Complexité et Calculabilité : projet 2023/2024
Partage équitable de graphe

1 Introduction

Ce document présente le projet du cours CoCa (Complexité et calculabilité) de 2023. Il s'intéresse au problème `PART-EQUI`, défini en section 3.

Il consiste en une partie théorique, décrite plus loin dans ce document, et une partie implémentation qui est à effectuer à partir d'une base de code qui est fournie dans l'archive qui accompagne ce document.

Le code fourni est documenté et dispose d'un README qui devrait contenir toutes les informations nécessaires à sa prise en main. Dans ce code, seul le dossier `src/EquitableRepartitionProblem` contient les fichiers que vous aurez à modifier. Le code fourni contient également le dossier `src/ColouringProblem` qui implémente un brute force et une réduction pour le problème de coloriage vu en cours (juste à titre d'exemple de réduction vers SAT).

Le dossier `examples` contient des codes utilisant les fonctions que vous aurez à utiliser pour servir d'exemple.

Il est plus que conseillé de traiter une partie théorique en même temps que la partie du code correspondante, de manière à pouvoir tester votre partie théorique.

La section 3 présente le problème `PART-EQUI` en détail.

La section 4.1 consiste à déterminer un vérificateur et un algorithme brute-force qui résout `PART-EQUI`. Il correspond à l'implémentation du fichier `RepartitionResolution.c`.

La section 4.2 consiste à déterminer une réduction d'un problème NP-complet vers `PART-EQUI`, et ne correspond à aucune implémentation.

La section 5 consiste à déterminer une réduction de `PART-EQUI` vers SAT, et est à implémenter dans le fichier `RepartitionReduction.c`.

2 Consignes de rendu

Le projet est à effectuer en binômes. Vous devez déclarer vos binômes sur le moodle du cours[1]. Si vous ne pouvez pas vous mettre en binôme, vous en demanderez l'autorisation à votre chargé de TD qui, s'il en donne l'autorisation, fera la manipulation nécessaire.

La date butoir de ce projet est le **22 novembre 2023**. Vous devez rendre sur le moodle du cours les documents suivants :

— Un fichier pdf contenant les réponses aux questions présentes dans ce sujet. Pour les dernières questions des sections 4.1 et 5, vous inclurez un rapport d'implémentation décrivant l'état d'avancement de votre implémentation si celle-ci n'est pas correcte, une explication de ce qui fonctionne ou non, des difficultés rencontrées, ainsi qu'une liste des instances sur lesquelles vous avez testé votre programme et si le résultat est correct ou non.

— Le dossier src/EquitableRepartitionProblem du code dûment complété avec votre implémentation, et **uniquement** ce dossier. Vous n'êtes pas autorisés à modifier les fichiers .h dans include – cela étant inutile.

3 Présentation du problème

Le problème du partage équitable, connu aussi sous le nom de problème de partage du gâteau (de l'anglais cake cutting problem), est le problème du partage d'une ressource de telle sorte que tous les participants estiment en avoir reçu une part « satisfaisante ».

On formalise ceci de la manière suivante. Une instance du problème est la donnée de :

— un graphe $G = (V, E)$, qui représente la ressource à partager. Chaque noeud $v \in V$ est appelé une ressource élementaire. On note $n = |V|$;

— $K = \{1, 2, \ldots, k\}$, un ensemble de k participants ;

— pour chaque participant $i \in K$, une fonction de valeur $\mathcal{P}_i : V \to \mathbb{N}$, avec la contrainte

$$\sum_{v \in V} \mathcal{P}_i(v) = \sum_{v \in V} \mathcal{P}_j(v)$$

pour tout $i, j \in K$.

Nous notons $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$ l'ensemble de ces fonctions de valeur ;

Lorsque l'on considère un ensemble de ressources élémentaires (nœuds) $X \subseteq V$, la valeur attribuée par le participant i à X est la somme des valeurs

1. `https://moodle.u-bordeaux.fr/course/view.php?id=11029`

attribuées aux éléments de X. On note cette valeur $\mathcal{P}_i(X) \stackrel{\text{def}}{=\!=} \sum_{v \in X} \mathcal{P}_i(v)$.
Rappel : $\mathcal{P}_j(V) = \mathcal{P}_{j'}(V)$ pour tous participants $j, j' \in K$.

