TD 10 : Algorithmes d'approximation

Soit $G = (V, E)$ un graphe connexe et non-orienté. On associe à G une fonction de poids $w : E \to \mathbb{R}^+$. Pour un sous-ensemble d'arêtes $M \subseteq E$, on note $w(M) = \sum_{e \in M} w(e)$.

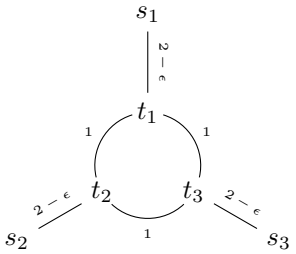
1 Multiway Cut

Etant donné un ensemble des noeuds $S = \{s_1, \ldots, s_k\} \subseteq V$, une coupe *multi-way* (angl. *multi-way cut*) est un ensemble $M \subseteq E$ d'arêtes t.q. s_i et s_j ne sont pas dans la même composante connexe dans le graphe $G' = (V, E - M)$ ($\forall i \neq j$). Le problème de la coupe *multi-way* optimale est de trouver une coupe *multi-way* M de taille $w(M)$ minimale.

Pour $k \geq 3$ (fixé), le problème est NP-dur. Pour $k = 2$ ce problème se résout en temps polynomial (avec un algorithme de calcul de flot maximum). Par la suite, on va utiliser ce résultat en boîte noire. Si $S = \{s, t\}$, on parlera de "$s-t$ coupe" au lieu de "coupe *multi-way*".

Exercice 1. *Le but de cet exercice est de donner un algorithme d'approximation* $2(1 - \frac{1}{k})$ *pour le problème de la coupe* multi-way *minimale.*

1. *Trouver une coupe* multi-way *minimale pour* $S = \{s_1, s_2, s_3\}$ *dans le graphe*



   où $\epsilon > 0$

2. *Considérer l'algorithme suivant:*

   (a) *Pour* $i \in \{1, \ldots, k\}$, *trouver une coupe minimale* $C_i \in E$ *qui sépare* s_i *de* $S \setminus \{s_i\}$.

   (b) *Se débarrasser de* $C_n = argmax_{C_i}(w(C_i))$, *considérer* $C = \bigcup_{i \neq n} C_i$.

   Montrer que C *est une coupe* multi-way *pour* $S = \{s_1, \ldots, s_k\}$.

3. *Expliquer comment exécuter la première étape en temps polynomial (Indication: on identifie les noeuds de* $S \setminus \{s_i\}$*).*

4. *Supposer que* A *est une coupe* multi-way *optimale. On peut voir* A *comme une union de* k *coupes de la manière suivante:*

   Le graphe $G' = (V, E \setminus A)$ *contient* k *composantes connexes, chacune contenant un noeud* s_i *(expliquer pourquoi* G' *ne peut pas contenir plus que* k *composantes).*

   Noter avec A_i *la coupe qui sépare* s_i *des autres noeuds dans* $S \setminus \{s_i\}$. *Montrer que*

$$\sum_{i=1}^{k} w(A_i) = 2w(A)$$

5. *Montrer que* $w(C_i) \leq w(A_i)$.
6. *Expliquer pourquoi* $w(C_n) \geq \frac{1}{k} \sum_{i=1}^{k} w(C_i)$.
7. *Conclure que l'algorithme fournit une garantie de performance* $2(1 - \frac{1}{k})$.
8. *Donner un exemple de graphe qui montre que la garantie de performance* $2(1-\frac{1}{k})$ *qu'on a trouvée est optimale pour cet algorithme. (Indication: c'est trivial)*

2 Gomory-Hu Trees

Un ensemble d'arêtes $M \in E$ s'appelle une k-coupe si $G' = (V, E \setminus M)$ contient k composantes connexes. Le problème de la k-coupe optimale est de trouver une k-coupe M avec $w(M)$ minimal.

Pour k fixé, le problème se résout en temps polynomial. Si k fait partie de l'entrée, le problème est NP-dur.

Notons avec $T = (V, E')$ un arbre (avec le même ensemble de noeuds que G; E' n'est pas nécessairement contenu dans E) et $w' : E' \to \mathbb{R}^+$ une fonction.

Si $e \in E'$, le graphe $T' = (V, E' - \{e\})$ contient deux composantes connexes $T_1 = (V_1, E_1)$ et $T_2 = (V_2, E_2)$ (t.q. $V = V_1 \cup V_2$ et $E_1 \cup E_2 \cup \{e\} = E'$).

