Polytech'Lille — GIS 3 — Calcul Numérique
Le Problème de la Coupe Maximale

---

1 Le projet

La principale difficulté consiste à accepter d'entrer dans le sujet, qui relève davantage de l'informatique théorique que de la thématique GIS.

C'est un sujet expérimental : je ne l'ai pas fait moi-même. On utilise des méthodes d'algèbre linéaire numérique sur des matrices exactes et l'expérience prouve que les algorithmes numériques classiques n'ont pas été prévus pour cela.

J'aimerais bien voir l'algorithme complet programmé. Et voir ce qu'il donne en pratique. Ce serait bien de lancer l'algorithme sur une famille d'exemples et de mesurer l'erreur par rapport à la solution optimale obtenue séparément par l'algorithme exponentiel naïf. Ce serait bien aussi de voir le comportement en pratique de l'algorithme de décomposition en valeurs singulières sur des matrices d'adjacence de graphes.

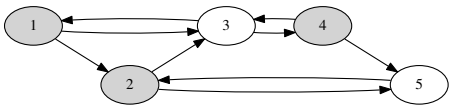
Le projet peut comporter une implantation de l'algorithme de SVD comme dans le sujet principal. Se reporter à ce sujet pour les détails mais utiliser impérativement la subroutine LAPACK spécialisée `DBDSQR` (la méthode fondée sur `DSYEV` ne marche pas sur des matrices d'adjacence de graphes).

2 Énoncé

Tout le document qui suit est fortement inspiré de [2, pages 22-24], avec quelques corrections mineures et quelques variantes dans l'énoncé des bornes lorsque je ne comprenais pas le raisonnement. La présentation de la SVD doit beaucoup à [3, Lectures 4-5].

Le problème de la coupe de valeur maximale consiste à partitionner l'ensemble S des n sommets d'un graphe orienté en deux sous-ensembles S_1 et $S_2 = S \setminus S_1$ de façon à maximiser la valeur de la coupe, c'est-à-dire le nombre d'arcs allant de S_1 vers S_2.

> **Exemple.** Voici un exemple de coupe de valeur maximale d'un graphe à $n = 5$ sommets. Les sommets de l'ensemble $S_1 = \{1, 2, 4\}$ sont grisés. Les sommets de l'ensemble $S_2 = \{3, 5\}$ sont blancs. La valeur de la coupe, c'est-à-dire le nombre d'arcs de S_1 vers S_2, est égale à 5.



3 Premières formulations

Soit A la matrice d'adjacence du graphe. À chaque sommet $1 \leq i \leq n$, on associe la variable indicatrice ξ_i, valant 1 si $i \in S_1$ et 0 sinon. On cherche donc un vecteur $x^T = \begin{pmatrix} \xi_1 & \xi_2 & \cdots & \xi_n \end{pmatrix}$ qui maximise la valeur de la coupe, c'est-à-dire le nombre

$$\sum_{1 \leq i,j \leq n} \xi_i \, (1 - \xi_j) \, a_{ij} \, . \tag{1}$$

En effet, on a $\xi_i \, (1 - \xi_j) \, a_{ij} = 1$ si, et seulement si, $i \in S_1$, $j \in S_2$ et il existe un arc de i vers j. En notant 1 le vecteur dont toutes les coordonnées valent 1, le problème à résoudre (maximiser (1)) peut se reformuler matriciellement en

$$\text{maximiser} \quad x^T \, A \, (1 - x) \quad (\xi_i \in \{0, 1\}) \, . \tag{2}$$

4 Considérations sur la complexité

La taille de la donnée est représentée par la variable n (le nombre de sommets). Le nombre d'arcs est majoré par n^2. On s'intéresse à la complexité en temps, dans le pire des cas. Cette complexité est donc une fonction $f(n)$ qui compte des opérations élémentaires (non précisées). La complexité est dite polynomiale si $f(n)$ est un polynôme en n. Les exposants ne doivent donc pas dépendre de n.

