CentraleSupélec

Algorithmics and complexity

TD 6/7 – Packing Problem

The goal of this TD is to compare different heuristics of the same family to solve/approximate an NP-complete problem.

Exercise 1 : Problem Analysis

A company provides *Cloud Computing* services to its customers. This company has powerful machines available upon reservation to perform computations that are CPU-intensive. Every customer can execute its tasks after reserving some amount of computation time. Each task will be processed on a single machine.

The company has M servers available B minutes a day. It may need to rent additional servers but wishes, as far as possible, to avoid this option. Every night, it must affect each of the N customers tasks to a machine. The tasks are then executed the next day.

The company wants to design a software to know if it is possible to assign each task on a machine without the need to rent additional servers.

Question 1

Give a formal definition of this problem (give the entries and the question of the problem).

Question 2

Show that the previously formalized problem is *NP*-Complete. You can use the *NP*-Complete Partition problem:

> **PARTITION**
> **Inputs** : Let E a set of natural numbers.
> **Question** : is there a partition of E into two sets E_1 and E_2 such that:
>
> $$\sum_{e_i \in E_1} e_i = \sum_{e_j \in E_2} e_j$$

Question 3

Define the corresponding optimization problem (give the entry and the question of the problem).

Question 4

Why do we think that there is no polynomial time algorithm to solve this problem?

Exercice 2 : Resolution algorithms

We want to solve the Bin-Packing problem formalized above:

Question 1

Propose a greedy algorithm to solve the problem. Are there many?

Question 2

Write the algorithm (in pseudo-code or in Python) and give its complexity.

Question 3

Run your algorithm on an instance with a bin of size 10 and the following items:

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

Question 4

Estimate the efficiency of your algorithm : in the worst case, how far is your approximate solution from the minimal number of servers you need?