Algorithmique Avancée, Complexité, calculabilité

TP n° 4 : heuristiques

Université de Lille-1 / FIL

12 novembre 2017

Description du problème

Dans ce TP, on s'intéresse à un problème d'*ordonnancement* (« *scheduling* »). On doit gérer un *cluster* (une « grappe ») de m machines (qui sont tous du même modèle). Les machines portent les numéros $0, 1, \ldots, m-1$.

Les utilisateurs de ce cluster soumettent des *tâches* (« *jobs* ») qui doivent s'exécuter sur le cluster. La i-ème tâche est catactérisée par :

- a_i : sa date d'arrivée (en secondes, le cluster est allumé à $t = 0$). C'est la date à laquelle la tâche est soumise au cluster par un utilisateur.
- size_i : le nombre de machines nécessaires pour l'exécuter.
- t_i : son temps d'exécution (en secondes).

Il s'agit de plannifier l'exécution de chacune des n tâche sur l'ensemble des m machines (**attention** : contrairement au TP précédent, les tâches peuvent nécessiter plus d'une machine). Il vous suffira de trouver la date de début D_i de chaque tâche (en secondes, depuis $t = 0$). Comme toutes les machines sont identiques, un algorithme simple affectera alors les machines aux tâches. La i-tâche aura donc *attendu* un temps $W_i = D_i - a_i$ avant de démarrer.

L'objectif auquel on s'intéresse consiste à minimiser la somme des délais, c'est-à-dire qu'on cherche un ordonnancement qui minimise $\sum W_i$.

Au passage, la date de complétion de la i-ème tâche est $C_i = a_i + W_i + t_i$. La somme des dates de complétion est donc :

$$\sum_{i=1}^{n} C_i = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} t_i + \sum_{i=1}^{n} W_i.$$

La somme des dates d'arrivée et la somme des temps d'exécution sont des caractéristiques de l'instance, qui ne dépendent pas de l'ordonnancement des tâches sur le cluster. Par conséquent, minimiser la somme des délais est équivalent à minimiser la somme des dates de complétion de tâches.

Dans la nomenclature standard des problèmes d'ordonnancement, ce problème se nomme « $P|r_j, \text{size}_j|\sum C_j$ ».

Format d'entrée-sortie

Une instance du problème $P|r_j, \text{size}_j|\sum C_j$ est constituée par un entier m (le nombre de machines du cluster), un entier n (le nombre de tâches), puis la liste des tâches. Chaque tâche est donnée par un triplet $(a_i, \text{size}_i, t_i)$.

Les instances du problème sont donc décrites par un fichier du type :

```
100
7
0 50 1200
0 20 3600
0 20 3600
0 20 3600
2400 75 120
```