Il existe un algorithme naïf de complexité exponentielle : il consiste à énumérer les 2^n coupes possibles et en extraire une qui soit de valeur maximale.

Le problème de la coupe maximale est NP-complet. Par conséquent, si on connaissait un algorithme polynomial de résolution du problème de la coupe maximale, tout algorithme NP serait aussi P.

L'idée, c'est que la SVD (décomposition en valeurs singulières) peut être utilisée pour calculer une solution approchée du problème de la coupe maximale avec une complexité polynomiale. Quand on explique la SVD à quelqu'un au détour d'un couloir, on explique que « c'est une factorisation de matrice qui permet de trouver la matrice singulière la plus proche d'une matrice A donnée ». Un peu plus précisément, la SVD va nous permettre de calculer des matrices A_k $(1 \leq k \leq n)$ qui sont toutes des approximations de la matrice d'adjacence A, parce qu'elles vérifient (8), page 4 (voir [3, Theorem 5.8, page 35]). Très informellement (c'est l'idée bien qu'elle soit un peu fausse, énoncée comme ça), en travaillant sur A_k plutôt que sur A, l'exposant n de l'algorithme naïf va se transformer en exposant k. Par conséquent, pour k fixé, on va obtenir un algorithme de complexité polynomiale en n.

5 La décomposition en valeurs singulières

Le texte qui suit est fortement inspiré de [3, Lectures 4, 5]. On se restreint au cas d'une matrice réelle carrée A de dimension $n \times n$ (dans notre cas, A sera la matrice d'adjacence du graphe).

Voici l'idée dans le cas d'une matrice A de dimension 2×2. L'ensemble des vecteurs de longueur 1 peut être assimilé à un cercle C. L'ensemble des vecteurs $A\,v$ tels que $v \in C$ est une ellipse (pour un dessin, voir [3, Figure 4.1, page 26]). Cette ellipse a deux demi-axes principaux $\sigma_1\,u_1$ et $\sigma_2\,u_2$ où u_1 et u_2 sont deux vecteurs orthogonaux de dimension n de longueur 1 et $\sigma_1 \geq \sigma_2 \geq 0$ sont les longueurs des demi-axes. Les réels σ_i sont les *valeurs singulières* de A. Les vecteurs u_1, u_2 sont appelés *vecteurs singuliers gauches*. Il existe deux vecteurs $v_1, v_2 \in C$ (orthogonaux) tels que

$$A\,v_i \;=\; \sigma_i\,u_i\,.$$

Ces deux vecteurs sont appelés *vecteurs singuliers droits*. Notons

$$\begin{aligned}
\Sigma &= \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}, \\
U &= \begin{pmatrix} u_1 & u_2 \end{pmatrix}, \\
V &= \begin{pmatrix} v_1 & v_2 \end{pmatrix}.
\end{aligned}$$

On obtient alors la décomposition en valeurs singulières de A

$$\begin{aligned}
A\,V &= U\,\Sigma \\
A &= U\,\Sigma\,V^T
\end{aligned} \tag{3}$$

La décomposition en valeurs singulières (3) peut aussi s'écrire $A = A_n$, où

$$A_k \;=\; \sum_{i=1}^{k} \sigma_i\,u_i\,v_i^T\,. \tag{4}$$

Voir [3, Theorem 5.7, page 35]. Plusieurs théorèmes lient la norme deux et la norme de Frobenius de A à ses valeurs singulières. On rappelle que

$$\begin{aligned}
\|A\|_2 &= \max_{v \text{ t.q. } \|v\|_2=1} \|A\,v\|_2\,, \\
\|A\|_F &= \sqrt{\sum a_{ij}^2}\,.
\end{aligned}$$

On a les relations suivantes [3, Theorem 5.3, page 34] :

$$\|A\|_2 \;=\; \sigma_1\,, \tag{5}$$

$$\|A\|_F \;=\; \sqrt{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2}\,. \tag{6}$$

