TD 5 : Algorithmes d'approximation

Exercice 1. *Coupe maximale*

Soit $G = (S,A)$ un graphe. Une *coupe* de G est une partition des sommets $S = X \sqcup Y$ en deux sous-ensembles disjoints. La *taille* d'une coupe $X \sqcup Y$ est le nombre d'arêtes dont une extrémité est dans X et l'autre dans Y. On cherche à calculer une coupe $X \sqcup Y$ de taille maximale.

On utilise l'algorithme probabiliste simpliste suivant : chaque sommet $s \in S$ est affecté indépendamment à X avec probabilité $\frac{1}{2}$ et à Y avec la même probabilité.

1. Soit $a = \{u, v\}$ une arête de G. Calculer la probabilité que a soit *coupée*, c'est-à-dire qu'une de ses extrémités appartienne à X et l'autre à Y.
2. Quelle est l'espérance de la taille de la coupe renvoyée par l'algorithme probabiliste ? *Utiliser la linéarité de l'espérance.*
3. En déduire que l'algorithme probabiliste est une 2-approximation espérée pour le problème de la coupe maximale.

Exercice 2. *Partition*

Étant donné un ensemble de n entiers positifs $A = \{a_0, \dots, a_{n-1}\}$, on cherche à faire une partition de A en deux sous-ensembles X et Y de manière assez équilibrée. Pour un ensemble $S \subseteq A$, on note $\Sigma_S = \sum_{a_i \in S} a_i$ la somme des éléments de S. Pour équilibrer la partition, on cherche à minimiser $\max(\Sigma_X, \Sigma_Y)$. On propose l'algorithme glouton suivant.

```
X ← ∅, Σ_X ← 0
Y ← ∅, Σ_Y ← 0
Pour i = 0 à n − 1 :
  Si Σ_X < Σ_Y :
     X ← X ∪ {a_i}, Σ_X ← Σ_X + a_i
  Sinon :
     Y ← Y ∪ {a_i}, Σ_Y ← Σ_Y + a_i
Renvoyer (X, Y)
```

1. **i.** Appliquer l'algorithme sur l'entrée $A = \{4, 2, 3, 2, 7\}$. Fournit-il une solution optimale ?
   ii. Quelle est la complexité de l'algorithme ?

Pour une entrée A, soit $(X^\star, Y^\star)$ une solution optimale et $\textsc{opt} = \max(\Sigma_{X^\star}, \Sigma_{Y^\star})$. Soit (X, Y) la solution renvoyée par l'algorithme glouton, et on suppose sans perte de généralité $\Sigma_X \geq \Sigma_Y$.

2. **i.** Montrer que pour tout i, $a_i \leq \textsc{opt}$.
   ii. Montrer que $\Sigma_A \leq 2\textsc{opt}$.
3. On considère le dernier élément a_k ajouté par l'algorithme glouton à X.
   i. Montrer que $\Sigma_X - a_k \leq \frac{1}{2}(\Sigma_A - a_k) \leq \textsc{opt} - \frac{1}{2}a_k$.
   ii. En déduire que l'algorithme glouton est une $\frac{3}{2}$-approximation pour le problème.
   iii. Construire un exemple pour lequel l'algorithme glouton fournit une solution égale exactement à $\frac{3}{2}\textsc{opt}$.

On modifie très légèrement l'algorithme, en commençant par trier les a_i par ordre décroissant. On suppose donc dans la suite que $a_0 \geq a_1 \geq \cdots \geq a_{n-1}$. On note toujours (X, Y) la solution renvoyée par l'algorithme glouton et on suppose toujours $\Sigma_X \geq \Sigma_Y$.

4. **i.** Montrer que sur l'entrée $\{10, 10, 9, 9, 2\}$, l'algorithme glouton avec tri n'est pas optimal.
   ii. Montrer que si X ne contient qu'un seul élément de A, alors la solution renvoyée est optimale.
   iii. On suppose que X contient au moins 2 éléments. Montrer que le dernier élément a_k ajouté par l'algorithme glouton à X vérifie $a_k \leq \frac{2}{3}\textsc{opt}$.
   iv. En déduire que l'algorithme glouton avec tri décroissant est une $\frac{4}{3}$-approximation pour le problème.

