I Sous-graphe le plus dense

Soit $G = (S, A)$ un graphe non orienté à n sommets et p arêtes. Pour $S' \subseteq S$, on définit la fonction de densité par :

$$\rho(S') = \frac{|A(S')|}{|S'|}$$

où $A(S')$ est l'ensemble des arêtes de G ayant leurs deux extrémités dans S'.

1. Quelles sont les valeurs minimum et maximum de $\rho(S')$, en fonction de $|S'|$?

> Solution : Les minimum est $\rho(S') = 0$ quand S' n'a aucune arête et le maximum est $\rho(S') = \frac{|S'|(|S'|-1)}{2}$ quand S' est un sous-graphe complet.

On s'intéresse aux problèmes suivants :

DENSE

Entrée : un graphe $G = (S, A)$.
Sortie : un ensemble $S' \subseteq S$ tel que $\rho(S')$ soit maximum.

DENSE-DEC

Entrée : un graphe $G = (S, A)$, un entier k et un réel α.
Sortie : existe t-il un ensemble $S' \subseteq S$ tel que $|S'| = k$ et $\rho(S') \geq \alpha$?

CLIQUE-DEC

Entrée : un graphe $G = (S, A)$ et un entier k.
Sortie : existe t-il une clique de taille k ($S' \subseteq S$ tel que $|S'| = k$ et tous les sommets de S' sont adjacents) ?

2. En admettant que CLIQUE-DEC est NP-complet, montrer que DENSE-DEC est NP-complet.

> Solution :
>
> - DENSE-DEC $\in$ NP : on peut vérifier en temps polynomial qu'un ensemble S' est de taille k et que $\rho(S') \geq \alpha$.
> - DENSE-DEC $\geq$ CLIQUE-DEC : soit $G = (S, A)$ un graphe et k un entier. On pose $\alpha = \frac{k(k-1)}{2}$. Alors G a un sous-graphe complet de taille k si et seulement si G a un sous-graphe S' de taille k tel que $\rho(S') \geq \alpha$.

On propose un algorithme glouton pour DENSE :

- Itérativement retirer un sommet de degré minimum (ainsi que toutes les arêtes adjacentes) jusqu'à ce qu'il n'y ait plus de sommet.
- À chacune de ces itérations, calculer la valeur de ρ et conserver le maximum.

3. Expliquer comment on pourrait implémenter cet algorithme en complexité temporelle $\mathrm{O}(n + p)$.

> Solution : Pour obtenir $\mathrm{O}(n+p)$ en moyenne, on peut utiliser un tableau t tel que $t[i]$ soit une liste doublement chaînée des sommets de degré i. On conserve aussi en mémoire le i_{min} minimum tel que $t[i_{min}]$ soit non-vide et, pour chaque sommet, un pointeur vers sa postition dans sa liste chaînée de t.
> À chaque itération, on retire un sommet v de $t[i_{min}]$ et on met à jour les voisins de v (en les retirant de $t[j]$ et en les ajoutant à $t[j-1]$). On met aussi à jour la densité ($\frac{|A(S')|}{|S'|}$ est remplacé par $\frac{|A(S')| - \deg(v)}{|S'|-1}$) et i_{min}.
> Comme l'ajout et la suppression d'un élément dans une liste doublement chaînée se fait en temps constant, la complexité est bien $\mathrm{O}(n + p)$.

Soit S'^* tel que $\rho(S'^*)$ soit maximum, $v^* \in S'^*$ le premier sommet de S'^* retiré par l'algorithme glouton et S'' l'ensemble des sommets restants juste avant de retirer v^*.

4. Montrer que $\rho(S'') \geq \frac{\deg_{S''}(v^*)}{2}$, où $\deg_{S''}(v^*)$ est le degré de v^* dans S''.

Solution : Par choix de l'algorithme glouton : $\forall v \in S'', \deg_{S''}(v) \geq \deg_{S''}(v^*)$. Donc :

$$\rho(S'') = \frac{|A(S'')|}{|S''|} = \frac{\sum_{v \in S''} \deg_{S''}(v^*)}{2|S''|} \geq \frac{\deg_{S''}(v^*)|S''|}{2|S''|} \geq \frac{\deg_{S''}(v^*)}{2}$$

5. Justifier que $\rho(S'^*) \geq \rho(S'^* \setminus \{v^*\})$.

Solution : Par optimalité de S'^*.