La relation (5) paraphrase la façon dont on a présenté la SVD. L'inégalité suivante donne une intuition sur la manière dont ces bornes vont pouvoir être utilisées

$$\|A\|_F \;\leq\; n \qquad \text{(parce que } a_{ij} = 0 \text{ ou } a_{ij} = 1) \tag{7}$$

Si on suppose les k premières valeurs singulières non nulles, les sommes partielles A_k sont les meilleures approximations possibles de A par une matrice de rang k. Plus précisément, on a pour tout $k < n$

$$\|A - A_k\|_2 \;=\; \inf_{rang(B)\leq k} \|A - B\|_2 \;=\; \sigma_{k+1} \tag{8}$$

Voir [3, Theorem 5.8, page 35].

6 Formulations approchées

On considère une décomposition en valeurs singulières (3) de la matrice d'adjacence A du graphe. On fixe un entier $k \leq n$ et on effectue une première approximation du problème initial (2) par le problème suivant, où A_k est définie dans (4) :

$$\text{maximiser} \quad x^T A_k \,(1 - x) \quad (\xi_i \in \{0, 1\})\,. \tag{9}$$

On effectue même une seconde approximation du problème initial en remplaçant A_k par $\tilde{A}_k$ définie par

$$\tilde{A}_k \;=\; \sum_{i=1}^{k} \sigma_i \,\tilde{u}_i \,\tilde{v}_i^T\,, \tag{10}$$

où $\tilde{u}_i$ et $\tilde{v}_i$ sont les vecteurs obtenus en approximant les coordonnées des vecteurs u_i et v_i par le multiple entier le plus proche de

$$\frac{1}{n\,k^2}.$$

On obtient le problème :

$$\text{maximiser} \quad x^T \tilde{A}_k \,(1 - x) \quad (\xi_i \in \{0, 1\})\,. \tag{11}$$

Exemple (suite). La matrice d'adjacence est

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Pour $k = 1$, on a en appliquant (4),

$$A_1 = \sigma_1 \,u_1 \,v_1^T \simeq \begin{pmatrix} 0 & 0.296 & 0.827 & 0 & 0.592 \\ 0 & 0.375 & 1.045 & 0 & 0.749 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0.375 & 1.045 & 0 & 0.749 \\ 0 & 0.078 & 0.218 & 0 & 0.0156 \end{pmatrix}.$$

On remarque que les matrices A_k ($k < n$) ne sont pas des matrices d'adjacence. Leurs coordonnées sont des réels de signe quelconque.

Dans l'algorithme de la section 8, on évite de former explicitement les matrices A_k et $\tilde{A}_k$ en considérant directement les vecteurs $u_i, v_i, \tilde{u}_i, \tilde{v}_i$. Pour $k = 1$, on est donc amené à considérer

$$\sigma_1 \simeq 2.189\,, \quad u_1 = \begin{pmatrix} 0.484 \\ 0.612 \\ 0 \\ 0.612 \\ 0.128 \end{pmatrix}, \quad v_1 \simeq \begin{pmatrix} 0 \\ 0.280 \\ 0.780 \\ 0 \\ 0.559 \end{pmatrix}.$$

Les coordonnées des vecteurs u_i et v_i sont des réels. La dernière approximation (10) consiste à les approximer par des rationnels, multiples de $1/5$:

$$\tilde{u}_1 = \begin{pmatrix} \frac{2}{5} \\ \frac{3}{5} \\ 0 \\ \frac{3}{5} \\ \frac{1}{5} \end{pmatrix}, \quad \tilde{v}_1 = \begin{pmatrix} 0 \\ \frac{1}{5} \\ \frac{4}{5} \\ 0 \\ \frac{3}{5} \end{pmatrix}.$$