5. Trouver une entrée pour laquelle la solution renvoyée par l'algorithme glouton avec tri vérifie $\max(\Sigma_X, \Sigma_Y) = \frac{7}{6}\text{OPT}$. *Remarque. On peut en fait montrer (mais c'est difficile) que l'algorithme glouton avec tri renvoie toujours une solution* $\leq \frac{7}{6}\text{OPT}$.

Exercice 3. *Couverture gloutonne par sommets*

On propose l'algorithme glouton suivant pour calculer une couverture par sommets d'un graphe $G = (S, A)$: tant qu'il existe une arête non couverte, choisir le sommet de degré maximal et supprimer les arêtes qu'il couvre.

1. Écrire formellement l'algorithme et analyser sa complexité, en supposant que G est représenté par matrice d'adjacence.

On note $G_0 = G$ le graphe initial, et G_i le graphe obtenu après la $i^{\text{ème}}$ itération de la boucle. On note m_i le nombre d'arêtes dans le graphe G_i, et $\deg_i(v)$ le degré d'un sommet v dans le graphe G_i. Enfin, on note $C^\star$ une couverture par sommets optimale de G, et OPT son nombre de sommets.

2. Montrer que $\sum_{v\in C^\star} \deg_i(v) \geq m_i$ pour tout i.
3. On note d_i le degré maximal d'un sommet dans G_i ($d_i = \max_v \deg_i(v)$). Montrer que $d_i \geq m_i/\text{OPT}$.
4. En déduire que $\sum_{i=0}^{\text{OPT}-1} d_i \geq m - \sum_{i=0}^{\text{OPT}-1} d_i$.
5. En déduire qu'après OPT itérations, le nombre d'arêtes de G a été divisé par au moins 2.
6. En déduire que l'algorithme glouton est une $O(\log m)$ approximation pour le problème de la couverture par sommets.

Exercice 4. *Sac-à-dos*

Le problème du sac-à-dos est le suivant : étant donné un ensemble O de n objets décrits par un couple (t_i, v_i) où t_i est la taille de l'objet i et v_i sa valeur, ainsi qu'une taille de sac-à-dos T, on cherche un sous-ensemble des objets dont la taille totale est $\leq T$ et de valeur la plus grande possible. Pour un ensemble d'indice $I \subset \{0, \dots, n-1\}$, on note $T_I = \sum_{i\in I} t_i$ et $V_I = \sum_{i\in I} v_i$. Formellement, on cherche un ensemble I tel que $T_I \leq T$ et qui maximise V_I. *On suppose que $t_i \leq T$ pour tout i, c'est-à-dire que chaque objet rentre, individuellement, dans le sac-à-dos. Si ce n'était pas le cas, on pourrait commencer par éliminer tous les objets trop gros.*

On considère l'algorithme glouton suivant : pour $i = 0$ à $n-1$, si l'objet (t_i, v_i) rentre dans le sac-à-dos, on l'ajoute et on met à jour la taille libre restante du sac-à-dos.

1. **i.** Écrire l'algorithme et analyser sa complexité.
   ii. Donner un exemple où cet algorithme renvoie une solution arbitrairement mauvaise.

Pour améliorer l'algorithme, on ne parcourt plus les objets dans un ordre quelconque, mais en les triant selon différents critères.

2. **i.** On trie les objets par valeur décroissante. Montrer que l'algorithme glouton peut toujours être arbitrairement mauvais.
   ii. On trie les objets par ratio v_i/t_i décroissant. Montrer que l'algorithme peut encore une fois être très mauvais.

Les deux critères de tri précédents ne fonctionnent pas. Cependant, ils fonctionnent si on les combine : on applique donc l'algorithme glouton avec le tri par valeur décroissante, puis avec le tri par ratio décroissant, et on garde la meilleure des deux solutions. Soit I_1 l'ensemble d'indices choisi par le premier tri (valeur décroissante), I_2 l'ensemble d'indices choisi par le second (ratio décroissant), et I_{OPT} un ensemble d'indice optimal. On note $V_1 = V_{I_1}$, $V_2 = V_{I_2}$ et $\text{OPT} = V_{I_{\text{OPT}}}$ la valeur optimale.

