IFT800 - Série d'exercices #4

Algorithmes de branchement

Exercice 1. Dans le problème du CLUSTER-VERTEX-EDITING, on veut savoir s'il est possible de retirer au plus k **sommets** d'un graphe pour que chaque composante connexe soit une clique. Donnez un algorithme $O(3^k \cdot n^3)$ pour ce problème.

Défi. Essayez de faire mieux que ça. C'est-à-dire, un algo $O(a^k n^c)$ où $a < 3$. Ça se fait, mais les solutions que je connais semblent assez difficiles.

Exercice 2. Considérez la version de VERTEX-COVER avec poids sur les sommets, où on paramétrise par le poids total de la solution.

VERTEX-COVER avec poids
Entrée : graphe $G = (V, E)$ et fonction $f : V \to \mathbb{N}$ sur les sommets
Paramètre : k, le poids total d'une solution
Sortie : un ensemble $X \subseteq V$ qui couvre chaque arête de G tel que $\sum_{x \in X} f(x) \leq k$, ou *null* si non-existant

Soit $w = \min_{x \in V} f(x)$ le poids minimum. Notez que tous les poids sont des entiers non-négatifs. En supposant que $w \geq 1$, donnez un algorithme en temps $O(2^{k/w} \cdot n^c)$ pour ce problème.

Exercice 3. Considérez l'algorithme bidon suivant.

fonction *bidon*(I, k)
Si $k < 0$, return $false$
Si I satisfait une propriété X, return $true$
Créer quatre sous-instances I_1, I_2, I_3, I_4 d'une façon non-spécifiée
$S_1 = bidon(I_1, k-1)$
$S_2 = bidon(I_2, k-3)$
$S_3 = bidon(I_3, k-3)$
$S_4 = bidon(I_4, k-4)$
Si un de S_1, S_2, S_3, S_4 n'est pas $null$, le retourner, sinon retourner $null$

En notation O, combien de noeuds l'arbre de récursion de cet algorithme a-t-il?

Exercice 4. Nous allons améliorer la complexité de l'algorithme de branchement de VERTEX-COVER vu en classe.

1. Soit G un graphe dans lequel chaque sommet a $0, 1$ ou 2 voisins. Il est possible de montrer que dans G, chaque composante connexe est un chemin ou un cycle. Montrez que VERTEX-COVER peut se résoudre en temps polynomial sur un tel graphe.
2. Donnez un algorithme en temps $O(1.47^k \cdot n^c)$ pour VERTEX-COVER paramétré par k, la taille de la solution.

Exercice 5. Considérez le problème MAX-INDSET, que nous avons déjà rencontré en approximation. La version paramétrée est la suivante.

MAX-INDSET
Entrée : graphe $G = (V, E)$
Paramètre : k, le taille d'un ensemble indépendant
Sortie : un ensemble indépendant $X \subseteq V$ de taille k, ou null si non-existant

Ce problème est $W[1]$-complet. Par contre, si le degré de chaque sommet est borné, on peut développer un algorithme FPT.

Supposons que chaque sommet de G a au plus 4 voisins. Montrez qu'il existe un algorithme en temps $O(5^k \cdot n^c)$ pour ce problème.

Si vous préférez, vous pouvez montrer que MAX-INDSET pour se résoudre en temps $O((d+1)^k \cdot n^c)$, où d est le degré maximum du graphe.

Suggestion : utilisez le fait que si X est un ensemble indépendant maximum, alors pour tout sommet $v \in V(G)$, on a $v \in X$, ou bien un voisin de v est dans X. Idéalement, argumentez aussi ce fait.

Exercice 6. Un graphe est *planaire* si on peut le dessiner en deux dimensions sans que deux arêtes ne se croisent. Cette propriété n'est pas fondamentale pour l'instant. On a plutôt deux propriétés importantes sur les graphes planaires :

- si on enlève un sommet d'un graphe planaire, on obtient un autre graphe planaire;
- un graphe planaire a toujours un sommet qui a 5 voisins ou moins.

Montrez que le problème MAX-INDSET de l'exercice précédent peut se résoudre en temps $O(6^k \cdot n^c)$ sur els graphes planaires.

Exercice 7. Dans le problème MAX-SAT, on a en entrée un ensemble de clauses $C_1, \ldots, C_m$ sur variables $x_1, \ldots, x_n$. On veut savoir s'il existe une assignation des x_i qui satisfait au moins k clauses.

Donnez un algorithme FPT en le paramètre k pour ce problème. Un algorithme $O(2^k \cdot (n+m))$ n'est pas extrêmement difficile. Avec un peu plus de travail, on peut atteindre $O(1.618^k \cdot (n+m))$.

Kernelisation

Exercice 8. Dans le problème SET-SPLITTING, on a en entrée des ensembles $S = \{S_1, \ldots, S_n\}$ sur un univers U. On veut attribuer une couleur $f : U \to \{R, B\}$ à chaque élément. Notre but est qu'il y ait au moins k ensembles contenant au moins un élément de chaque couleur (donc, s'arranger pour que k ensembles contiennent un élément B et un élément R avec notre coloriage).

Donnez un noyau pour SET-SPLITTING.

Exercice 9. Donnez un noyau pour CLUSTER-EDITING, où le paramètre est le nombre d'arêtes à modifier (ajouter et supprimer) pour obtenir un graphe où chaque composante connexe est une clique.

Indice. Considérez trois règles de réduction: 1) si $v \in V$ n'est dans aucun P_3, supprimer v; 2) si $uv \in E$ est dans $k+1$ P_3, supprimer uv; 3) si $uv \notin E$ est dans $k+1$ P_3, ajouter uv.

Montrez qu'après avoir appliqué ces règles, le nombre de sommets est borné par une fonction de k. Pour bien faire les choses, vous devriez argumenter que ces règles sont saines.

Exercice 10. Donnez un noyau pour le problème 3-HITTING-SET.

Indice. D'abord, montrez que si deux éléments x et y appraissent dans beaucoup de triplets ensemble, alors on peut simplement enlever un de ces triplets. Ma définition de beaucoup est $k+2$, mais d'autres peuvent fonctionner.

Ensuite, montrez qu'après avoir fait les éliminations ci-haut, s'il y a un élément x qui est dans plus de $f(k)$ triplets, il faut que x soit dans la solution. On peut donc retirer x et réduire k de 1. Ici, ma définition de $f(k)$ est $k \cdot (k+1) + 1$.

Finalement, montrez qu'après application de ces règles, chaque x peut couvrir une quantité $f(k)$ de triplets, et donc qu'il doit y avoir au plus $k \cdot f(k)$ triplets.