TD 10-11 – Approximation

Exercice 1. *Problème du k-centre*

On rappelle quelques définitions :

– Dans un graphe $G = (V, E)$, un *ensemble indépendant* est un sous-ensemble de sommets V' non reliés par des arêtes (si $u \in V'$ et $v \in V'$, alors $(u, v) \notin E$).

– Dans un graphe $G = (V, E)$, un *ensemble dominant* est un sous-ensemble de sommets V' tel que tout sommet de $V \setminus V'$ est adjacent à un sommet de V'. On note $dom(G)$ le cardinal minimal d'un ensemble dominant.

1. Montrer que trouver un ensemble dominant de cardinal minimal $dom(G)$ est NP-difficile.

Soit $G = (V, E)$ un graphe non-orienté, complet, dont les arêtes sont pondérées par une fonction de poids w qui vérifie l'inégalité triangulaire : $w(u, v) \leq w(u, w) + w(w, v)$ pour tout triplet de sommets (u, v, w). Soit aussi un entier $k \geq 1$.

Pour tout $S \subset V$ et tout $v \in V \setminus S$, on définit $connect(v, S)$ comme le poids minimal d'une arête reliant v à un sommet de S : $connect(v, S) = \min_{s \in S}\{w(v, s)\}$. Le problème est de trouver un k-centre, c'est à dire un sous-ensemble S de cardinal k et tel que $center(S) = \max_{v \in V \setminus S}\{connect(v, S)\}$ soit minimal.

2. À quoi peut bien servir de déterminer un k-centre (donner un exemple d'application) ?

3. Montrer que trouver un k-centre est NP-difficile.

On va chercher une 2-approximation, i.e. un S de cardinal k tel que $center(S) \leq 2 \cdot OPT$, où $OPT = \min_{S \subset V, |S|=k}\{center(S)\}$.

On ordonne les arêtes de E par poids croissant : $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_m)$, où $m = |E|$. On pose $G_i = (V, E_i)$ où $E_i = \{e_1, e_2, \ldots, e_i\}$ est l'ensemble des i premières arêtes.

4. Montrer que résoudre le problème du k-centre revient à trouver le plus petit indice i tel que G_i a un ensemble dominant de cardinal au plus k.

Une dernière définition : le carré d'un graphe $G = (V, E)$, noté $G^{(2)} = (V, E^{(2)})$, contient les chemins de longueur au plus deux : $(u, v) \in E^{(2)}$ si $(u, v) \in E$ ou s'il existe $w \in V$ tel que $(u, w) \in E$ et $(w, v) \in E$.

5. Étant donné un graphe H, soit I un ensemble indépendant du graphe carré $H^{(2)}$. Montrer que $|I| \leq dom(H)$.

6. L'algorithme d'approximation du k-centre est le suivant :

```
début
    Construire G_1^(2), G_2^(2), ..., G_m^(2);
    Trouver de manière gloutonne un ensemble indépendant inextensible (auquel on ne peut pas
    rajouter des sommets) M_i dans chaque graphe G_i^(2);
    Trouver le plus petit indice i tel que |M_i| ≤ k, soit j cet indice;
    retourner M_j;
```

(a) Montrer que $w(e_j) \leq OPT$.

(b) Montrer que l'algorithme est bien une 2-approximation.

7. Montrer que la borne 2 est stricte : donner un exemple de graphe où l'algorithme réalise effectivement une 2-approximation.

8. Montrer que si $P \neq NP$, il n'existe pas de $(2 - \varepsilon)$-approximation au problème du k-centre, pour tout $\varepsilon > 0$.

Exercice 2. *Sac à dos*

On s'intéresse au problème du sac-à-dos :
*I*nstance : Un ensemble fini X d'objets ; pour chaque objet $x_i \in X$, une valeur $p_i \in \mathbb{N}$ et une taille $a_i \in \mathbb{N}$. Une capacité $B \in \mathbb{N}$.
*S*olution : Un sous-ensemble Y de X tel que $\sum_{x_i \in Y} a_i \leq B$
*M*esure : La valeur totale des objets choisis, i.e. $\sum_{x_i \in Y} p_i$.
On notera $m^*(x)$ la mesure optimale (maximale).

