Exercice 1 : Serveurs et calculs

Q1 : Modélisation $(d_i)_{1 \leq i \leq N}$ Π serveur $B \in \mathbb{R}^+$ borne

Existe-t-il une fonction $f : [1, N] \longrightarrow [1, \Pi]$

telle que $\forall j \in [1, \Pi], \sum_{i \in f^{-1}(j)} d_i \leq B$: la somme des durées des tâches affectées à un serveur ne dépasse pas B.

Q2 : Sachant que Partition est NP-complet, montrer Bin Packing est NP-complet

Partition : Un ensemble E d'entiers positifs

Existe-t-il une partition E_1, E_2 de E

telle que $\sum_{e_i \in E_1} e_i = \sum_{e_j \in E_2} e_j$

1°) Bin Packing est NP : Vérifier en temps polynomial qu'une fonction f est solution

pour chaque i, vérifier que $1 \leq f(i) \leq \Pi$ $\mathcal{O}(N)$

pour chaque i, soit $j = f(i)$, on ajoute d_i à D_j

ensuite, on vérifie que $\forall j$, $D_j \leq B$

$\mathcal{O}(N + \Pi)$

2°) Bin Packing est NP-difficile

On va réduire Partition à Bin Packing

Soit $I_p = \langle (e_i)_{1 \le i \le N} \rangle$ une instance de Partition

On construit $I_{BP} = \langle (d_i)_{1 \le i \le N}, \Pi, B \rangle$ une instance de Bin Packing

avec : $d_i = e_i$

$\Pi = 2$

$B = \frac{1}{2} \sum e_i$

$\Rightarrow$ Si I_p est positive, il existe E_1 et E_2 une partition de E telle que

$\sum_{E_1} e_i = \sum_{E_2} e_j$ En mettant sur le serveur 1 les éléments de E_1 et sur le serveur 2 ceux de E_2

$= \frac{1}{2} \sum_{E} e_k$ on a une affectation qui respecte la borne B

$\Leftarrow$ Si I_{BP} est positive, on met dans E_1 les tâches du serveur 1

E_2 ——— 2

$\sum_{E_1} e_i \le \frac{1}{2} \sum_{E} e_j$ et $\sum_{E_2} e_i \le \frac{1}{2} \sum_{E} e_j$

et $\sum_{E_1} e_i + \sum_{E_2} e_j = \sum_{E} e_k$ $\Rightarrow \sum_{E_1} e_i = \frac{1}{2} \sum_{E} e_j = \sum_{E_2} e_k$

$\Rightarrow$ On a une partition

Q3 : Problème d'optimisation associé ?

Trouver l'affectation des N tâches sur le moins de serveurs possible (minimiser Π à B fixé).

Q4 : Résoudre le problème d'optimisation, c'est trouver k la plus petite valeur de Π qui permet d'affecter les N tâches. Si on a un algo polynomial pour trouver k, la résolution du problème de décision se fait en comparant Π à k. On aurait donc un algo polynomial pour résoudre un problème NP-complet

Or on pense actuellement que $P \neq NP$

Exercice 2 : Algorithmes de résolution approchés

Q1 : Algorithme glouton

First Fit : on traite les objets un par un. On place chaque objet dans le 1er sac qui peut le contenir. Si aucun sac ne convient, on ouvre un nouveau sac.

First Fit Decreasing : idem, mais en traitant les objets par poids décroissant

Next Fit : on place chaque objet dans le seul sac qu'on considère s'il tient dedans. Sinon, on ferme le sac et on place l'objet dans un nouveau sac

Variante : On a un seul sac ouvert, on essaie d'y placer le maximum d'objets, on passe au sac suivant si aucun objet ne peut y être placé.

Best Fit : on place chaque objet dans le sac qui peut l'accepter mais qui a la plus petite capacité résiduelle.

Best Fit Decreasing : idem mais en traitant les objets par poids décroissant.

Q4 : Qualité de l'approximation

On prend comme unité de poids la capacité B des sacs.

On normalise les poids $p(o_i) = \frac{o_i}{B}$

Nombre minimal de sacs nécessaires

Il faut au minimum $\left\lceil \sum_{i=1}^{N} p(o_i) \right\rceil$

Nombre maximal de sacs utilisés :

Pour First Fit : on aura au plus un unique sac rempli à moins que la moitié.

s'il y en avait 2, le contenu du 2e sac aurait dû être placé dans le premier puisqu'il y tient

Donc tous les sacs, sauf peut-être 1, sont remplis à plus de la moitié, on n'utilisera donc pas plus de

$$\left\lceil 2 \sum_{i=1}^{N} p(o_i) \right\rceil \text{ sacs.}$$

En fait, FF et BF sont des $\frac{17}{10}$-approx

BFD et FFD sont des $\frac{11}{9}$-approx

Une k-approx ne donne jamais une solution qui utilise plus de $k \times opt$ sacs si opt est le nombre optimal.