Cours *conception et analyse d'algorithmes*
Correction du TD 8

## 1. Problème du placement d'usines

**i.** Le coût d'une affectation donnée par une bijection $\pi$ de $\{1,\dots,n\}$ vers $\{1,\dots,n\}$ (l'usine $U_i$ est affecté à l'emplacement $E_{\pi(i)}$) est $C(\pi) = \sum_{1\leq i<j\leq n} v_{i,j}c_{\pi(i),\pi(j)}$.

Une affectation partielle est définie par deux sous-ensembles $I$ et $P$ de $\{1,\dots,n\}$ correspondant respectivement aux usines et aux emplacements affectés et par une bijection $\alpha$ de $I$ vers $P$ (l'usine $U_i$ est affectée à l'emplacement $E_{\alpha(i)}$). On notera également $J = \{1,\dots,n\}\setminus I$ et $Q = \{1,\dots,n\}\setminus P$.

Une borne inférieure sur le coût d'une affectation $\pi$ complétant une application partielle $(I,P,\alpha)$ est donnée par

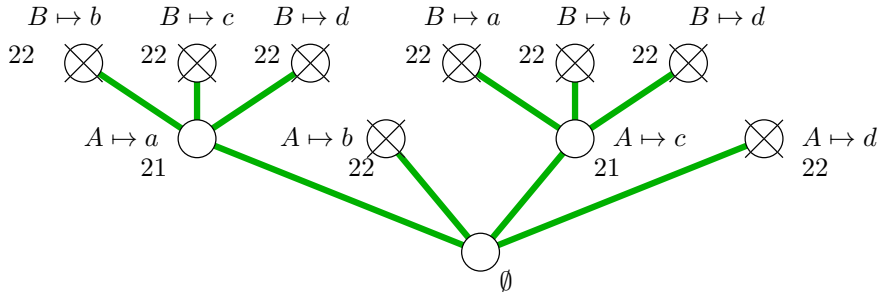
$$B(I,P,\alpha) = \sum_{i<i'\in I} v_{i,i'}c_{\alpha(i),\alpha(i')} + \sum_{i\in I,j\in J} v_{i,j}c_{\alpha(i),\beta_i(j)} + \sum_{j<j'\in J} v_{j,j'}c_{\gamma(j,j')},$$

où $\forall i \in I$, $\beta_i$ est une bijection de $J$ vers $Q$ telle que $\forall j,j' \in J$, $v_{i,j} < v_{i,j'} \Rightarrow c_{\alpha(i),\beta_i(j)} \geq c_{\alpha(i),\beta_i(j')}$ et où $\gamma$ est une bijection de $J^2$ vers $Q^2$ telle que $\forall j,j',k,k' \in J$, $v_{j,j'} < v_{k,k'} \Rightarrow c_{\gamma(j,j')} \geq c_{\gamma(k,k')}$.

**ii.** Le calcul des bijections $\beta_i$ $\forall i \in I$ et de la bijection $\gamma$ revient à faire $p$ tris sur $q$ éléments (coût $O(pq\log q)$) plus un tri sur $q^2$ éléments (coût $O(q^2\log(q))$). Il faut ensuite sommer toutes les contributions (coût $O(n^2)$). On obtient donc une complexité $O(n^2 + pqlog(q) + q^2log(n))$ qui est majorée par $O(n^2logn)$.

**iii.** Pour l'affectation $\pi$ le coût est $C(\pi) = 22$. La borne inférieure pour l'affectation partielle $\alpha : A \mapsto d$ est $B(\{A\},\{d\},\alpha) = 22$ qui n'est pas inférieure à $C(\pi)$. Cette branche ne sera donc pas explorée lors d'un algorithme *branch and bound*.

**iv.** L'arbre représentant une recherche *branch and bound* avec connaissance à priori d'une affectation $\pi$ de score $C(\pi) = 22$ est dessiné ci-dessous.




## 2. Couverture de points par des lignes

On utilise la réduction suivante : s'il existe une droite couvrant un ensemble $S$ de plus de $k$ points, alors on les supprime de $P$. L'instance $(P,k)$ se réduit alors en $(P\backslash S, k-1)$.

