TD 9. Algorithmes d'approximation

Exercice 1. *Partition*

Notation. Pour un ensemble S d'entiers, on note $\Sigma_S = \sum_{s\in S} s$.

Étant donné un ensemble de n entiers positifs $A = \{a_0, \dots, a_{n-1}\}$, on cherche une partition $A = X \sqcup Y$ équilibrée, c'est-à-dire telle que $\Sigma_X \simeq \Sigma_Y$. Plus précisément, on cherche à minimiser $\max(\Sigma_X, \Sigma_Y)$. On propose l'algorithme glouton suivant.

PARTITION(A) :

```
(X, Y, Σ_X, Σ_Y) ← (∅, ∅, 0, 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.** PARTITION fournit-il une solution optimale sur l'entrée $A = \{4, 2, 3, 2, 7\}$?
ii. Quelle est sa complexité ?

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

2. **i.** Montrer que pour tout i, $\text{OPT} \geq a_i$.
ii. Montrer que $\text{OPT} \geq \frac{1}{2}\Sigma_A$.

3. On considère le dernier élément a_k ajouté par PARTITION à X.

i. Montrer que $\Sigma_X - a_k \leq \frac{1}{2}(\Sigma_A - a_k) \leq \text{OPT} - \frac{1}{2}a_k$.
ii. En déduire que PARTITION est une $\frac{3}{2}$-approximation pour le problème.
iii. Construire un exemple pour lequel PARTITION fournit une solution égale exactement à $\frac{3}{2}\text{OPT}$.

4. *(bonus)* On modifie très légèrement PARTITION en triant les a_i en ordre décroissant : $a_0 \geq a_1 \geq \cdots \geq a_{n-1}$. On garde les mêmes notations que précédemment.

i. Montrer que sur l'entrée $\{10, 10, 9, 9, 2\}$, l'algorithme n'est pas optimal.
ii. Montrer que si $\#X = 1$, alors la solution renvoyée est optimale.
iii. On suppose que $\#X \geq 2$. Montrer que le dernier élément a_k ajouté par PARTITION à X vérifie $a_k \leq \frac{2}{3}\text{OPT}$.
iv. En déduire que PARTITION avec tri est une $\frac{4}{3}$-approximation.
v. Construire un exemple pour lequel PARTITION avec tri fournir une solution exactement égale à $\frac{7}{6}\text{OPT}$. [1]

1. *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 2. *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 3. MAXSAT

On considère des *formules sous forme normale conjonctive* (formule CNF), comme par ex_emple $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_3) \wedge (x_2 \vee x_3)$: c'est une *conjonction de clauses*, chaque clause est une *disjonction de littéraux* et chaque littéral est soit une vairable booléenne soit sa négation. Le problème SAT consiste, étant donné une formule CNF, à décider s'il existe une affectation qui satisfait la formule.

Dans cet exercice, on s'intéresse à une variante du problème : MAXSAT. Étant donné la formule CNF, il s'agit de trouver l'affectation qui satisfait *le plus possible de clauses*.

1. Justifier que si on sait résoudre de manière exacte MAXSAT, alors on peut résoudre SAT.
2. Vérifier que dans la formule de l'exemple, au plus 3 clauses sur 4 peuvent être satisfaites simultanément.
3. Quel algorithme (déjà vu) peut-on modifier pour résoudre MAXSAT ? Quelle est sa complexité ?
4. On propose l'algorithme suivant : on renvoie une affectation des variables choisie uniformément, c'est-à-dire qu'on choisit, pour chaque variable, la valeur VRAI ou FAUX de manière aléatoire uniforme, indépendamment des autres variables. *On suppose que chaque variable apparaît au plus une fois dans chaque clause.*
   - **i.** Quelle est la complexité de cet algorithme ?
   - **ii.** Quelle est la valeur OPT pour ce problème ?
   - **iii.** Soit C une clause de taille k. Montrer que la probabilité que C soit satisfaite est $1 - 1/2^k$.
   - **iv.** En déduire que l'espérance du nombre de clauses satisfaites est $\geq m/2$ où m est le nombre total de clauses.

Exercice 4. *Sac-à-dos*

On rappelle le problème du sac-à-dos : étant donné un n objets de taille t_i et de valeur v_i, $0 \leq i < n$, et une taille T, on cherche un sous-ensemble d'objets dont la somme des tailles est $\leq T$ et qui dont la somme de valeur est maximale. Pour $I \subset \{0, \ldots, n-1\}$, on note $T_I = \sum_{i \in I} t_i$ et $V_I = \sum_{i \in I} v_i$: on cherche I tel que $T_I \leq T$ et qui maximise V_I. *On suppose que $t_i \leq T$ pour tout i.*

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.** Donner un exemple où cet algorithme renvoie une solution arbitrairement mauvaise.
   ii. On commence par trier les objets par valeur décroissante. Montrer que l'algorithme peut toujours être arbitrairement mauvais.
   iii. On trie maintenant les objets par ratio v_i/t_i décroissant. Montrer que l'algorithme peut encore une fois être très mauvais.

On veut montrer que si on *combine* les deux critères de tri proposés, on obtient un algorithme d'approximation. On applique l'algorithme glouton avec le tri par valeur décroissante, ce qui donne un ensemble d'indices I_1, puis avec le tri par ratio décroissant, ce qui donne un second ensemble d'indices I_2. On garde la meilleure des deux solutions, c'est-à-dire qu'on renvoie un sac-à-dos de valeur $\max(V_{I_1}, V_{I_2})$.

Dans la suite, on note I_{OPT} un ensemble d'indices optimal et $\text{OPT} = V_{I_{\text{OPT}}}$. On suppose que les objets sont triés par ratio décroissant : $v_0/t_0 \geq v_1/t_1 \geq \cdots \geq v_{n-1}/t_{n-1}$.

2. Soit j le plus petit indice non sélectionné par l'algorithme glouton avec tri par ratio décroissant. On note $I_{\leq j} = \{i \leq j : i \in I_{\text{OPT}}\}$, $I_{>j} = \{i > j : i \in I_{\text{OPT}}\}$ et $J = \{i \leq j : i \notin I_{\text{OPT}}\}$.
   i. Montrer que $T_J > T_{I_{>j}}$. *Indication. Montrer que $T_{\{0,\ldots,j\}} > T \geq T_{I_{\text{OPT}}}$ et conclure.*
   ii. En déduire que $\sum_{i \in J} v_i > \sum_{i \in I_{>j}} v_i$.
   iii. En déduire que $\sum_{i=0}^{j} v_i > \text{OPT}$.
3. **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 ?