7 La borne sur l'erreur

L'algorithme de la section 8 résout le problème (11) : il retourne un vecteur x, représentant une coupe, qui maximise (11). D'après les propositions 1 et 2, la valeur de cette coupe diffère de la valeur maximale des coupes d'une quantité inférieure ou égale à :

$$\frac{n^2}{\sqrt{k+1}} + \frac{3\,n^{\frac{3}{2}}}{k}\cdot$$

Remarque. Imaginons un algorithme qui tire une coupe aléatoirement. Quelle erreur ferait-il dans le pire des cas ? Il retournerait une coupe de valeur nulle alors que la coupe maximale est de valeur inférieure ou égale à

$$\frac{n^2}{4}\cdot$$

En effet, la valeur maximale des coupes d'un graphe à n sommets correspond à deux ensembles S_1 et S_2 ayant même nombre de sommets (donc de cardinal $n/2$) et tels qu'il existe un arc de chaque sommet de S_1 vers chaque sommet de S_2. On voit donc que l'algorithme de la section 8 ne commence à être intéressant que pour des valeurs de $k \geq 15$.

7.1 Preuves

Proposition 1 *Pour tout vecteur x ($\xi_i \in \{0,1\}$)*

$$|x^T A_k (1-x) - x^T A (1-x)| \;\leq\; \frac{n^2}{\sqrt{k+1}}.$$

Preuve Comme les vecteurs x et $1-x$ ne contiennent que des zéros et des uns, leur longueur est inférieure ou égale à $\sqrt{n}$. Par définition de la norme deux de la matrice A, on a donc

$$\|(A - A_k)(1-x)\|_2 \;\leq\; \sqrt{n}\,\|A - A_k\|_2 .$$

Parce que $x^T (A - A_k)(1-x)$ est le produit scalaire de x^T avec le vecteur $(A - A_k)(1-x)$, on a (inégalité de Cauchy-Schwarz [3, Lecture 3, page 21])

$$|x^T (A - A_k)(1-x)| \;\leq\; n\,\|A - A_k\|_2 . \tag{12}$$

De plus,

$$(k+1)\,\sigma_{k+1}^2 \;\leq\; \sigma_1^2 + \cdots + \sigma_{k+1}^2 \;\leq\; \|A\|_F^2 \;\leq\; n^2 .$$

La première inégalité est évidente. La deuxième découle de (6) et la troisième de (7). Par conséquent

$$\sigma_{k+1} \;\leq\; \frac{n}{\sqrt{k+1}}.$$

En utilisant (12) et (8), on conclut la preuve de la proposition. □

Les lemmes élémentaires ci-dessous permettent d'alléger la preuve de la proposition 2.

Lemme 1 *Si $|a|, |b| \leq M$ et $|a - a'|, |b - b'| \leq \delta \leq M$ alors $|a\,b - a'\,b'| \leq 3\,M\,\delta$.*

Preuve En effet,

$$|a\,b - a'\,b'| \;=\; |a\,(b-b') + b'\,(a-a')| \;\leq\; |a| \times |b-b'| + (|b| + |-b+b'|) \times |a-a'| \;\leq\; 3\,M\,\delta .$$

□

Lemme 2 *Soit a un vecteur de coordonnées $\alpha_1, \alpha_2, \ldots, \alpha_n$ et de longueur 1. Alors*

$$|\alpha_1 + \alpha_2 + \cdots + \alpha_n| \;\leq\; \sqrt{n}\,.$$

Preuve La formule suivante est vraie génériquement :

$$n\,(\alpha_1^2 + \alpha_2^2 + \cdots + \alpha_n^2) \;=\; (\alpha_1 + \alpha_2 + \cdots + \alpha_n)^2 + \sum_{i=1}^{n} \sum_{j=i+1}^{n} (\alpha_i - \alpha_j)^2 . \tag{13}$$

Comme a est de longueur 1 et que la double somme est positive ou nulle, on voit que $n \geq (\alpha_1 + \alpha_2 + \cdots + \alpha_n)^2$, ce qui conclut la preuve du lemme. □