Nous avons trois points à démonter :

1. Il s'agit bien d'une réduction. En effet, si l'instance $P\backslash S, k-1)$ est acceptée, alors $(P,k)$ a aussi une solution. Inversement, si $(P,k)$ a une solution alors $(P\backslash, k-1)$ doit aussi en avoir une, car nécessairement une seule droite couvre $S$ dans la solution de $(P,k)$, et donc les $k-1$ autres droites suffisent à couvrir $P\backslash S$. En effet si une seule droite ne couvre pas $S$, alors $|S|$ différentes droites doivent être utilisées pour couvrir $S$, ce qui est impossible puisque $|S| > k$.
2. La réduction est polynomiale. En effet, on peut supposer que chaque droite couvre au moins deux points ou alors est horizontale, si bien qu'il y a au plus $n(n-1)/2 + n = \mathcal{O}(n^2)$ droites candidates à tester. La réduction est polynomiale.
3. Le noyau (c'est-à-dire l'instance réduite finale) est bien de taille au plus $g(k)$. Si on ne peut plus appliquer la réduction et qu'il existe plus de $k^2$ points, alors il n'y a pas de solution. Donc ce problème a un noyau de taille $k^2$.

### 3. Ensemble indépendant pour les graphes planaires

**i.** Chaque face du dessin du graphe est bordée d'au moins trois arêtes et chaque arête borde deux faces, cela permet d'obtenir la relation suivante entre nombre de faces et nombre d'arêtes : # faces $\leq$ 3/2# arêtes. En utilisant la formule d'Euler, on obtient qu'un graphe planaire à $n$ sommets possède au plus $3n-6$ arêtes.

**ii.** On déduit que tout graphe planaire possède un sommet de degré au plus 5. En effet $\sum_{u\in V(G)} \deg(u) = 2m$, où $m$ est le nombre d'arêtes de $G$. Or, si tous les sommets $u$ vérifient $\deg(u) \geq 6$, on aurait que $2m = \sum_u \deg(u) \geq 6n$, ce qui implique que $m \geq 3n$, contradiction.

**iii.** On considère le programme suivant (où $B(u,1)$ désigne la boule de rayon 1 centrée en $u$, c'est-à-dire $u$ et l'ensemble de ses voisins).

**Entrée :** un graphe $G$ et un entier $k$

**Sortie :** Vrai ssi $G$ possède un Ensemble indépendant de taille $k$.

1. Si $k \geq |V(G)|$, renvoyer Faux.
2. Si $E(G) = \emptyset$ ou si $k = 0$, renvoyer Vrai.
3. Choisir un sommet $u_0$ de degré $d \leq 5$, avec pour voisins $u_1, \ldots, u_d$.
4. Pour tout $i \in \{0, \ldots, d\}$, construire le graphe $G_i := G\backslash B(u_i, 1)$.
5. Renvoyer $\bigvee_{i=0}^{d} Algo(G_i, k-1)$.

Montrons par récurrence sur $k$ que l'algorithme est correct. Il l'est pour $k = 0$, et supposons que $Algo(H, k-1)$ soit correct pour tout graphe $H$. Soit $u_0$ un sommet de $G$ ayant pour voisins $u_1, \ldots, u_d$, il y a deux cas :

- Soit $G$ possède un ensemble indépendant de taille $k$, dans ce cas il en possède un contenant un des $u_i$. En effet sinon, on peut toujours rajouter $u_0$ à l'ensemble indépendant construit sans briser l'indépendance et il suffit d'enlever n'importe lequel des autres sommets pour obtenir un ensemble indépendant de taille exactement $k$.
  Soit $i_0$ l'indice d'un tel $u_i$, alors $Algo(G_{i_0}, k-1)$ est Vrai et sera évalué correctement par récurrence.
- Si $G$ ne possède pas d'ensemble indépendant de taille $k$, en appliquant le même raisonnement, pour tout $i \in \{0, \ldots, d\}$, $Algo(G_i, k-1)$ est faux et donc $Algo(G, k)$ sera évalué à faux.

L'opération la plus coûteuse, à part les appels récursifs, est la ligne 4, qui prend un temps $d \times \mathcal{O}(n+m) = \mathcal{O}(dn + 5dn) = \mathcal{O}(n)$ car le nombre d'arêtes est $n \leq tn/2$.

Soit $a_k$ le nombre maximum de fois que la ligne 4 est exécutée pour un appel à $Algo(G, k)$, on a alors $a_k \leq 1 + 5a_{k-1}$ avec $a_0 = 0$. Cela implique que :

$$a_k \leq \sum_{i=0}^{k-1} 6^i < 6^k.$$

La complexité totale est donc en $\mathcal{O}(6^k \cdot n)$.

