TD5 – Approximation

Exercice 1 – Structures interdites

1. Montrer que le cycle à n sommets n'est pas un graphe d'intervalles si $n \geqslant 4$.
2. Montrer que l'étoile à 10 branches n'est pas un unit disk graph.

Exercice 2 – Set Cover

Un *hypergraphe* est une paire $H = (V, E)$ où V est un ensemble de sommets et E un ensemble de sous-ensembles de sommets, appelés *hyperarêtes*. Un *set cover* de H est un ensemble d'hyperarêtes S de H tel que tout sommet appartienne à au moins une hyperarête de S. On veut trouver un set cover de H de taille minimum.

1. Donner un set cover de taille minimum de l'hypergraphe de la Figure 1.

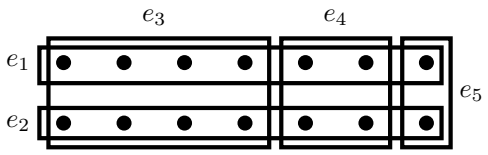


FIGURE 1 – Un hypergraphe à 14 sommets et 5 hyperarêtes

2. Si toutes les hyperarêtes sont de taille 2, l'hypergraphe est un graphe. Et on cherche alors un ensemble d'arêtes telles que tous les sommets appartiennent à au moins une d'entre elles. Montrer qu'un set cover dans un graphe a taille au moins $\frac{|V|}{2}$.
3. On repasse maintenant dans le monde des hypergraphes. On considère l'algorithme suivant :

Algorithme 1 : Algorithme glouton pour Set Cover

```
Input  : un hypergraphe H
Output : un set cover S de H
S ← ∅
tant que H contient un sommet faire
    e ← hyperarête de H de taille maximum
    S ← S ∪ {e}
    Supprimer de H tous les sommets de e
    pour f hyperarête de H faire
        Remplacer f par f \ e
retourner S
```

Exécuter l'algorithme sur l'hypergraphe de la Figure 1. Vous donnerez l'hyperarête choisie à chaque étape et dessinerez l'hypergraphe induit par les sommets restants.

4. En s'inspirant de la Figure 1, donner un exemple d'hypergraphe ayant un set cover minimum de taille 2 mais où l'algorithme glouton renvoie un set cover de taille 4.
5. Généraliser en montrant que l'algorithme glouton ne peut pas obtenir mieux qu'une $(\log n)/2$-approximation.
6. Représenter le problème du set cover minimum par un PLNE.
7. Écrire le dual de ce programme linéaire. L'interpréter comme un problème d'hypergraphes.

Exercice 3 – Explorer *Grand Line*

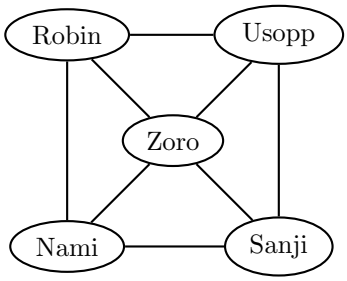
Mon trésor ? Je vous le laisse si vous voulez. Trouvez-le !
Je l'ai laissé quelque part dans ce monde ! – Gol D. Roger

À la recherche du *One Piece*, Luffy décide de séparer son équipage en deux équipes, chacune chargée d'explorer une île. Il est bien connu que son équipage ne cesse de se chamailler (par exemple Sanji et Zoro sont constamment en désaccord sur leurs compétences respectives), ce qui ralentit souvent leurs missions. En tant que capitaine, Luffy souhaite minimiser les conflits.

Pour résoudre son problème, Luffy crée un graphe G où chaque sommet représente un membre de l'équipage, et ajoute une arête entre deux sommets si les membres sont en conflit. Son but est de séparer les sommets en deux parties A et son complémentaire $\bar{A}$ de sorte à minimiser le nombre d'arêtes dans A et dans $\bar{A}$, ou, de manière équivalente, à maximiser le nombre d'arêtes entre A et $\bar{A}$.

Une telle séparation est appelée *coupe* de G, et sa *valeur* est le nombre d'arêtes entre les deux parties.

1. Déterminer une coupe maximum dans le graphe suivant :



2. Si le graphe est biparti, quelle est la taille de la coupe maximum ? Justifier.

Luffy n'a pas l'habitude de réfléchir avant d'agir. Il utilise donc l'algorithme glouton suivant : on commence en mettant tous les pirates dans la même équipe puis, on considère successivement chaque pirate et on le change d'équipe si cela augmente la valeur de la coupe. On répète ce processus autant que possible. Un pirate v est *déplaçable* dans la coupe $(A, \bar{A})$ si :

— soit $v \in A$ et $(A \setminus \{v\}, \bar{A} \cup \{v\})$ est une plus grande coupe que $(A, \bar{A})$.
— soit $v \notin A$ et $(A \cup \{v\}, \bar{A} \setminus \{v\})$ est une plus grande coupe que $(A, \bar{A})$.

L'algorithme est donc le suivant :

Algorithme 2 : Algorithme glouton pour Coupe Maximum

Input : un graphe G
Output : une coupe $(A, \bar{A})$ de G

$A \leftarrow \varnothing$
tant que G *contient un sommet* v *déplaçable* **faire**
  si $v \in A$ **alors**
    $A \leftarrow A \setminus \{v\}$
  sinon
    $A \leftarrow A \cup \{v\}$
retourner A

3. Combien de fois la boucle **tant que** peut s'exécuter au pire ?
4. En déduire la complexité de cet algorithme.

On note $\deg_S(v)$ le nombre de voisins de v dans S. Soit $(A, \bar{A})$ la coupe renvoyée par l'algorithme.

