

Algorithme d'approximation

Le problème du k -centre

Dans cette partie, nous allons nous concentrer sur le problème du k -centre : étant donné un ensemble de villes dont les distances sont spécifiées, choisir k villes afin d'installer des entrepôts de façon à minimiser la distance maximale d'une ville à l'entrepôt le plus proche. Un tel ensemble de k villes est appelé k -centre.

Nous allons nous focaliser sur le problème de k -CENTRE MÉTRIQUE. Ici, la fonction de poids que l'on notera w respecte l'inégalité triangulaire, c'est-à-dire pour tout triplet de villes u, v , et z , on a $w(u, v) \leq w(u, z) + w(z, v)$.

Exercice 1 Approximation du problème

Question 1.1 Le carré d'un graphe G , noté G^2 , est défini comme le graphe qui contient une arête (u, v) pour chaque couple de sommets u et v reliés par un chemin de longueur au plus 2 dans G .

Montrer que si le sous-ensemble de sommets I est un stable de G^2 et alors pour tout ensemble dominant D de G , on a $|I| \leq |D|$.

Correction

Soit D^* un dominant minimum de G . Par définition, G possède $|D^*|$ étoiles couvrant tous les sommets de G . Comme les sommets d'une étoile de G forment une clique dans G^2 , G^2 se décompose en au plus $|D^*|$ cliques qui couvrent aussi tous ses sommets.

Par conséquent, tout stable I de G^2 ne pourra avoir au plus qu'un unique sommet dans chacune de ces cliques de G^2 .

$$\text{Donc } |I| \leq |D^*|.$$

□

Question 1.2 Dans la suite de l'exercice, nous supposons que les arêtes de K sont triées par poids croissant, c'est-à-dire $w(e_1) \leq w(e_2) \leq \dots \leq w(e_m)$ et nous notons $K_i = (V, E_i)$, où $E_i = \{e_1, e_2, \dots, e_i\}$.

Montrer que si G_i admet un dominant de taille au plus k alors K admet un k -centre de coût au plus $w(e_i)$

Correction

Si D est un dominant de taille au plus k dans G_i , alors tous les sommets de K sont à distance au plus $w(e_i)$

□

Question 1.3 Voici l'algorithme \mathcal{A} du k -centre :

1. Construire $(G_1)^2, (G_2)^2, \dots, (G_m)^2$.
2. Calculer un stable maximal, M_i , pour chaque $(G_i)^2$.
3. Calculer le plus petit indice i tel que $|M_i| \leq k$. Notons-le j .
4. Renvoyer M_j .

Donner la complexité de cet algorithme.

Correction

Construire la suite de graphes peut se faire en $O(mn)$ opérations.

Calculer un stable maximal peut se calculer en $O(n)$ opérations. Il suffit tout simplement parcourir tous les sommets et construire en parallèle un stable en ajoutant ou pas le noeud courant.

On calcule au pire des cas m stables. Cela nécessite $O(mn)$ opérations. □

Notons OPT le coût d'un k -centre optimal. Notons aussi par i^* le plus petit indice, $w(e_{i^*}) = OPT$ et tel que G_{i^*} contient un k -centre optimal.

- Montrer que si j est l'indice calculé par l'algorithme, alors $w(e_j) \leq OPT$.

Correction

Pour tout entier $i < j$, on a $|M_i| > k$ (sinon l'algorithme retournerait i). D'après la question précédente, en notant D_i^* un dominant optimal du graphe G_i , $|D_i^*| > k$, et donc que $i^* > i$. Par conséquent, $j \leq i^*$ et $w(e_j) \leq w(e_{i^*})$. □

- Montrer que l'algorithme \mathcal{A} est une 2-approximation.

Correction

Un stable maximal I d'un graphe est également un dominant : s'il existe un sommet v non dominé par I , alors $I \cup \{v\}$ serait un stable et ceci est en contradiction avec la propriété que I soit maximal.

Dans G_j^2 , les sommets de M_j sont des centres des étoiles qui couvrent tous les sommets. Or, l'inégalité triangulaire donne que toutes les arêtes de G_j utilisées par ces étoiles ont un coût à $2w(e_j)$.

Comme $w(e_j) \leq w(e_{i^*})$, on retourne un k -centre de coût $2w(e_j)$ et $2w(e_j) \leq 2w(e_{i^*})$ □

Question 1.4 Considérons l'instance suivante : le graphe complet de $n + 1$ sommets possède un sommet singulier u . Chaque arête incidente à u est de poids 1 et les autres sont de poids 2. Appliquer l'algorithme sur ce graphe avec $k = 1$ en considérant la pire solution. Qu'en déduisez-vous ?

Correction

Pour $k = 1$, la solution optimale est le centre de la roue et $OPT = 1$.