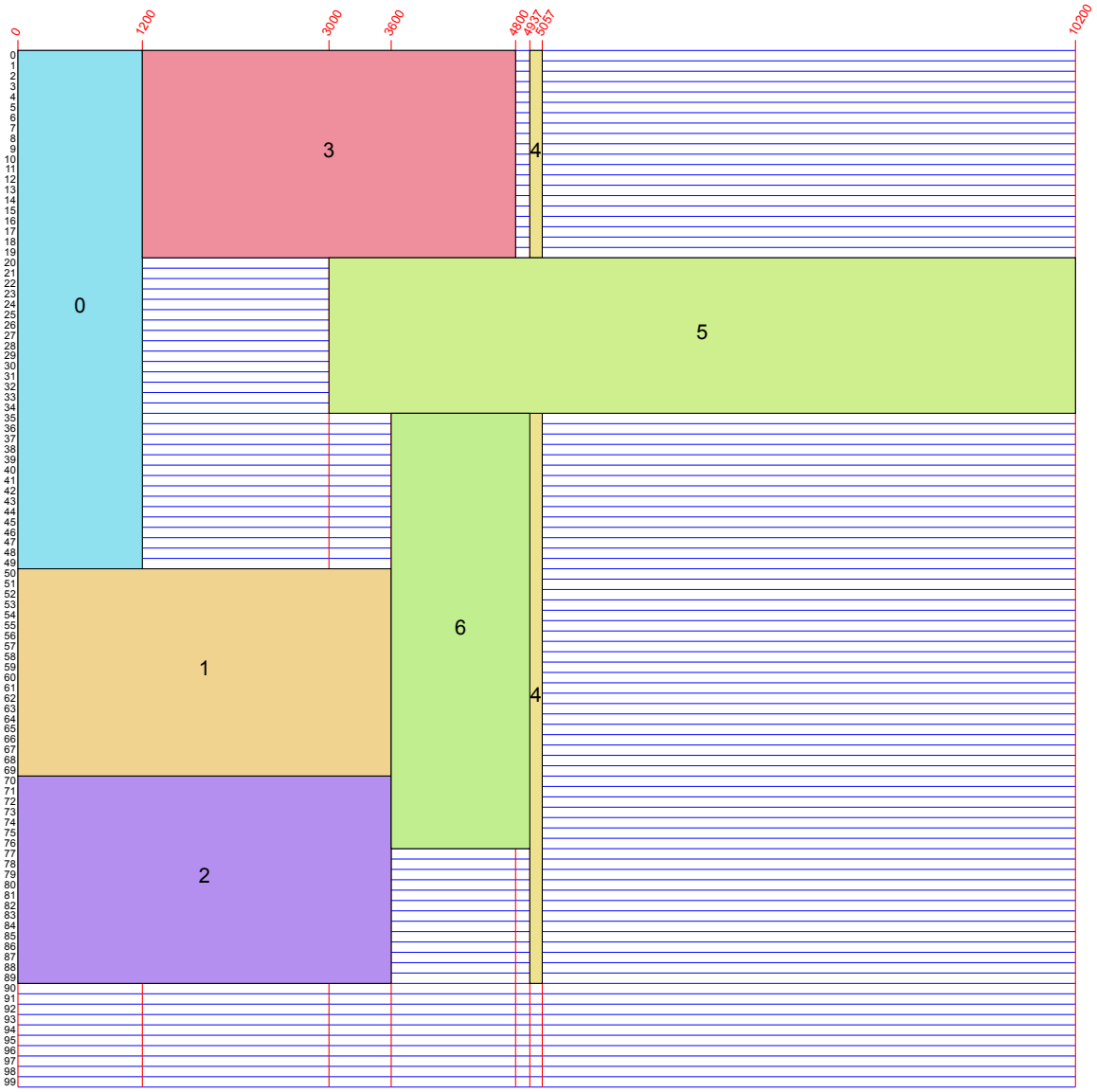
```
3000 15 7200
3400 42 1337
```

Les tâches sont données par date d'arrivée croissante. Une solution est un ordonnancement, c'est-à-dire la donnée de D_i pour chacune des tâches. Bien sûr, il faut que suffisamment de machines soient disponibles pour que la tâche puisse démarrer. Une solution potentielle de l'instance ci-dessus est donc :

```
0
0
0
1200
3600
3720
3720
```

Chaque ligne correspond à une tâche, dans le même ordre que dans le fichier d'entrée : il s'agit de sa date de début D_i.

Un ordonnancement est aisément représenté par un *diagramme de Gantt*. Un script python fourni génère des diagrammes de Gantt au format `SVG` à partir d'une instance et d'un ordonnancement. Avec les deux fichiers ci-dessus, on obtient :



Travail à accomplir

Vous trouverez une série d'instances sur la plate-forme d'entrainement à l'algorithmique du FIL (`http://contest.fil.univ-lille1.fr`). Trouvez des solutions de bonne qualité et soumettez-les sur le site de la plate-forme.