5. Montrer que si $v \in A$, alors $\deg_A(v) \leqslant \deg_{\bar{A}}(v)$, et si $v \notin A$, alors $\deg_A(v) \geqslant \deg_{\bar{A}}(v)$.

On note X l'ensemble des arêtes dont les deux extrémités sont dans A, Y l'ensemble des arêtes dont aucune extrémité n'est dans A, et Z le reste des arêtes.

6. Montrer que $\sum_{v \in A} \deg_A(v) = 2|X|$.
7. Montrer que $2|X| \leqslant |Z|$ et que $2|Y| \leqslant |Z|$.
8. En déduire que $|Z| \geqslant |E(G)|/2$.
9. En déduire que l'algorithme glouton renvoie une approximation de la coupe maximum.

Exercice 4 – Voyageur de commence.

Dans le problème du voyageur de commerce (VdC), on se donne une clique G et une fonction ω de pondération des arêtes. Le but est de trouver un cycle *hamiltonien* C (c'est-à-dire qui passe exactement une fois par tous les sommets) de poids minimum. On note $VdC(G, \omega)$ la valeur d'une solution optimale.

On dit qu'une instance du problème du voyageur de commerce est *métrique* si ω satisfait l'inégalité triangulaire, c'est-à-dire si $\omega(uv) + \omega(vw) \geq \omega(uw)$ pour chaque triplet de sommets (u, v, w) (intuitivement : "aller de u à w en passant par v est plus long que d'y aller directement").

Un cycle hamiltonien dans un graphe $G = (V, E)$ (pas forcement complet) est un cycle qui passe exactement une fois par chaque sommet.

1. Montrer que s'il n'existe pas d'algorithme polynomial pour trouver un chemin hamiltonien alors, pour toute constante ρ, le problème du VdC n'a pas de ρ-approximation en temps polynomial.
2. Soit $ST(G, \omega)$ le coût d'un arbre couvrant minimum de G. Montrer que $ST(G, \omega) \leqslant VdC(G, \omega)$.
3. Un *circuit* dans un graphe est une suite de sommets $x_1, \ldots, x_r$ tel que $x_r = x_1$ et pour tout i, $x_i x_{i+1}$ est une arête (certains sommets peuvent être répétés).
   Montrer qu'il existe un circuit qui passe par tous les sommets dont le poids est au plus $2ST(G, \omega)$.
4. En déduire une 2-approximation du problème VdC lorsque l'instance est métrique.
5. On dit qu'un graphe est *eulérien* si tous les sommets ont degré pair. Montrer que si un graphe est eulérien alors il existe un circuit qui passe exactement une fois par chaque arête.
6. On considère une instance métrique G du problème de VdC. Soit T un arbre couvrant de poids minimum et X l'ensemble des sommets de degré impair. Montrer que X a taille paire.
7. Montrer qu'il existe un couplage parfait de X dont le poids est au plus $\frac{1}{2} \cdot VdC(G, \omega)$.
8. En déduire une $\frac{3}{2}$-approximation pour le problème VdC.

Exercice 5 – k-centres

Si $\{P_1, \ldots, P_i\}$ est un ensemble de points du plan et P est un point, la distance $(P, \{P_1, \ldots, P_i\})$ est la distance minimum entre P et un point de $\{P_1, \ldots, P_i\}$, c'est-à-dire $\min_{1 \leq j \leq i} (P, P_j)$.

On se donne un entier k et un ensemble de n points $\mathcal{P}$ dans le plan. On cherche à sélectionner k points $P_1, \ldots, P_k$ dans $\mathcal{P}$, qui minimisent la quantité $\max_{P \in \mathcal{P}} (P, \{P_1, \ldots, P_k\})$. Cette quantité est appelée le *coût* de $\{P_1, \ldots, P_k\}$.

1. Lorsque $k = 1$, donner un algorithme permettant de résoudre le problème. Quelle est sa complexité ?

On considère l'algorithme suivant : on choisit un point P_1 quelconque. Puis, pour $i = 2, \ldots, k$, on choisit P_i comme un point de $\mathcal{P}$ qui est le plus loin de l'ensemble $\{P_1, \ldots, P_i\}$.

2. Quel coût est calculé par l'algorithme pour $k = 3$ si on considère l'ensemble de points $\{A, B, C, D, E, F\}$ dont les distances sont les suivantes ? On suppose qu'on choisit $P_1 = A$.
   Préciser les trois points choisis (et expliquez comment vous les avez trouvés).

	A	B	C	D	E	F
A	0	1	4	3	2	2
B	1	0	1	2	3	4
C	4	1	0	3	2	1
D	3	2	3	0	2	3
E	2	3	2	2	0	1
F	2	4	1	3	1	0

3. Comment s'appelle ce type d'algorithme ?
4. Quelle est la complexité de cet algorithme ?
5. On note $\{P_1, \ldots, P_k\}$ les points sélectionnés par notre algorithme et $\{P_1^*, \ldots, P_k^*\}$ une solution optimale, de coût d^*. On suppose maintenant par l'absurde que la solution renvoyée par l'algorithme a un coût strictement supérieur à $2d^*$, c'est-à-dire qu'il existe un point P à distance plus grande que $2d^*$ de tous les P_i.

(a) Montrer que si $i \neq j$ alors $\dot{(}P_i, P_j) > 2d^*$.

(b) En déduire que pour tout i, il existe un unique j tel que $\dot{(}P_i, P_j^*) \leqslant d^*$.

(c) Montrer que P est à distance au plus d^* d'un P_j^*.

(d) Déduire une contradiction. Quel facteur d'approximation pouvez vous en déduire pour l'algorithme décrit ci-dessus ?