TD4 : schémas d'approximation

1. Partition

Partition
ENTRÉES – des entiers positifs $a_1, \cdots, a_n$ codés en binaire.
SORTIE – oui, s'il existe $I \subseteq \{1, \cdots, n\}$ tel que $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$.

1.1. Donner le problème d'optimisation associé au problème de partition.

1.2. Soit l'algorithme glouton `Glouton` qui consiste à prendre les tâches dans l'ordre, et à les allouer au processeur le moins chargé.

1. Sans perte de généralité, on suppose que la tâche qui termine l'ordonnancement est celle de temps a_j ($j \leq n$) et était placée sur le processeur P_1. On note T_i le temps d'exécution sur P_i, pour $i \in \{1, 2\}$.
   Montrer que $2T_1 \leq T_1 + T_2 + a_j$.
2. En déduire que `Glouton` fournit une $\frac{3}{2}$-approximation.
3. Donner une instance pour laquelle ce facteur d'approximation est atteint.

1.3. On considère à présent `GloutonTri` qui, avant d'appliquer `Glouton`, trie les tâches par temps d'exécution décroissant.

1. Avec les notations précédentes, montrer que si $a_j > \frac{T^*}{3}$, alors $j \leq 4$.
2. Prouver que `GloutonTri` est une $\frac{7}{6}$-approximation.
3. Donner une instance pour laquelle ce facteur d'approximation est atteint.

1.4. Soit $\varepsilon > 0$ fixé. Pour $I = \{a_1, \cdots, a_n\}$ une instance de Partition, on note $A = \sum_{i=1}^{n} a_i$, $m = \max_{1 \leq i \leq n} a_i$, et $L = \max(\frac{A}{2}, m)$. On distingue les *grosses tâches* telles que $a_i > \varepsilon L$ et les *petites tâches* telles que $a_i \leq \varepsilon L$. On note enfin $P = \{i \mid a_i \leq \varepsilon L\}$ et $A_p = \sum_{i \in P} a_i$ la somme des temps d'exécution des petites tâches.
À partir de l'instance I, on définit celle, notée J qui contient toutes les grosses tâches de I, et $\lfloor \frac{A_p}{\varepsilon L} \rfloor$ tâches de taille exactement εL.

1. Montrer que $T^*(J) \leq (1 + \varepsilon) T^*(I)$.
2. Donner un algorithme en temps polynomial à ε fixé pour trouver un ordonnancement optimal pour l'instance J.
3. À partir de τ un ordonnancement optimal pour l'instance J, définir un ordonnancement σ pour I tel que

$$T^\sigma(I) \leq T^*(J) + 2\varepsilon L \ ,$$

où $T^\sigma(I)$ dénote le temps d'exécution des tâches de I sous l'ordonnancement σ, et $T^*(J)$ le temps d'exécution optimal de J.

4. Que peut-on en déduire sur la classe de complexité de Partition ?