6. En déduire que $\deg_{S'^*}(v^*) \geq \rho(S'^*)$.

Solution : En utilisant $\rho(S'^* \setminus \{v^*\}) = \dfrac{|A(S'^*)| - \deg_{S'^*}(v^*)}{|S'^*| - 1}$ et la question 6, on obtient $\deg_{S'^*}(v^*) \geq \rho(S'^*)$.

7. En déduire que l'algorithme glouton est une 2-approximation pour DENSE.

Solution :

$$\rho(S'') \overset{5}{\geq} \frac{\deg_{S''}(v^*)}{2} \overset{(*)}{\geq} \frac{\deg_{S'^*}(v^*)}{2} \overset{7}{\geq} \frac{\rho(S'^*)}{2}$$

Où $(*)$ vient de $S'^* \subseteq S''$.

II Ensemble dominant

Soit $G = (V, E)$ un graphe. Un ensemble dominant de G est un sous-ensemble D de V tel que tout sommet de V est soit dans D, soit adjacent à un sommet de D.
On note $d(G)$ la taille d'un plus petit ensemble dominant de G.

1. Calculer $d(G)$ si G est un chemin à n sommets.

Solution : Il existe un chemin dominant à $\left\lceil \dfrac{n}{3} \right\rceil$ sommets :

Comme chaque sommet peut dominer au plus 3 sommets, k sommets peuvent dominer au plus $3k$ sommets.
Ainsi, un ensemble dominant à k sommets doit vérifier $3k \geq n$ et donc $k \geq \left\lceil \dfrac{n}{3} \right\rceil$.

2. On suppose que G est connexe et contient au moins 2 sommets. Montrer que $d(G) \leq \left\lfloor \dfrac{n}{2} \right\rfloor$.

Solution : Soit T un arbre couvrant de G et v un sommet.
On considère les ensembles D_0 et D_1 de sommets à distance paire et impaire de v dans T.
D_0 et D_1 sont des ensembles dominants de G car tout sommet à distance paire est adjacent à un sommet à distance impaire (on utilise le fait que G pour cela) et réciproquement.
Et $D_0 \sqcup D_1 = V$ donc D_0 ou D_1 est de taille au plus $\dfrac{n}{2}$.

3. On suppose que G ne contient pas de sommet isolé (sommet de degré 0). Montrer que $d(G) \leq \left\lfloor \dfrac{n}{2} \right\rfloor$.

Solution : On peut appliquer la question précédente sur chaque composante connexe de G.

Une couverture par sommets de G est un sous-ensemble C de V tel que toute arête de G ait au moins une extrémité dans C.
On s'intéresse aux problèmes suivants :

DOMINANT

Entrée : Un graphe $G = (V, E)$ et un entier k.
Sortie : G possède-t-il un ensemble dominant de taille k ?

COUVERTURE
Entrée : Un graphe $G = (V, E)$ et un entier k. **Sortie** : G possède-t-il une couverture par sommets de taille k ?

4. Soit G un graphe sans sommet isolé. Est-ce qu'une couverture par sommets est un ensemble dominant ? Et réciproquement ?

> Solution : Soit C une couverture par sommets de G. Si v est un sommet de G alors v est adjacent à une arête couverte par C donc v est adjacent à un sommet de C. Donc C est un ensemble dominant de G.
> Par contre, si G est un triangle (cycle de longueur 3) et v un sommet de G alors $\{v\}$ est un ensemble dominant de G mais pas une couverture par sommets.

