Feuille d'exercices n°11 - Algorithmes d'approximation

Notions abordées

- Problème d'optimisation, problème de décision associé
- Problèmes BinPacking, VoyageurCommerce et une variante de Knapsack
- Stratégie gloutonne, non optimale, admettant ou non un ratio d'approximation
- Preuve de non existence d'un schéma d'approx. reposant sur une réduction
- Arbre couvrant, cycles et chemins hamiltoniens
- Forêt associée à un parcours de graphe

Exercice 1 : Un problème proche du Knapsack

On considère dans cet exercice le problème d'optimisation suivant.

$$\text{SommeMax}_O : \begin{cases} \textbf{Entrée :} \text{ Un entier } n \in \mathbb{N}, \text{ une suite finie } (a_i)_{i\in[\![1,n]\!]} \in \mathbb{N}^n, \text{ un entier } B \in \mathbb{N} \\ \textbf{Sortie } \textbf{:} \max\{\sum_{i\in I} a_i | I \subseteq [\![1,n]\!] \sum_{i\in I} a_i \leqslant B\} \end{cases}$$

Q. 1 Donner le problème de décision SommeMax associé au problème d'optimisation SommeMax_O.

Q. 2 Quel est le lien entre les problèmes de décision SommeMax et Knapsack ?
Que peut-on en déduire ?

Q. 3 Montrer que SommeMax est NP-difficile. *On peut ici admettre les résultats de NP-difficulté vus en TD.*

Q. 4 Proposer un algorithme glouton **en temps linéaire** qui construit une solution pour le problème SommeMax_O.

Q. 5 L'algorithme proposé est-il optimal♣ ? Sinon construire des instances pour lesquelles la solution produite par cet algorithme est particulièrement mauvaise, et ce en vue de montrer que cet algorithme n'admet pas de rapport d'approximation standard.

Q. 6 Proposer un algorithme glouton en $\mathcal{O}(n \log n)$ qui évite l'écueil des précédentes instances.

Q. 7 Montrer que l'algorithme proposé à la question précédente est une $\frac{1}{2}$-approximation pour SommeMax_O. *On pourra commencer par trouver une borne supérieure de la valeur optimale M, et ensuite raisonner par disjonction de cas selon que l'algorithme a produit une solution de valeur $\geqslant \frac{M}{2}$ ou $< \frac{M}{2}$.*

Q. 8 Proposer un algorithme en $\mathcal{O}(n)$ qui est lui aussi une $\frac{1}{2}$-approximation pour SommeMax_O. *On pourra apporter une petite amélioration fort bénéfique à l'algorithme proposé en **Q 4**.*

♣. On demande ainsi si cet algorithme renvoie toujours une solution optimale.

Exercice 2 : Le problème du BinPacking

Le problème de décision BinPacking est défini comme suit.

$$\text{BinPacking} \quad : \begin{cases} \textbf{Entrée :} \text{ Un entier } n \in \mathbb{N},\ (t_i)_{i\in[\![1,n]\!]} \in \mathbb{N}^n,\ C \in \mathbb{N}^* \text{ et un seuil } K \in \mathbb{N} \\ \textbf{Sortie :} \text{ Existe-t-il } \varphi : [\![1,n]\!] \to [\![1,K]\!] \text{ telle que } \forall i \in [\![1,K]\!], \sum_{j\in\varphi^{-1}(\{i\})} t_j \leqslant C \text{ ?} \end{cases}$$

Q. 1 Donner le problème d'optimisation BinPacking_O associé au problème BinPacking.

Q. 2 Montrer que BinPacking est NP-complet.
On admet ici que le problème Partition est NP-difficile.

Ce problème d'optimisation peut être vu comme la recherche du nombre minimal de boîtes de tailles C qu'il faut pour ranger n objets de tailles respectives $(t_i)_{i\in[\![1,n]\!]}$. Pour résoudre ce problème, on propose l'algorithme glouton qui consiste à traiter les objets par indices croissants, et à ranger chacun d'eux dans la dernière boîte ouverte si c'est possible, et à ouvrir une nouvelle boîte sinon.
On remarque que si le nombre total de boîtes ainsi utilisées est K, alors le rangement obtenu est une fonction **croissante** de $[\![1,n]\!] \to [\![1,K]\!]$, puisqu'un objet i traité avant j (*i.e.* $i \leqslant j$) ne peut être rangé dans une boîte ouverte après celle où est rangé j, *i.e.* $\varphi(i)$ ne peut être strictement supérieur à $\varphi(j)$, soit $\varphi(i) \leqslant \varphi(j)$.

Q. 3 Donner le pseudo-code de l'algorithme glouton décrit ci-avant.

Q. 4 Proposer une instance sur laquelle cet algorithme est optimal, et une sur laquelle il ne l'est pas.

Q. 5 Donner un minorant et un majorant de la valeur optimale pour une instance (n,t,C).

Q. 6 Montrer que l'algorithme glouton proposé fournit une 2-approximation pour BinPacking_O.
On pourra raisonner sur la parité de la valeur optimale.