La solution retournée par l'algorithme est la suivante $j = n$. En effet G_n^2 forme une clique. De plus si un sommet (autre que u) est sélectionné pour construire un stable maximal, alors le coût de la solution sera de 2. □

Exercice 2 Problème SAC à DOS.

Le problème du SAC à DOS est un problème classique en informatique. Il modélise une situation analogue au remplissage d'un sac. Une personne veut remplir un sac à dos ne pouvant pas supporter plus d'un certain poids $C \in \mathbb{N}$, et elle dispose de n objets (On note l'ensemble des objets par $\mathcal{O} = \{1, \dots, n\}$). Chaque objet i a une valeur $v_i \in \mathbb{N} \setminus \{0\}$ et un poids $p_i \in \mathbb{N} \setminus \{0\}$. Le problème est de trouver un ensemble d'objets tels que

- tous les objets de cet ensemble puissent être mis dans le sac.
- la somme des valeurs de ces objets soit maximale.

Le problème d'optimisation correspond à trouver un sous-ensemble d'objets dont le poids total est inférieure à C et dont la valeur totale soit maximum. **Notons** $\mathbf{v} = \max\{v_i : i \in \mathcal{O}\}$.

Rappelons l'algorithme de la programmation dynamique. Notons $T[i, j]$ représentera la valeur maximale pour un sac à dos de capacité j à l'aide des i premiers objets.

Entrée : un ensemble d'objets $\mathcal{O} = \{1, \dots, n\}$. L'objet i a une valeur v_i et un poids p_i .

Sortie : un entier

1. Initialiser tous les éléments du tableau T à zéro.
2. Pour tout i allant de 1 à n faire
 - (a) Pour tout j allant de 1 à C faire

Si $j < v_i$, alors $T[i, j] = T[i, j]$
 sinon $T[i, j] = \max(T[i - 1, j], T[i - 1, j - p_i] + v_i)$

3. Retourner $T[n, C]$

Question 2.1 Donner la complexité de cet algorithme. Peut-on modifier l'algorithme de telle façon que sa complexité soit exprimée en fonction de v ?

Correction

Il suffit de remplacer
 par
 L'algorithme se réalise en $O(n^2v)$ opérations.

Pour tout j allant de 1 à C faire
Pour tout j allant de 1 à nv faire

□

Maintenant, considérons l'algorithme suivant :

1. Etant donné $\varepsilon > 0$, $K \leftarrow \frac{\varepsilon \cdot v}{n}$
2. Pour chaque objet i , $v'_i = \lfloor \frac{v_i}{K} \rfloor$
3. Lancer l'algorithme de programmation dynamique avec ces valeurs arrondies v' et trouver un ensemble S' dont la valeur (v') totale soit maximum.
4. Renvoyer S' .

Question 2.2 Donner la complexité de cet algorithme.

Correction

Le temps de calcul de l'algorithme est $O(n^2 \lceil \frac{v}{K} \rceil) = O(n^2 \lceil \frac{n}{\varepsilon} \rceil)$. C'est bien un polynôme en n et $1/\varepsilon$, □

Notons OPT le coût de la solution optimale.

Question 2.3 Montrer que $\sum_{i \in S'} v_i \geq (1 - \varepsilon) \cdot OPT$

Correction

Notons O un ensemble des objets dont la valeur totale soit maximum : $v(O) = OPT$. Pour tout objet a , on a bas, $K \cdot v'_a \leq v_a \leq K$ et $v_a - K \cdot v'_a \leq K$. Ainsi,

$$v(O) - K \cdot v'(O) \leq n \cdot K$$

L'étape correspondant à la programmation dynamique renvoie un ensemble de valeur au moins aussi performant que celui de O avec les valeurs arrondies.

$$v(S') \geq K \cdot v'(O) \geq v(O) - nK = v(O) - \varepsilon \cdot v$$

en observant que $K = \frac{\varepsilon \cdot v}{n}$. Comme $v(O) \geq v$,

$$v(S') \geq (1 - \varepsilon) \cdot v(O)$$

□

Question 2.4 Qu'en concluez-vous ?

Correction

$$OPT \geq v(S') \geq (1 - \varepsilon) \cdot OPT.$$

Le temps de calcul de l'algorithme est $O(n^2 \frac{v}{K}) = O(n^2 \frac{n}{\varepsilon})$. C'est bien un polynôme en n et $1/\varepsilon$, □

Exercice 3 Les graphes planaires

Exercice 4 Le problème du coloriage de graphe Un graphe est *planaire* si on peut le dessiner dans le plan sans que les arêtes ne se croisent.