On considère que les objets sont triés par ratio décroissant, c'est-à-dire $v_0/t_0 \geq v_1/t_1 \geq \cdots \geq v_{n-1}/t_{n-1}$. Soit j le plus petit indice non sélectionné par l'algorithme glouton avec tri par ratio décroissant.

3. On note $I_{\leq j}$ les indices de la solution optimale qui sont $\leq j$ et $I_{>j}$ les autres indices. De la même façon, on note J les indices de $\{0, \dots, j\}$ qui n'appartiennent pas I_{OPT}. Ainsi, $I_{\text{OPT}} = I_{\leq j} \sqcup I_{>j}$ et $\{0, \dots, j\} = I_{\leq j} \sqcup J$.

   i. Montrer que $\sum_{i\in J} t_i > \sum_{i\in I_{>j}} t_i$. *Indication. Montrer que $\sum_{i=0}^{j} t_i > T \geq \sum_{i\in I_{\text{OPT}}} t_i$ et conclure.*
   ii. En déduire que $\sum_{i\in J} v_i > \sum_{i\in I_{>j}} v_i$.
   iii. En déduire que $\sum_{i=0}^{j} v_i > \text{OPT}$.

4. **i.** Montrer que pour tout i, $v_i \leq V_1$.

ii. Montrer que $V_2 \geq \sum_{i=0}^{j-1} v_i$.
iii. En déduire que $V_1 + V_2 \geq \sum_{i=0}^{j} v_i$.
iv. En déduire que $\max(V_1, V_2) \geq \frac{1}{2}\text{OPT}$.

Exercice 5. *k-centre*

Étant donnés un ensemble de n points $P = \{p_0, \ldots, p_{n-1}\}$ dans le plan et un entier k, on cherche à déterminer un ensemble de k cercles qui couvrent les n points, en minimisant le rayon du plus grand cercle. Formellement, on cherche à déterminer un ensemble de k centres $C = \{c_0, \ldots, c_{k-1}\}$ qui minimise la quantité

$$R(C) = \max_{0\leq i<n} \min_{0\leq j<k} d(p_i, c_j)$$

où $d(p_i, c_j)$ est la distance euclidienne entre les points p_i et c_j. On appelle $R(C)$ le *rayon maximal* de l'ensemble C.

On utilise l'algorithme glouton suivant : on choisit un des points de P, arbitrairement, comme étant le premier centre c_0 ; quand i centres c_0, …, c_{j-1} ont été choisis, on cherche le point p_i qui se trouve le plus loin possible d'un des centres c_0, …, c_{j-1} et on le choisit comme $(j+1)^{\text{ème}}$ centre.

1. Écrire formellement l'algorithme et analyser sa complexité. *Pour écrire un algorithme efficace, retenir pour chaque point p_i sa distance d_i au centre déjà fixé le plus proche.*

On considère une solution optimale $C^\star = \{c_0^\star, \ldots, c_{k-1}^\star\}$, de rayon maximal $R^\star = R(C^\star)$ minimal. Pour chaque centre $c_j^\star$, on définit le *cluster* P_j des points dont $c_j^\star$ est le centre le plus proche. Formellement, $P_j = \{p_i : \forall \ell, d(p_i, c_j^\star) \leq d(p_i, c_\ell^\star)\}$.

2. Montrer que deux points d'un même cluster P_j sont à distance $\leq 2R^\star$. *Utiliser l'inégalité triangulaire.*

Soit $C = \{c_0, \ldots, c_{k-1}\}$ la solution renvoyée par l'algorithme glouton.

3. On suppose qu'il existe un centre c_j dans chaque cluster. Montrer que sous cette hypothèse, $R(C) \leq 2R^\star$.
4. On suppose maintenant qu'il existe un cluster P_j contenant deux centres de C.

   i. Montrer que lorsque l'algorithme glouton choisit le deuxième centre dans P_j, tout point est à distance $\leq 2R^\star$ d'un centre déjà choisi.
   ii. En déduire que $R(C) \leq 2R^\star$.

5. Quel est le facteur d'approximation de l'algorithme glouton ?