Q. 7 Pour $p \in \mathbb{N}^*$, on considère $I_p = (n,t,C)$ où $n = 2p$, $C = p$ et $\forall i \in [\![1,n]\!], t_i = \begin{cases} C \text{ si } i \text{ est pair} \\ 1 \text{ sinon} \end{cases}$
Donner la valeur optimale pour cette instance d'une part, et la valeur renvoyée par l'algorithme glouton d'autre part.

Q. 8 Déduire des questions précédentes quel est le ♣ rapport d'approximation de l'algorithme glouton proposé.

Dans la seconde partie de cet exercice on s'intéresse à la complexité du problème, mais pas seulement au regard des classes P et NP. On ne s'intéresse pas à la difficulté de résoudre BinPacking_0 mais à celle de l'approximer.

Q. 9 Montrer qu'il n'existe pas d'algorithme de complexité polynomiale de ratio d'approximation $\frac{3}{2} - \varepsilon$ avec $\varepsilon \in \mathbb{R}_+^\star$ pour le problème (à moins que P =NP). *On pourra utiliser une réduction bien choisie depuis le problème* Partition.

♣. **le** rapport d'approximation d'un algorithme est le meilleur rapport d'approximation qu'il admet.

Exercice 3 : Le problème VoyageurCommerce

Le problème VoyageurCommerce est le problème de minimisation associé au problème de décision suivant, où l'on qualifie d'**hamiltonien** un chemin ou un circuit élémentaire qui passe par chaque sommet.

VoyageurCommerce $\begin{cases}\textbf{Entrée :} \text{ Un graphe orienté } G=(S,A),\ d\in\mathcal{F}(A,\mathbb{R}^+), \text{ un seuil } K\in\mathbb{N}.\\ \textbf{Sortie :} \text{ Existe-t-il un circuit hamilt. dont la longueur selon } d \text{ est } \leqslant K\ ?\end{cases}$

Ici on se restreint au cas où le graphe est **complet** et où la pondération des arcs est **symétrique** et vérifie **l'inégalité triangulaire**, ce qui modélise bien la recherche d'un tour de distance kilométrique minimale. Puisque, pour chaque arc, l'arc opposé existe dans le graphe et qu'il est de même coût, on suppose dans la suite que les graphes en entrée sont non orientés. Ce problème est parfois appelé voyageur de commerce euclidien.
Un tour définit donc un cycle qui couvre tous les sommets du graphe, et donc presque un arbre couvrant : il suffit d'enlever une arête quelconque.

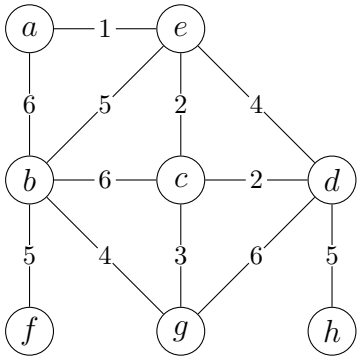
Q. 1 Justifier la dernière affirmation.

Q. 2 En utilisant la remarque précédente, expliquer comment trouver un minorant sur la longueur d'un tour optimal.

Q. 3 Suffit-il d'ajouter une arête quelconque à une arbre couvrant pour former un tour ? Ou existe-t-il pour tout arbre couvrant une arête qu'il suffit d'ajouter pour former un tour ?

En vue de construire un tour à partir d'un arbre couvrant, on va s'intéresser au parcours en profondeur de celui-ci, et raisonner sur l'arborescence associée. La **forêt associée à un parcours** L de G (non orienté) est un graphe orienté des sommets parcourus, dans lequel les arcs indiquent comment les sommets ont été ajoutés au parcours. Plus précisément, pour chaque sommet L_j qui n'est pas un point de régénération, la forêt contient un unique arc (L_i, L_j) avec $i < j$ et L_i voisin de L_j (dans le graphe G) . On dit dans ce cas que i est le **père** de j, et c'est bien l'un des sommets qui justifient que j est dans la bordure de $L[1..j[$. Chaque point de régénération du parcours est en revanche de degré entrant nul dans la forêt, (*i.e.* il n'y a aucun arc (L_i, L_j) si L_j est point de régénération).
Dans le cas d'un parcours en profondeur, on veut que la forêt témoigne du caractère en profondeur, on choisit donc comme père du sommet L_j le dernier sommet voisin de L_j dans $L[1..j[$, *i.e.* L_i où $i = \max\{k \in [\![1, j[\![\mid\ L_k\text{voisin de } L_j\}$. Cette condition impose que le père de chaque sommet est unique (pour un parcours donné). Si de plus le graphe de départ est connexe, le parcours n'a qu'un point de régénération, et la forêt associée n'a qu'une seule arborescence. Ces remarques justifient qu'on parle dans la suite de l'arborescence associée à un parcours en profondeur d'un arbre couvrant.

Q. 4 Pour le graphe ci-dessous, calculer un ACPM puis un parcours en profondeur de cet ACPM, et dessiner l'arborescence associée. Proposer un tour construit à partir de ce parcours.



Q. 5 Proposer une manière de fabriquer un tour à partir d'un arbre couvrant. Établir un lien entre la longueur du tour et le poids de l'arbre.

Q. 6 Déduire des questions précédentes un algorithme d'approximation et préciser son rapport d'approximation.