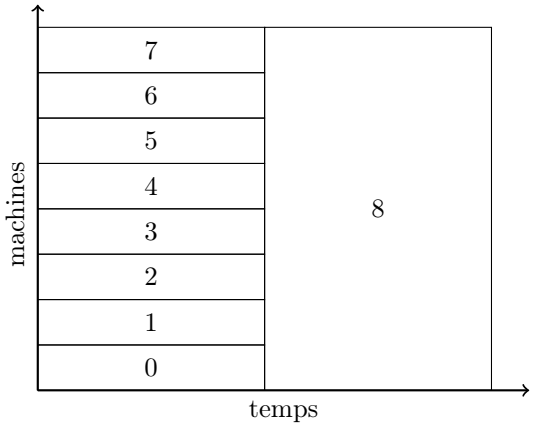
Suggestions

Tous les ordonnancements optimaux possibles peuvent être obtenus par la procédure suivante : mettre les tâches dans une liste selon un ordre déterminé ; tant que la liste n'est pas vide, exécuter la première tâche de la liste dès que c'est possible. Le seul problème... c'est de trouver l'ordre.

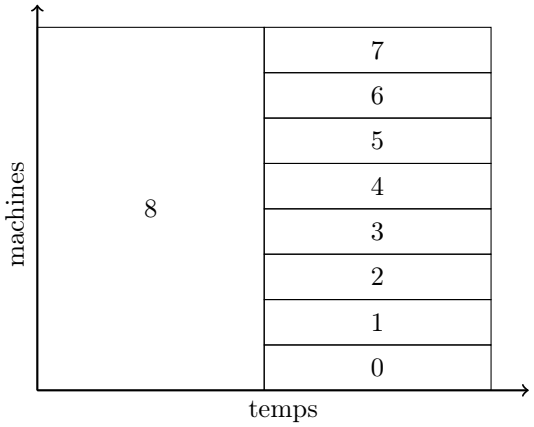
Les techniques gloutonnes constituent une piste (très) raisonnable à poursuivre dans un premier temps. Dans cette optique, voici deux remarques :

Il vaut mieux mettre les petites tâches d'abord

Par exemple, considérons ces 9 tâches qui durent 1 heure, toutes soumises à $t = 0$. Dans le dessin de gauche, seule la tâche #8 subit un délai, et la somme des délais est de 1 heures. Dans le dessin de droite, les tâches #0 à #7 subissent toutes un délai, donc la somme des délais est de 8h.



$\sum W_i = 1\text{h}$

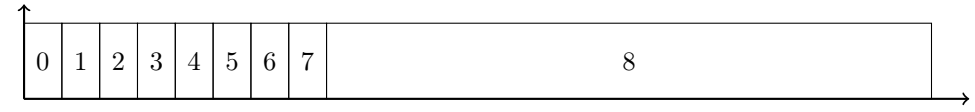


$\sum W_i = 8\text{h}$

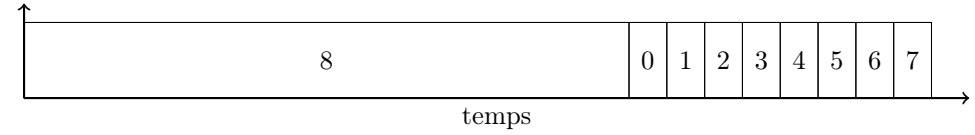
Sur une machine, il vaut mieux mettre les tâches courtes d'abord

Par exemple, considérons ces 9 tâches toutes soumises à $t = 0$. Les 8 premières durent 1h, la dernière dure 16h. Dans le dessin du haut, la somme des délais est de $0+1+2+3+4+5+6+7+8 = 36$ heures. Dans le dessin du bas, toutes les tâches courtes doivent attendre la terminaison de la tâche longue, et la somme des délais est de $0 + 16 + 17 + 18 + 19 + 20 + 21 + 22 + 23 = 180$ heures.

Les durées des premières tâches exécutées se répecute davantage sur la somme des délais que celles des dernières.



$\sum W_i = 36\text{h}$



$\sum W_i = 180\text{h}$

Dernières remarques

Le problème c'est que ces les deux idées précédentes sont incompatibles : que faire des petites tâches longues et des grandes tâches courtes ?

Une dernière observation digne d'intérêt est la suivante : le problème se déroule dans le temps. Il se peut qu'on puisse avoir une approche « diviser pour régner » en décidant de traiter d'abord toutes les tâches soumises tôt, et seulement ensuite celles qui sont soumises tard. De la même manière, s'il y a une longue période sans tâches soumises, cela peut permettre de découper le problème en deux sous-problèmes vraiment indépendants.