Approximation

On ne connait pas d'algorithme polynomiaux pour résoudre les problèmes d'optimisation NP-complets, mais pour certains on connaît des algorithmes efficaces qui retournent une "bonne" solution. Encore faut-il savoir ce qu'on entend par "bonne".

Problème d'optimisation $\mathcal{P}$, ensemble d'instance $\mathcal{I}$: soit v^* la valeur d'une solution optimale pour une instance $I \in \mathcal{I}$.
Algorithme $\mathcal{A}$ donné : retourne une valeur $v(\mathcal{A}, I)$
⇒écart entre $v^*(\mathcal{I})$ et $v(\mathcal{A}, I)$?
On note $\alpha(\mathcal{A}) = \max_{I \in \mathcal{I}} v(\mathcal{A}, I)/v^*(\mathcal{I})$
⇒$\alpha(\mathcal{A}) \geq 1$ si $\mathcal{P}$ est un problème de minimisation (et $v^*(I) > 0$)
⇒pour toute instance I : $v(\mathcal{A}, I) \geq \alpha(\mathcal{A}) v^*(I)$
Plus alpha est petit (proche de 1), meilleure est la garantie sur la qualité de la solution retournée par $\mathcal{A}$.

Le voyageur de commerce avec inégalité triangulaire

Un ensemble $\mathcal{V}$ de n sommets / villes, une fonction de coût $c : \mathcal{V} \times \mathcal{V} \to \mathbf{N} \cup \{+\infty\}$. On cherche un cycle qui passe par toutes les villes de coût minimal.

Algorithme approché
1. calculer un arbre couvrant A de coût minimal v ; *//algorithme de Kruskal*
2. $s \leftarrow$ un sommet de $\mathcal{V}$;
3. $C \leftarrow$ la liste des sommets rencontrés lors d'un parcours "préfixe" de A ;
4. $C' \leftarrow$ la liste obtenue à partir de C en ne gardant que la première occurrence de chaque sommet ;
5. $C' \leftarrow C' \cdot s$;

On montre que $\alpha(\mathcal{A}) \leq 2$.

Autrement dit : si on trouve, avec $\mathcal{A}$, un circuit de coût w, alors le coût d'un circuit optimal est $v^* \geq w/2$.

Remarque : un autre algorithme a un $\alpha = 1,5$. (Nicos Christofides, 1976)

Le voyageur de commerce SANS inégalité triangulaire Pas d'algorithme polynomial ayant $\alpha < +\infty$!!

Ensemble couvrant minimal

- Ensemble de villes $V = v_1, \ldots, v_n$
- Pour chaque ville v : $\mathsf{Couv}(v) \subseteq V$

⇒ On cherche $S \subseteq V$ tel que $\bigcup_{v \in S} \mathsf{Couv}(v) = V$

- Optimisation : on veut $|S|$ minimal

Algorithme glouton
1. $S \leftarrow \{\}$; $U \leftarrow \{\}$
2. Tant que $U \neq V$ faire :
   (a) $v \leftarrow v \in V - S$ qui maximise $|\mathsf{Couv}(v) - U|$
   (b) $S \leftarrow S + v$; $U \leftarrow U + \mathsf{Couv}(v)$
3. Retourner S

Propriété Le coefficient d'approximation de cet algorithme est en $\Omega(\log(n))$.

Remarque Pas très bon, si 100 villes, $\log(100) = 4,6$...
On montre que, quelque soit $\epsilon > 0$, il ne peut y avoir d'algorithme ayant un $\alpha \leq (1 - \epsilon)\log(n)$.

Exercice 1 Etant donné un graphe $G = (S, A)$, où S est l'ensemble des sommets et A l'ensemble des arêtes, le problème de *couverture de sommets* consiste à trouver un sous-ensemble S' de S, le plus petit possible, tel que toute arête (x, y) de A ait au moins un sommet dans S'.

Question 1.1 Montrez le déroulement de l'algorithme sur le graphe en exemple de l'algorithme de Kruskal.
Question 1.2 En considérant l'algorithme ci-contre, donnez un k le meilleur possible tel que si S^* est une solution optimale, et S' une solution retournée par l'algorithme ci-dessus, alors $|S'| \leq k|S^*|$ (en justifiant ce résultat !).

1. $S' \leftarrow \{\}$.
2. Tant que $A \neq \{\}$ faire :
   (a) choisir une arête $\{x, y\} \in A$;
   (b) $S' \leftarrow S' \cup \{x, y\}$;
   (c) $A \leftarrow A - \left\{\begin{array}{l}\text{les arêtes dont un som-}\\ \text{met au moins est } x \text{ ou } y\end{array}\right\}$
3. retourner S'.