobi et à une bibliothèque pour les graphes

Pour résoudre les programmes linéaires, le solveur gurobi (http ://www.gurobi.com/) a été installé dans la salle en libre-service permanent 31-305. Les étudiants de MOGPL ont la possibilité de travailler sur gurobi soit en salle libre-service 31-305, soit dans les salles 31-308 et 31-309 en dehors du déroulement des enseignements dans ces deux salles. Les étudiants de MOGPL peuvent également atteindre les machines des salles 31-305, 31-308, 31-309 et 31-207 à distance en ssh pour exécuter gurobi, que ce soit depuis une autre salle TME ou depuis l'extérieur en se connectant au préalable à la passerelle ssh.ufr-info-p6.jussieu.fr. Pour utiliser l'application gurobi 563 installée dans /opt/gurobi563/ il est nécessaire d'avoir dans son .bashrc :

```
export GUROBI_HOME="/opt/gurobi563/linux64"
export PATH="${PATH}:${GUROBI_HOME}/bin"
export LD_LIBRARY_PATH="${LD_LIBRARY_PATH}: ${GUROBI_HOME}/lib"
```

Concernant les algorithmes de graphes pour la recherche d'un flot maximum, on pourra faire appel à la librairie pygraph (module minmax) de python qui est disponible sur les machines, ou toute autre librairie fournissant un algorithme de flot maximum. A noter qu'en python, la loi triangulaire évoquée en section 2°) peut être facilement mise en oeuvre pour le tirage aléatoire des utilités avec la fonction random.triangular.

Organisation et dates

- le language d'implémentation recommandé est python, d'autres options sont possibles
- les projets doivent s'effectuer en binôme (étudiants du même groupe pour faciliter les soutenances)
- le projet doit être rendu le **1er décembre 2014**. Votre livraison sera constituée d'une archive zip qui comportera les sources du programme et un rapport au format pdf avec les réponses aux différentes questions. Le plan du rapport suivra le plan du sujet. L'archive zip devra être adressée par mail à l'adresse correspondant à votre numéro de groupe dans le tableau ci-dessous (deux destinataires pour le groupe 4), en respectant la convention indiquée pour le sujet du mail :

Groupe	Mail	Sujet du mail
1	`patrice.perny@lip6.fr`	Projet MOGPL GR1
2	`safia.kedad-sidhoum@lip6.fr`	Projet MOGPL GR2
3	`olivier.spanjaard@lip6.fr`	Projet MOGPL GR3
4	`patrice.perny@lip6.fr,olivier.spanjaard@lip6.fr`	Projet MOGPL GR4

- la soutenance du projet est prévue à l'heure du TD la semaine du 8 décembre.