CentraleSupélec

Algorithmique et complexité

TD 6/7 – Problème d'empaquetage

L'objectif de ce TD est de comparer différentes heuristiques d'une même famille pour résoudre/approcher un problème NP-complet.

Exercice 1 : Analyse d'un problème

Une entreprise met à disposition de ses usagers des services de *Cloud Computing*. Ce sont de puissants ordinateurs accessibles sur réservation pour réaliser des calculs très consommateurs en CPU. Chaque client a la possibilité de réserver du temps de calcul pour des tâches. Chaque tache sera traitée sur une machine unique.

De son coté, la société dispose de M serveurs principaux disponibles B minutes par jour. Elle peut au besoin louer des serveurs complémentaires mais souhaite, dans la mesure du possible, ne pas y avoir recours. Tous les soirs, elle doit affecter chacune des N tâches de ses clients à une machine. Les tâches seront ensuite exécutées le lendemain.

L'entreprise souhaite concevoir un logiciel permettant de déterminer s'il est possible d'affecter chaque tâche à une machine sans avoir recours à la location de serveurs de calcul supplémentaires.

Question 1

Donnez une définition formelle de ce problème de décision (donnez les entrées et la question du problème).

Question 2

Montrez que le problème formalisé précédemment est *NP*-Complet. Pour cela, on pourra s'appuyer sur le problème de Partition qui est également *NP*-Complet.

> **PARTITION**
> **Entrée** : Soit E un ensemble d'entiers positifs.
> **Question** : Existe-t-il une partition de E en deux sous-ensembles E_1 et E_2 telle que $\sum_{e_i \in E_1} e_i = \sum_{e_j \in E_2} e_j$

Question 3

Définissez le problème sous forme de problème d'optimisation (donnez les entrées et la question du problème).

Question 4

Pourquoi pense-t-on qu'il n'existe pas d'algorithme en temps polynomial pour résoudre ce problème d'optimisation ?

Exercice 2 : Algorithmes de résolution

Le problème d'optimisation que l'on souhaite résoudre est le problème du Bin-Packing, vu à l'exercice précédent :

Question 1

Proposez un algorithme **glouton** pour résoudre ce problème. Il en existe plusieurs.

Question 2

Écrivez cet algorithme (en pseudo-code ou en python) et donnez sa complexité.

Question 3

Testez cet algorithme sur l'instance suivante où la taille des sacs est 10 et la taille des éléments est :

$$4, 4, 5, 5, 5, 4, 4, 6, 6, 2, 2, 3, 3, 7, 7, 2, 2, 5, 5, 8, 8, 4, 4, 5$$

Question 4

Essayez d'estimer les performances de votre algorithme : de combien de fois au pire une solution renvoyée par votre algorithme dépasse le nombre minimal de serveurs.