Soit $G = (V, E)$ un graphe planaire non vide *connexe*. Notons $n = |V|$, $a = |E|$. Considérons un dessin de G dans le plan (sans que les arêtes ne se croisent). Soit f le nombre de régions du plan délimitées par les arêtes de G .

Question 4.1 Montrer que les graphes planaires respectent la relation d'Euler $n - a + f = 2$. (*Indication : raisonner par récurrence sur le nombre de faces*)

Correction

On procède par récurrence sur le nombre f de faces. Si $f = 1$, le graphe possède uniquement une unique face. Par conséquent, le graphe connexe ne possède aucun cycle et est un arbre. Ainsi, $n - a + f = n - (n - 1) + 1 = 2$ et la formule est vérifiée.

Supposons que la formule d'Euler satisfaite pour les valeurs inférieur à f .

Soit e une arête d'un cycle du graphe. Par définition, l'arête appartient sépare deux faces A et B . En supprimant cette arête, le nouveau graphe obtenu possède le même nombre de sommets (n), $a - 1$ arêtes. De plus ce graphe a $f - 1$ faces puisque A et B forment une seule face de ce nouveau graphe. En appliquant l'hypothèse de récurrence, on a

$$n - (a - 1) + f - 1 = 2$$

En simplifiant on obtient $n - a + f = 2$ et le graphe respecte bien la formule d'Euler. Donc tout graphe planaire satisfait la formule d'Euler. □

Question 4.2 On suppose $n \geq 3$. Montrer que $3f \leq 2a$.

Correction

Soit \mathcal{F} l'ensemble de faces. Notons $nb(F)$ le nombre d'arêtes qui délimitent la face F . Chaque face F est délimitée par au moins 3 arêtes.

$$nb(F) \geq 3$$

Donc $\sum_{i \in \mathcal{F}} nb(F) \geq 3 \cdot f$

Chaque arête est une frontière de deux faces et donc on peut en déduire que

$$\sum_{i \in \mathcal{F}} nb(F) = 2a$$

Donc en combinant les deux équations, on obtient $3f \leq 2a$. □

Question 4.3 Montrer qu'il existe un sommet de degré au plus 5 dans le graphe planaire G .

Correction

Si G possède des sommets de degré 1, alors il existe bien un sommet de degré au plus 5 dans G .

Supposons que G ne possède pas des sommets de degré 1. Raisonnons par l'absurde et supposons que pour tout sommet v de G , $d(v) \geq 6$. On a $6n \leq 2a$ car la somme de tous les sommets est égale à 2 fois le nombre de arêtes.

En appliquant la formule d'Euler ($n - a + f = 2$), on a $f = 2 + a - n$.

$$\begin{aligned} 3f &\leq 2a \\ 3(2 + a - n) &\leq 2a \\ a &\leq 3n - 6 \\ 2a &\leq 6n - 12 \end{aligned}$$

Ce qui est en contradiction avec $6n \leq 2a$. Donc il existe un sommet de degré au plus 5 dans G . □

Question 4.4 Concevoir un algorithme de 6-coloration des graphes planaires en temps polynomial.

Correction

Notons $\mathcal{D}(G)$ l'ensemble des sommets de G tel que leur degrés sont inférieurs ou égale à 5. Nous allons construire une suite de graphes G_1, \dots, G_n de la façon suivante

1. $G_1 \leftarrow G$
2. pour i allant de 2 à n faire
 - (a) calculer $\mathcal{D}(G_i)$;
 - (b) extraire un sommet v_i tel que $v_i \in \mathcal{D}(G_i)$;
 - (c) construire $G_{i+1} = (V_{i+1}, E_{i+1})$ tel que $V_{i+1} = V_i \setminus \{v_i\}$, et $E_{i+1} = E_i \setminus \{e : e \text{ est adjacent à } v_i \text{ dans } G_i\}$.

Tout d'abord remarquons que ces graphes sont des graphes planaires (puisqu'on enlève simplement un sommet entre deux graphes G_i et G_{i-1} .)

A partir de cette suite, nous pouvons construire une coloration $c : V \rightarrow \{1, \dots, 6\}$ de la façon suivante :

1. pour i allant de n à 1 faire
 - (a) colorier v_i avec la plus petite couleur qui n'appartient aux couleurs de son voisinage dans G_i ;

Nous construisons une coloration utilisant 6 couleurs : à chaque itération i , le sommet a au plus 5 voisins et donc il peut être voisin de sommets ayant au plus 5 couleurs différentes.

□

Remarque : En fait, tout graphe planaire peut être colorer avec 4 couleurs. Ce résultat est connu sous le nom de théorème des quatre couleurs. Il a été démontré en 1976 par Appel et Haken.

Exercice 5 Le problème de l'ensemble indépendant

Le problème (de décision) de l'ensemble indépendant reste NP-complet même si les graphes sont planaires. Nous allons concevoir un algorithme calculant un ensemble indépendant de taille k pour les graphes planaires de n sommets en temps $O(6^k n)$.

