TD3 : Algorithmes d'approximation

1. Clustering

On s'intéresse au problème du *clustering* qui cherche à regrouper des points du plan en *clusters* les plus petits possible. Plus formellement :

Clustering

Entrées	–	Un ensemble de points du plan $X = \{x_1, \cdots, x_n\}$ et un entier k.
Sortie	–	Une partition de X en k clusters $C_1, \cdots C_k$ telle que le diamètre $\max_j \max_{x,y \in C_j} d(x, y)$, est minimal, avec d la distance euclidienne .

1.1. Définir le problème de décision associé au problème d'optimisation ci-dessus.

1.2. Donner une réduction polynomiale du problème de décision du clustering au problème de coloration de graphe : étant donné un graphe G, et un entier k, existe-t-il un coloriage de G utilisant k couleurs ? Que peut-on en déduire pour le problème de clustering ?

Pour le problème d'optimisation, on considère l'algorithme suivant, en supposant pour simplifier qu'il n'existe pas deux paires de points de X séparés par la même distance.

Algorithme 1 cluster(X, k)

Choisir de façon arbitraire un point $y_1 \in X$ comme premier « centre » de cluster
for $i \in \{2, \ldots, k\}$ **do** Déterminer $y_i \in X$ qui maximise $\min_{j<i} d(\cdot, y_j)$
end for
for $i \in \{1, \ldots, k\}$ **do** Définir $C_i = \{x \in X \mid \forall j,\ d(x, y_i) < d(x, y_j)\}$
end for
Renvoyer $\max_j \max_{x,y \in C_j} d(x, y)$

1.3. Exhiber une instance du problème et une exécution de cet algorithme sur cette instance qui ne donne pas le diamètre optimal.

1.4. Déterminer le facteur d'approximation de l'algorithme cluster. On pourra considérer le point x le plus éloigné des k centres $y_1, \ldots, y_k$, c'est-à-dire qui maximise $\min_j d(\cdot, y_j)$, et sa distance au plus proche centre $r = \min_j d(x, y_j)$.

2. Ordonnancement

On s'intéresse ici au problème d'ordonnancement (minimisation) :

OTMIN

ENTRÉES – Entiers $n \geq 1$, $m \geq 1$ et $p_i \geq 0$ avec $1 \leq i \leq n$.

SORTIE – Une affectation de n travaux indépendants de durées respectives p_i sur m machines fonctionnant indépendamment minimisant le temps de fonctionnement des machines.

2.1. Formuler le problème comme la minimisation d'une fonction $\{0,1\}^{nm} \to \mathbb{N}$, en posant des contraintes sur les nm variables.

2.2. On note T^* le coût d'un ordonnancement optimal. Montrer que $mT^* \geq \sum_{i=1}^{n} p_i$.

On considère l'algorithme d'approximation qui affecte les travaux (donnés dans un ordre quelconque) à la première machine disponible.

2.3. Ecrire le pseudo-code et montrer que l'algorithme est polynomial.

2.4. Soit A_k le travail qui se termine en dernier dans la solution fournie par l'algorithme d'approximation, et t_k sa date de début. Montrer que $t_k \leq \frac{1}{m} \sum_{i \neq k} p_i$. En déduire une majoration du coût T_g de la solution de l'algorithme d'approximation, puis que le facteur d'approximation de l'algorithme est $\frac{2m-1}{m}$.

3. TSP et inégalité triangulaire

TSP

ENTRÉES – Un ensemble V de n villes, et une fonction $d : V \times V \to \mathbb{N}$.

SORTIE – Une tournée, c'est-à-dire d'une permutation $\sigma \in \mathfrak{S}_n$, minimisant la distance parcourue : $C(\sigma) = \Sigma_{i=1}^{n-1} d(v_{\sigma(i)}, v_{\sigma(i+1)}) + d(v_{\sigma(n)}, v_{\sigma(1)})$

On se place dans le cas où d satisfait l'inégalité triangulaire ($\forall i, j, k, d(v_i, v_j) \leq d(v_i, v_k) + d(v_k, v_j)$) et l'on considère l'algorithme suivant.

Algorithme 2 ApproxTSP(V, d)

Construire T un arbre couvrant de poids minimal.
Faire un parcours préfixe P de T. Construire une tournée σ en ne gardant que la première occurrence de chaque sommet dans P.
Renvoyer σ

Montrer que cet algorithme est une 2-approximation de TSP.