Une allocation est une partition $V = V_1 \cup V_2 \cup \cdots \cup V_k$, où chaque V_i correspond à la part donnée au participant i. Nous souhaitons trouver une façon équitable de partager la ressource. Pour cela nous définissions :

— Une allocation sera dite valide si chaque sous-graphe (V_i, E) est **connexe**.

— Une allocation sera dite équitable si chaque participant estime avoir reçu au moins une fraction $\frac{1}{k}$ de la valeur totale qu'il accorde au graphe. En symbole, nous voulons :

$$\forall i \in K,\ \mathcal{P}_i(V_i) \geq \frac{\mathcal{P}_i(V)}{k}\,.$$

Le problème `PART-EQUI` qui nous intéresse est le suivant :

Entrée : Un graphe $G = (V, E)$, un entier k et des fonctions de valeur $\mathcal{P}_1, \ldots, \mathcal{P}_k$.

Sortie : Existe-t-il une allocation à la fois valide et équitable ?

4 Complexité de `PART-EQUI`

4.1 Vérificateur et algorithme "brute-force"

1. Dans un premier temps, écrivez en pseudo-code un vérificateur, c'est-à-dire un algorithme qui prend en entrée une instance du problème (un graphe, un nombre de participants k et des fonctions de préférences) et une allocation (une partition du graphe en k sous-ensembles de sommets), et indique si cette allocation est bien valide et équitable.
2. Quelle est la complexité de votre algorithme ? Justifiez.
3. Qu'en déduisez-en sur la complexité du problème `PART-EQUI` ? Justifiez votre réponse.
4. Utilisez le vérificateur de la question précédente pour écrire un algorithme "brute-force" qui, étant donnée une instance de `PART-EQUI`, indique si il existe une solution à cette instance du problème. Votre programme doit énumérer toutes les allocations possibles et les vérifier une à une.
5. Quelle est la complexité de cet algorithme ?
6. (À faire en parallèle du reste) Implémentez les fonctions nécessaires du fichier `RepartitionResolution.c` (en ajoutant des fonctions auxiliaires).

4.2 PART-EQUI et PARTITION

On rappelle que PARTITION correspond au problème suivant :
— **Donnée :** des entiers $x_1, \ldots, x_n > 0$.
— **Question :** existe-t-il $X \subseteq \{1, \ldots, n\}$ tel que

$$\sum_{i \in X} x_i = \sum_{i \notin X} x_i$$

Le but de cette partie est de donner une réduction polynomiale de PARTITION vers PART-EQUI. On rappelle que PARTITION est NP-complet.

1. Étant donné une instance de PART-EQUI, si le graphe est complet, que devient la contrainte "l'allocation est valide" ?
2. Étant donné une instance de PART-EQUI telle que le graphe complet et que tous les participants ont la même fonction de préférence, à quoi correspond une allocation valide et équitable ?
3. Donnez une réduction polynomiale de PARTITION vers PART-EQUI, en justifiant la réponse.
4. Qu'en déduisez-vous sur la complexité de PART-EQUI ? Justifiez.

5 Réduction à SAT

On simplifie PART-EQUI : les préférences sont à valeurs dans $\{0, 1\}$. Le problème reste NP-complet, mais plus facile à encoder dans SAT.

Pour simplifier la notation, on va écrire $[n]$ pour l'ensemble $\{1, \ldots, n\}$, pour tout entier non-négatif n.

On propose d'utiliser les variables suivantes pour encoder le problème :
— Pour tout noeud du graphe $i \in [n]$ et tout participant $j \in K$, une variable booléenne V_i^j signifiant que "le noeud i est alloué au participant j".
— Pour tous noeuds $i, i', \ell \in [n]$ et participant $j \in K$, une variable $C_{i,i',\ell}^j$ dont le rôle est explicité au paragraphe 5.2.
— Pour tout noeud $i \in [n]$, participant $j \in K$ et entier $1 \leq s \leq \lceil \frac{m}{k} \rceil$ avec $m \stackrel{\text{def}}{=\!=} \mathcal{P}_i(V)$ la valeur totale de chaque participant, une variable $M_{i,s}^j$ dont le rôle est explicité au paragraphe 5.3.