1. Formuler le problème de décision associé et montrer qu'il est $\mathcal{NP}$-complet.
2. Mais alors, comment a-t-on pu résoudre ce problème en cours par programmation dynamique ?

Glouton Dans l'algorithme glouton, on trie les éléments par rapports p_i/a_i décroissants, et on choisit chaque élément examiné dans cet ordre s'il y a la place pour lui. Soit $m_g(x)$ la mesure de l'algorithme glouton.

3. Soit K un entier arbitrairement grand. Construire une instance x telle que $m^*(x)/m_g(x) > K$.
4. Soit p_{max} la valeur maximale d'un objet. Montrer que $m^*(x)/\max\{m_g(x), p_{max}\} < 2$.
   Indication : soit j l'indice du premier élément que l'algorithme glouton ne prend pas ; montrer que $m^*(x) \leq \sum_{i=1}^{j} p_i$.

Programmation dynamique revisited

5. Pour $1 \leq k \leq n$ et $0 \leq p \leq \sum_{i=1}^{n} p_i$, on cherche, parmi les sous-ensembles de $\{x_1, x_2, \ldots, x_k\}$ de valeur totale égale à p et de taille majorée par B, un qui soit de taille minimale. On note $M^*(k,p)$ une solution optimale de ce problème et $S^*(k,p)$ la taille correspondante. Expliquer comment calculer les $M^*(k,p)$ par programmation dynamique. Quelle est la complexité de la résolution du problème du sac-à-dos ?
6. Pour tout rationnel $r > 1$, on considère le schéma d'approximation suivant : soit p_{max} la valeur maximale d'un objet et $t = \lfloor \log(\frac{r-1}{r}\frac{p_{max}}{n}) \rfloor$. On résout par programmation dynamique comme plus haut une instance modifiée du problème original : les tailles des n objets sont toujours a_i mais les valeurs sont $p'_i = \lfloor \frac{p_i}{2^t} \rfloor$. Soit $m_{AS}(x,r)$ la mesure de la solution ainsi obtenue.
   (a) Montrer que la complexité du schéma d'approximation est $O(\frac{r}{r-1}n^3)$.
   (b) Montrer que $m^*(x)/m_{AS}(x,r) \leq r$.
   Indication : montrer que $\frac{m^*(x) - m_{AS}(x,r)}{m^*(x)} \leq \frac{n2^t}{p_{max}}$.

Exercice 3. *SetCover*

Le problème de décision Set-Cover est défini comme suit :
Entrée : Un ensemble X contenant n éléments, ainsi que m sous-ensembles $S_1, \ldots, S_m$ de X, et un entier $k \leq n$.
Sortie : Peut-on trouver au plus k ensembles parmi les k S_i de façon à couvrir tous les éléments de X ? Plus formellement, existe-t-il $I \subseteq \{1, \ldots, m\}$ tel que $|I| \leq k$ et $\cup_{i \in I} S_i = X$?
On admettra que Set-Cover est un problème $\mathcal{NP}$-complet[1](vous pouvez le chercher, ce n'est pas si compliqué !). Le problème d'optimisation associé est de trouver la plus petite couverture.
Pour cela, considérons l'algorithme glouton suivant :
– Prendre un ensemble S_j qui couvre le plus grand nombre d'élément de X.
– Remplacer X par $X \setminus S_j$ et tous les S_i par $S_i \setminus S_j$.
– Recommencer tant que X n'est pas vide.

1. Est-ce que l'algorithme glouton est optimal ?
2. Prouver que si une solution optimale contient k sous-ensembles, alors l'algorithme glouton prend au plus $\mathcal{O}(k \ln(n))$ sous-ensembles.
3. Est-ce que l'algorithme glouton est une λ-approximation de Set-Cover ? Si oui, quelle est la valeur de λ ?

1. M.R. Garey et D.S. Johnson, *Computers and Intractability, a Guide to the Theory of NP-completeness*. W.H. Freeman and Company, 1979