Proposition 2 *Si un vecteur x est solution du problème* (11) *alors*

$$|x^T \tilde{A}_k (1-x) - x^T A_k (1-x)| \leq \frac{3 n^{\frac{3}{2}}}{k}.$$

Preuve On peut réécrire $|x^T \tilde{A}_k (1-x) - x^T A_k (1-x)|$ en

$$\left| \sum_{i=1}^{k} \sigma_i x^T \tilde{u}_i \tilde{v}_i^T (1-x) - \sum_{i=1}^{k} \sigma_i x^T u_i v_i^T (1-x) \right| = \left| \sum_{i=1}^{k} \sigma_i (x^T \tilde{u}_i \tilde{v}_i^T (1-x) - x^T u_i v_i (1-x)) \right|.$$

Chaque produit scalaire $x^T u_i \leq \sqrt{n}$ d'après le lemme 2, parce que u_i est de longueur 1 et que le produit par x^T revient à calculer une somme partielle des coordonnées de u_i. De même, chaque produit scalaire $v_i^T (1-x) \leq \sqrt{n}$.

Comme chaque produit scalaire $x^T \tilde{u}_i$ est lui-aussi une somme partielle des coordonnées de $\tilde{u}_i$ et que les coordonnées de ces vecteurs diffèrent de celles des vecteurs u_i d'au plus $1/(n k^2)$, on a

$$|x^T \tilde{u}_i - x^T u_i| \leq \frac{n}{n k^2} = \frac{1}{k^2}.$$

La même borne s'applique à $|\tilde{v}_i^T (1-x) - v_i^T (1-x)|$.

Le lemme 1 peut alors s'appliquer en prenant $M = \sqrt{n}$ et $\delta = 1/k^2$ et on a, pour tout $1 \leq i \leq k$,

$$|x^T \tilde{u}_i \tilde{v}_i^T (1-x) - x^T u_i v_i (1-x)| \leq \frac{3\sqrt{n}}{k^2}.$$

Par conséquent,

$$|x^T \tilde{A}_k (1-x) - x^T A_k (1-x)| \leq k \sigma_1 \frac{3\sqrt{n}}{k^2} \leq \frac{3 n^{\frac{3}{2}}}{k}.$$

Pour la première inégalité, on a majoré toutes les valeurs singulières par σ_1. Pour la seconde, on a utilisé (6) et (7). □

8 L'algorithme

Dans cette section, on note n_0 le nombre de sommets du graphe. Le symbole n désigne le nombre de sommets considérés jusqu'ici. Bien sûr, on fait varier n de 1 à n_0 de sorte qu'à la fin de l'algorithme, les notations sont rentrées dans le rang (si j'ose dire).

Dans cet algorithme, l'indice k est une constante. L'algorithme que nous allons établir sera polynomial en n mais exponentiel en k.

À tout vecteur x, de dimension n, on associe une matrice $W(x)$ de dimension $k \times 2$ définie comme suit. Les vecteurs $\tilde{u}_i$ et $\tilde{v}_i$ ont été précalculés une fois pour toutes, jusqu'à la

dimension n_0, par une décomposition en valeurs singulières. On ne forme pas explicitement la matrice $\tilde{A}_k$.

$$W(x) \;=\; \begin{pmatrix} x^T\,\tilde{u}_1 & \tilde{v}_1^T\,(1-x) \\ x^T\,\tilde{u}_2 & \tilde{v}_2^T\,(1-x) \\ \vdots & \vdots \\ x^T\,\tilde{u}_k & \tilde{v}_k^T\,(1-x) \end{pmatrix}.$$

L'algorithme fait varier n de 1 à n_0. À chaque itération, il construit un nouveau dictionnaire [1] D_n à partir du dictionnaire précédent D_{n-1}. Ces dictionnaires contiennent des couples $\langle W(x),\, x\rangle$. Les clés des couples sont les matrices $W(x)$. Il se peut très bien que deux vecteurs $x \neq x'$ définissent la même matrice $W(x) = W(x')$. Dans ce cas, le dictionnaire ne mémorise qu'un seul des deux couples. C'est un point clé (si j'ose dire) de l'algorithme, puisque c'est lui qui va assurer la complexité polynomiale. Enfin, on remarque que les clés sont des matrices de nombres rationnels : les tests d'égalité entre clés sont exacts.