5. On admet que COUVERTURE est NP-complet. Montrer que DOMINANT est NP-complet.

> Solution : DOMINANT est dans NP car on peut vérifier en temps polynomial qu'un ensemble de sommets est dominant.
> Soit G, k une instance de COUVERTURE. On note n_i le nombre de sommets isolés de G.
> On construit G' à partir de G en ajoutant un sommet v_e à chaque arête $e = \{u, v\}$ de G et en ajoutant une arête entre v_e et u et une arête entre v_e et v.
> Supposons que G possède une couverture par sommets C de taille k et considérons D obtenu à partir de C en ajoutant les sommets isolés de G. Alors D est un ensemble dominant de G' de taille $k + n_i$.
> Supposons inversement que G' possède un ensemble dominant D de taille k'. Soit C obtenu à partir de D en remplaçant chaque sommet de la forme v_e par l'une des extrémités de e et en enlevant les sommets isolés.
> Pour chaque arête e de G, il y a au moins un sommet parmi u, v et e dans D. Donc u ou v est dans C, ce qui permet de conclure que C est une couverture par sommets de G de taille $k' - n_i$.
> G possède donc une couverture par sommets de k si et seulement si G' possède un ensemble dominant de taille $k + n_i$.
> Nous avons donc construit une réduction polynomiale de COUVERTURE vers DOMINANT, ce qui permet de conclure que DOMINANT est NP-complet.

6. Décrire un algorithme efficace pour résoudre DOMINANT si G est un arbre. L'implémenter en OCaml.

> Solution : fonction récursive qui renvoie un triplet (taille minimum d'un ensemble dominant, taille minimum d'un ensemble dominant contenant la racine, taille minimum d'un ensemble dominant ne contenant pas forcément la racine).

III k-Centres

k-CENTRES
Instance : un entier k et un graphe complet pondéré et non orienté $G = (S, A, d)$ dont les distances (poids) vérifient l'inégalité triangulaire ($d(u, w) \leq d(u, v) + d(v, u)$) Solution : un sous-ensemble $S \subseteq S$ de cardinal k Optimisation : minimiser $r = \max\limits_{v \in S} \min\limits_{s \in S} d(v, s)$.

Si l'on considère le cas particulier où les sommets du graphe sont des points du plan et le poids $d(u, v)$ est la distance euclidienne entre u et v, on demande donc de recouvrir S par k cercles de même rayon r centrés en k points de S, en minimisant r.

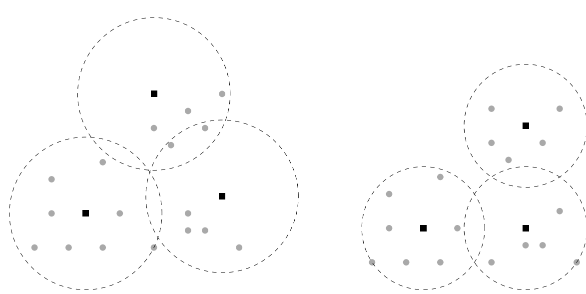

Figure 1: Une instance pour $k = 3$, avec deux solutions (la deuxième est meilleure).

On considère l'algorithme glouton suivant :

```
Choisir x un sommet quelconque
C ← {x}
Pour i = 1 à k − 1 :
    Choisir y tel que min_{c∈C} d(y, c) soit maximal.
    C ← C ∪ {y}
return C
```

On souhaite montrer que cet algorithme fournit une 2-approximation. On note C, r l'ensemble des centres et le rayon correspondant renvoyés par l'algorithme glouton et C^*, r^* une solution optimale.

1. Soit r la distance maximum d'un point à un centre en fin d'algorithme, u un point réalisant ce maximum et $S = C \cup \{u\}$. Montrer qu'il existe $c_0 \in C^*$ et deux éléments distincts x et y de S tels que $d(x, c_0) \leq r^*$ et $d(y, c_0) \leq r^*$.

2. Conclure.

Solution :

1. Pour chaque point x, il existe $c(x) \in C^*$ tel que $d(x, c(x)) \leq r^*$, par définition de r^*. Or S contient $k + 1$ éléments et C^* est de cardinal k, donc le principe des tiroirs assure qu'il existe $x, y \in S$ tels que $x \neq y$ et $c(x) = c(y)$, ce qui permet de conclure.

2. Il ne peut exister d'éléments $c, c' \in C$ tels que $d(c, c') < r$: en effet, le critère de choix utilisé dans l'algorithme assure que dans ce cas on aurait ajouté u à C plutôt que le deuxième de ces centres. On a donc $d(x, y) \geq r$ pour tous x, y distincts de S, et en particulier pour ceux trouvés à la question précédente. Or les distances vérifient l'inégalité triangulaire, donc $r \leq d(x, y) \leq d(x, c_0) + d(y, c_0) \leq 2r^*$.