Definition 1. *La coupe* $M(e)$ *définie dans* G *par la partition* (V_1, V_2) *est la coupe associée à* e *dans* G.

Definition 2. *L'arbre* T *est un arbre Gomory-Hu pour* G *ssi:*

- $\forall u, v \in V$, *si* M *est une* $u-v$ *coupe optimale dans* G *et* M' *est une* $u-v$ *coupe optimale dans* T, *alors* $w(M) = w'(M')$.
- $\forall e \in E'$, $w(M(e)) = w'(e)$

Exercice 2. *Le but de cet exercice est de trouver un algorithme qui calcule un arbre Gomory-Hu pour le graphe* G.

1. *Soit* $f(u, v)$ *le poids minimal d'une* $u - v$ *coupe dans* G. *Montrer que si* $f(u, v) \leq f(u, w) \leq f(v, w)$, *alors* $f(u, v) = f(u, w)$.
2. *Soit* $K = (V, V^2 - \{(i, i) \mid i \in \{1, \ldots, n\}\})$ *le graphe complet ou on associe à l'arête* (u, v) *le poids* $f(u, v)$. *Montrer que tout cycle dans* K *admet deux arêtes de même poids.*
3. *Montrer que, parmi les* $\binom{n}{2}$ *valeurs* $f(u, v)$, *il y a au plus* $n - 1$ *valeurs distinctes.*
4. *Montrer que,* $\forall u, v, w \in V$

$$f(u, v) \geq min\{f(u, w), f(w, v)\}$$

5. *Montrer que,* $\forall u, v, w_1, \ldots, w_r \in V$

$$f(u, v) \geq \min\{f(u, w_1), f(w_1, w_2), \ldots, f(w_r, v)\}$$

6. *Soit* T *un arbre couvrant de* K *de poids maximal. Montrer que* T *satisfait la première condition d'un arbre Gomory-Hu. (Indication: considérer le chemin unique de* u *à* v *dans* T*).*
7. *Soit* (V_1, V_2) *une* $s-t$ *coupe optimale t.q.* $s \in V_1$. *Soient* $x, y \in V_1$. *Considérer le graphe* G' *obtenu en fusionnant tous les noeuds de* V_2 *en un noeud* v. *Le poids d'une arête* (a, v) *dans le nouveau graphe est* $\sum_{b \in V_2} w(a, b)$. *Montrer qu'une* $x - y$ *coupe optimale de* G' *définit une* $x - y$ *coupe optimale de* G.

8. *L'algorithme maintient une partition $(S_1, \ldots, S_t)$ de V et un arbre T dont les noeuds sont $S_1, \ldots, S_t$. Les poids des arêtes de T sont données par w'.*

   L'arbre T satisfait l'invariant suivant:

   Invariant 1. *$\forall (S_i, S_j) \in T$, il existe $a \in S_i$ et $b \in S_j$ t.q. $w'(S_i, S_j) = f(a, b)$ et la coupe définie par l'arête (S_i, S_j) est une $a - b$ coupe optimale de G.*

   L'algorithme commence avec la partition triviale V. A chaque étape, l'algorithme choisit une partition S_i t.q. $|S_i| \geq 2$ et raffine la partition de la manière suivante:

   - *on choisit deux noeuds $x, y \in S_i$*
   - *on construit G' de la manière suivante:*
     - *G' contient tous les noeuds de S_i*
     - *on choisit S_i la racine de T; pour chaque sous-arbre S_j de S_i on fusionne tous les noeuds de S_j.*
     - *les arêtes sont pondérées de la façon habituelle*

     et on calcule (A, B) une $x - y$ coupe optimale de taille w_{xy} dans G'
   - *On sépare S_i en $S_i^x = S_i \cap A$ et $S_i^y = S_i \cap B$ et on ajoute une arête $w'(S_i^x, S_i^y) = w_{xy}$*
   - *Supposer sans perte de généralité qu'un sous-arbre $S_j \in A$. Alors on ajoute une arête entre S_i^x et S_j du même poids que l'arête (S_i, S_j) de l'étape précédente.*

   Montrer l'invariant.

9. *Conclure que l'algorithme termine en donnant un arbre Gomory-Hu.*