5.1 L'allocation est une partition

1. Pour un sommet i, donner une formule SAT qui exprime que ce sommet du graphe est alloué à au moins u participant :

2. Donner alors une formule SAT qui exprime le fait que chaque sommet a été alloué à au moins un participant.
3. Pour un sommet i, donner une formule SAT qui exprime que ce sommet est alloué à au plus un participant.
4. Donner alors la formule totale pour exprimer que l'allocation est bien une partition du graphe.

5.2 L'allocation est connexe pour chaque participant

Pour gérer la connectivité de chaque part, on va simuler l'algorithme de Floyd-Warshall. Pour chaque participant $j \in K$, couple de sommets $i, i' \in V$ et entier $\ell \in 0, \ldots, n$, on utilise la variable $C^j_{i,i',\ell}$ avec la signification suivante :

$C^j_{i,i',\ell}$ vaut 1 ssi il est possible d'aller du sommet i au sommet i' en n'empruntant que des sommets alloués à j et tel que les sommets intermédiaires du chemin soient dans l'ensemble $\{1, 2, 3, \ldots, \ell\}$.

1. Exprimer dans SAT l'implication suivante : "si les sommets i et i' ont été alloués au participant j, alors il est possible d'aller du sommet i au sommet i' en n'empruntant que des sommets alloués à j. On l'exprimera en fonction des variables $V^j_i, V^j_{i'}, C^j_{i,i',n}$.
2. Dans quel cas $C^j_{i,i',0}$ vaut 0 ? vaut 1 ? Donner la clause SAT correspondante à ajouter au graphe, en fonction de $V^j_i, V^j_{i'}$, dans le cas où l'arête (i, i') est présente dans le graphe G. Que rajouter si $(i, i') \notin E$?
3. Exprimer enfin $C^j_{i,i',\ell}$ en fonction de $C^j_{i,i',\ell-1}, C^j_{i,\ell,\ell-1}$, et $C^j_{\ell,i',\ell-1}$.

5.3 Chacun est satisfait par sa part

Rappel : $m = \mathcal{P}_i(V) = \mathcal{P}_{j'}(V)$ pour tous les participants $j, j' \in K$; la valeur m est forcément inférieure à n car les préférences sont dans $\{0, 1\}$.

Pour qu'un participant j soit satisfait par son allocation, il faut que ce participant ait obtenu un ensemble de sommets V_j tel que $\mathcal{P}_j(V_j) \geq C$ avec $C \stackrel{\text{def}}{=\!=} \lceil \frac{m}{k} \rceil$. En notant $P_j \stackrel{\text{def}}{=\!=} \{v \in V, \mathcal{P}_j(v) = 1\}$ cela revient à dire que $|V_j \cap P_j| \geq C$ pour tout participant j.

Pour encoder le fait que ce participant soit satisfait ou pas, nous pouvons énumérer les sommets de préférence 1 qui lui ont été attribués, donc les sommets de $V_j \cap P_j$, et nous assurer qu'il y en a au moins C. Pour cela, nous utilisons les variables $M^j_{i,s}$, $i \in P_j$, $1 \leq s \leq C$ avec la signification suivante : $M^j_{i,s}$ vaut 1 ssi le sommet i est le s-ème sommet de $V_j \cap P_j$.

1. Comment exprimer en formule SAT le fait qu'un sommet i n'est pas à la fois le s-ème et le s'-ème sommet de $V_j \cap P_j$ pour $s \neq s'$?

2. Comment exprimer en formule SAT le fait que pour chaque $1 \leq s \leq C$ il y a bien un sommet i qui est le s-ème sommet de $V_j \cap P_j$?
3. Expliquer pourquoi si les deux formules précédentes sont satisfaites, alors le participant j est satisfait par l'allocation.

5.4 Implémentation

Dans le fichier `RepartitionReduction.c`, implémentez la réduction décrite plus haut.