**4. Réduction à un noyau de couverture par sommets**

**i.** On considère l'algorithme suivant :

1. Si $E(G) = \emptyset$ et $k \geq 0$, renvoyer VRAI.
2. Si $|E(G)| > k \cdot \Delta(G)$, renvoyer FAUX.
3. On pose $H := G$ et $p := 0$.
   Tant que $p < k$ et $\exists\ u \in VH$ tel que $\deg_h(u) > k - p$ faire :
   $H := H \backslash \{u\}$ et $p := p + 1$.
4. Si $|E(H)| > (k-p)^2$, renvoyer FAUX.
5. Supprimer les sommets isolés de H.

Montrons que l'algorithme ci-dessus est correct. Le test de l'étape 2 est valide, car un sommet $u$ appartenant à une couverture ne peut couvrir qu'au plus $\deg(u) \leq \Delta(G)$ arêtes.

Après l'étape 3, $G$ possède une couverture de taille $k$ si et seulement si $H$ possède une couverture de taille $k - p \geq 0$. Le test de l'étape 4 se justifie de la même manière que celui de l'étape 2. Si $|E(H)| > (k-p) \cdot \Delta(H)$, alors la couverture n'existe pas. Or, à cause de la double condition du "tant que" de l'étape 3, soit $p = k$, soit $\Delta(H) \leq k - p$. Dans les deux cas, le test est valide.

Après l'étape 4, $H$ possède au plus $(k-p)^2$ arêtes. Donc après l'étape 6, il possède au plus $2(k-p)^2$ sommets. Le problème possède donc un noyau de taille $\mathcal{O}(k^2)$.

Cet algorithme peut être implémenter en temps $\mathcal{O}(kn)$.

**ii.** Si $\Delta(G) \leq 2$, alors $G$ est une union disjointe de sommets isolés, de cycles et de chemins. Clairement, la taille d'une couverture par sommets minimum pour un chemin ou cycle à $t$ arêtes est $\lceil t/2 \rceil$.

**iii.** On considère l'algorithme suivant :

1. Si $E(G) = \emptyset$ et $k \geq 0$, renvoyer VRAI.
2. Si $k \leq 0$, renvoyer FAUX.
3. Si $\Delta(G) \leq 2$, conclure grâce à **ii.**
4. Choisir un sommet $u$ de $G$ de degré $\geq 3$.
5. Construire $G_1 := G \backslash \{u\}$ et $G_2 := G \backslash N(u)$.
6. Renvoyer $Algo(G_1, k-1) \bigvee Algo(G_2, k - \deg_G(u))$.

Il faut montrer que $(G, k)$ est VRAI ssi $(G_1, k-1)$ ou $(G_2, k - \deg_G(u))$ est VRAI. Si $G$ possède une couverture à $k$ sommets, alors soit $u$ appartient à cette couverture et donc $G_1$ possède une

couverture à $k-1$ sommets, soit $N(u)$ appartient à la couverture et donc $G_2$ possède une couverture à $k - \deg_G(u)$ sommets (si $\deg_G(u) < 0$, l'étape 2 renvoie FAUX, ce qui est correct.

Calculons la complexité de l'algorithme. Les instructions les plus coûteuses (en dehors des appels récursifs) sont les instructions 3 et 5 qui prennent un temps $\mathcal{O}(n+m)$. Soit $a_k$ le nombre maximum de fois que l'une des instructions 3 ou 5 est exécutée lors d'un appel à $Algo(G, k)$. La complexité totale est donc $\mathcal{O}(a_k(n+m))$.

On a donc $a_0 = 0$, $a_1 = 1$ et $a_2 = 2$. De manière générale $a_k \leq 1 + a_{k-1} + a_{k-3}$ pour tout $k > 2$. On pose $c = 5^{1/4} \sim 1.495$ et $\alpha = \max_{k\in\{0,1,2\}}\{(a_k+1)/c^k\} \sim 1.34$, on montre par récurrence que $a_k \leq \alpha \cdot c^k - 1$, pour tout $k \geq 0$. D'où une complexité totale de $\mathcal{O}(5^{k/4} \cdot (n+m))$.

**iv.** Il suffit d'appliquer l'algorithme de **i** pour se réduire à un noyau de taille au plus $k$, l'algorithme de la question **iii.** permet donc de conclure en $\mathcal{O}(5^{(k/4)} \cdot (k)^2)$, ce qui donne une complexité totale de $\mathcal{O}(kn + 5^{(k/4)} \cdot (k)^2)$.