Soit G un graphe planaire et k en entier. Par la question , il existe un sommet u_0 un sommet de degré au plus 5 : on notera ces voisins u_1, \dots, u_d avec $d \leq 5$.

Question 5.1 Supposons que G possède un ensemble indépendant J de taille $k > 0$. Prouver que

1. Si J contient aucun sommet u_i avec $0 \leq i \leq d$, alors, il existe un ensemble indépendant de taille k contenant u_0 .

Correction

Soit v un sommet de J , $J \setminus \{v\} \cup \{u_0\}$ est un ensemble indépendant de taille k puisque J et $J \setminus \{v\} \cup \{u_0\}$ ne contient aucun voisin de u_0 . □

2. Si J contient un sommet u_i avec $0 \leq i \leq d$, alors $J \setminus \{u_i\}$ est un ensemble indépendant de taille $k - 1$ dans le graphe G_i , correspondant à G privé de tous les sommets v voisins de u_i et de leurs ses arêtes incidentes.

Correction

Il suffit de constater que

1. les sommets $J \setminus \{u_i\}$ sont uniquement des sommets qui ne sont pas voisins de u_i : ils sont des sommets de G_i
2. les sommets $J \setminus \{u_i\}$ forment un ensemble indépendant dans G_i puisque ils forment un ensemble indépendant dans G

□

Question 5.2 Supposons que G ne possède pas un ensemble indépendant J de taille $k > 1$. Soit i entier entre 0 et d , Prouver que pour le graphe G_i correspondant à G privé de tous les sommets v voisins de u_i , ne possède pas un ensemble indépendant J de taille $k - 1$.

Considérons l'algorithme $\mathcal{A}(G, k)$ suivant

Entrée : un graphe planaire G et un entier k

Sortie : Vrai si et seulement si G possède un ensemble indépendant de taille k .

1. Si $k > |V(G)|$, alors renvoyer Faux.
2. Si $E(G) = \emptyset$ ou si $k = 0$, alors renvoyer Vrai.
3. Choisir un sommet u_0 de degré $d \leq 5$ avec pour voisins u_1, \dots, u_d
4. Pour tout i allant de 0 à d , construire le graphe G_i telque $V(G_i) = V(G) \setminus \Gamma_G(u_i)$ et $E(G_i) = E(G) \setminus \{e \in E : e \text{ a une extrémité dans } \Gamma_G(u_i)\}$
5. Renvoyer $\bigwedge_{i=0}^d \mathcal{A}(G_i, k - 1)$

Question 5.3 Exécuter cet algorithme sur une étoile à $n + 1$ sommets avec $k = 7$.

Correction

Les tests des lignes 1 et 2 ne s'appliquent pas. Puis la ligne 3 sélectionne une feuille u_0 qui est une feuille. On construit deux graphes : G_0 composé de 5 sommets isolés, et G_1 vide. Ensuite, $\mathcal{A}(G_0, 6)$ $\mathcal{A}(G_1, 6)$ sont évalués à Faux tous les deux (instruction 1). Et donc $\mathcal{A}(G_0, 7) = \text{Faux}$, ce qui est correct. \square

Question 5.4 Montrer que cet algorithme est correct.

Question 5.5 Donner la complexité de l'algorithme.

Correction

La construction de chaque graphe G_i nécessite $O(n + m)$ opérations. Au total, la construction de tous les graphes nécessite $O(5n + 5m) = O(5^2n)$ opérations. Soit a_k le nombre maximum de fois que l'on exécute l'instruction "construction des graphes G_i ". On

$$\begin{aligned} a_1 &= 1 = (6^0) \\ \text{a h} \quad a_k &\leq 1 + (6)a_{k-1} && \text{Par récurrence} \\ a_2 &\leq 1 + (6)a_1 \\ a_3 &\leq 1 + (6)a_2 \leq 1 + (6)^1 + (6)^2 \end{aligned}$$

$$a_k \leq (6^0) + 6^1 + 6^2 + 6^{k-1} < \frac{6^k - 1}{5} < \frac{6^k}{5}$$

$$a_k < \frac{6^k}{5}$$

Donc la complexité totale est $O(a_k 5^2n)$ soit $O(6^k 5n) = O(6^k n)$. \square

Question 5.6 Etendre ces résultats aux graphes t -dégénérés. Un graphe $G = (V, E)$ est t -dégénéré s'il existe un sommet v de degré inférieur ou égal à t dans G et si le graphe G privé de v possède la même propriété. Le e graphe G privé de v est identique à G , sauf qu'on lui a retiré le sommet v ainsi que toutes les arêtes incidente à v .