Le premier dictionnaire D_0 contient un unique couple constitué d'un vecteur x de dimension 0 et de la matrice nulle de dimension $k \times 2$.

Plaçons-nous à l'étape n et construisons D_n à partir de D_{n-1}. Pour chaque couple $\langle W(x),\, x\rangle$ de D_{n-1}, on enregistre (au plus) deux couples dans D_n en prolongeant les $n-1$ coordonnées déjà fixées de x par $\xi_n = 1$ ou par $\xi_n = 0$. Dans chaque cas, la nouvelle matrice $W(x)$ s'obtient par une mise-à-jour élémentaire de l'ancienne : si $\xi_n = 1$, il suffit d'ajouter à la première colonne de W le vecteur formé des n-ièmes coordonnées des vecteurs $\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_k$; si $\xi_n = 0$, il suffit d'ajouter à la seconde colonne de W le vecteur formé des n-ièmes coordonnées des vecteurs $\tilde{v}_1^T, \tilde{v}_2^T, \ldots, \tilde{v}_k^T$.

À la fin de l'algorithme, on énumère tous les couples du dictionnaire et on extrait la solution x du problème (11) par un calcul de maximum naïf.

8.1 Complexité

Par un raisonnement très proche de celui tenu dans la preuve de la proposition 2, on remarque que $x^T\,u_i \leq \sqrt{n}$ et donc que

$$x^T\,\tilde{u}_i \;\leq\; \sqrt{n} + \frac{n}{n\,k^2} \;=\; \sqrt{n} + \frac{1}{k^2} \;=\; b\,.$$

Le produit scalaire $x^T\,\tilde{u}_i$ est une somme de multiples entiers de $1/(n\,k^2)$. Il est donc lui-aussi un multiple entier de $1/(n\,k^2)$. Dans l'intervalle $[-b, b]$ il existe au plus

$$p \;=\; (2\,b+1)\,n\,k^2 \;=\; 2\,n^{\frac{3}{2}}\,k^2 + n\,(k^2+2)$$

multiples entiers de $1/(n\,k^2)$. Les matrices $W(x)$ ne peuvent donc prendre que $p^{2\,k} \in \Theta(n^{3\,k})$ valeurs distinctes. La taille du dictionnaire et donc la complexité de l'algorithme sont alors majorés par une fonction polynomiale en n mais exponentielle en k.

1. Un dictionnaire au sens des structures de données [1, Partie 3, page 191]. Ceux qui ne connaissent pas peuvent assimiler un dictionnaire à une liste.

Remarque. Les bornes sur l'erreur commencent à être intéressantes pour $k \geq 15$. Sachant que $n \geq k$, dans le premier cas intéressant, la taille du dictionnaire est de l'ordre de $450\, n^{45}$ clés, ce qui rend cet algorithme inutile en pratique, bien qu'intéressant théoriquement.

Il faut bien sûr de se méfier de ce type de remarque. En d'autres temps, on a considéré que la multiplication d'entiers par FFT était un algorithme impraticable. La technologie a évolué et c'est aujourd'hui un algorithme courant. Enfin, les bornes sur l'erreur concernent le pire des cas. En pratique, l'algorithme se comporte peut-être beaucoup mieux que cela.

Références

[1] Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction à l'algorithmique*. Dunod, Paris, 2ème edition, 2002.

[2] Venkatesam Guruswami. Singular value decomposition. `https://www.cs.cmu.edu/~venkatg/teaching/CStheory-infoage/book-chapter-4.pdf`, 2012.

[3] Lloyd Nicholas Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997.