Rapport du projet

Federico Canale, Nguyen Phan Nguyet

24 Octobre 2015

Table des matières

1 Introduction

Le but de ce projet est d'étudier le problème d'ordonnancement de tâches sur trois machines en série. Il s'agit d'analyser et d'impléménter deux algorithmes :

- L'algorithme de Johnson qui retourne une solution 2-approchée.
- L'algorithme arborescente qui retourne une solution exacte.
  Dans cette méthode, chaque noeud contient la liste des tâches n'ayant pas encore été traités. C'est sur cette liste que l'on va appliquer l'algorithme de Johson pour obtenir un ordre des tâches à passer sur les machines. La valeur de retour par cette approche sera utilisée comme la born supérieur. La valeur b_1 proposé par l'énoncé sera la borne inférieur.

2 Etudier des algorithmes

2.1 Algorithme approché

1. Toujours 3-approché
   Soit I une instance de notre problème, P une permutation arbitraire des tâches de I. On notera OPT la date de fin d'un ordonnancement optimal de I , P la date de fin du ordonnancement associé à P. Sans optimiser P, on a toujours que P est 3-approché, i.e $P \leq 3OPT$.
   D'abord, $\sum_{i=1}^{n} d_i^A$ est le temps maximum pour que les taches soient finies sur la machine A.
   En suite, $\sum_{i=1}^{n} d_i^A + \sum_{i=1}^{n} d_i^B$ est le temps maximum pour que les tâches soient finies sur la machine B.

Enfin, $\sum_{i=1}^{n} d_i^A + \sum_{i=1}^{n} d_i^B + \sum_{i=1}^{n} d_i^C$ est le temps maximum pour que les tâches soient finies sur la machine C.
Il est donc, $P \leq \sum_{i=1}^{n} d_i^A + \sum_{i=1}^{n} d_i^B + \sum_{i=1}^{n} d_i^C$.
Or, sur chaque machine $j \in \{A, B, C\}, OPT \geq \sum_{i=1}^{n} d_i^j$.
On a donc, $P \leq 3OPT$.

2. 2-approché grâce à l'algorithme de Johnson.
Soit J l'agorithme de Johnson effectué sur 3 machines.
On notera J la date de fin du ordonnancemant associé à l'algorithme J et à nouveau OPT la date associée à la solution optimale.
J_B la date de J sur la machine B.
OPT_B la date de OPT sur la machine B.
Comme l'algorithme de Johson retourne la solution optimale pour le problème de 2 machines, il est donc $J_B \leq OPT_B$.
Il nous faut ajouter à J_B un temps pour traiter les tâches sur la machine C. Cet quantité est au plus $\sum_{i=1}^{n} d_i^C$.
Donc, $J \leq J_B + \sum_{i=1}^{n} d_i^C \leq OPT_B + OPT \leq 2OPT$.

2.2 Méthode exacte

1. La présence de t_B^π dans la formule $b_B^\pi = t_B^\pi + \sum_{i\in\pi} d_i^B + \min_{i\in\pi'}\{d_i^C\}$ est pour assurer que les tâches de π sont finies sur la machine B. Si $t_A^\pi + \min_{i\in\pi'} d_i^A > t_B^\pi$, il est alors sûre qu'à la date de $t_A^\pi + \min_{i\in\pi'} d_i^A$, les tâches de π sont finies sur la machine B.
De même, si $t_C^\pi \leq t_B^\pi + \min_{i\in\pi'} d_i^B$ ou $t_C^\pi \leq t_A^\pi + \min_{i\in\pi'}\{d_i^A + d_i^B\}$, il est certain qu'à la date de $t_B^\pi + \min_{i\in\pi'} d_i^B$ ou de $t_A^\pi + \min_{i\in\pi'}\{d_i^A + d_i^B\}$, les tâches dans π sont finies sur la machine C.

2. Soit P une permutation des tâches commençant par π.
k une tâche dans π'.
On notera
$\pi + 1$ la première tâche dans π', i.e la tâche traitée just après la dernière tâche dans π.
$\pi + 2$ la deuxième tache dans π'.
...
$k - 1$ la tâche dans π' traitée just avant la tâche k.
$|\pi|$ le nombre de tâches dans π.
Alors, les $(|\pi| + k)$ premières tâches dans P sont $\pi, \pi + 1, \pi + 2, .., k - 1, k$.
Le temps pour ces $(|\pi| + k)$ premières tâches traitées sur la machine A est $t_A^\pi + \sum_{i=\pi+1}^{k-1} d_i^A + d_k^A$.
La tâche k doit passer sur les deux dernières machines B et C, ce qui prend au moins un temps $d_k^B + d_k^C$.
Après la tâche k, les tâches qu'il nous reste doivent au moins passer sur la machine C, ce qui prend au moins un temps $\sum_{i>k} d_i^C$.

On a $P \geq (t_A^\pi + \sum_{i=\pi+1}^{k-1} d_i^A + d_k^A) + (d_k^B + d_k^C) + \sum_{i>k} d_i^C$.
Dans l'étape suivante, on effectue un remplacement comme suivant :

— Dans la somme $\sum_{i=\pi+1}^{k-1} d_i^A$, on ne garde que les d_i^A satisfaisant que $d_i^A \leq d_i^C$, et on remplace les autres d_i^A par d_i^C.
— Dans la somme $\sum_{i>k} d_i^C$, on ne garde que les d_i^C satisfaisant que $d_i^C \leq d_i^A$, et on remplace les autres d_i^C par d_i^A.

La nouvelle somme obtenue est de la forme $\sum_{i\in\pi', i\neq k | d_i^A \leq d_i^C} d_i^A + \sum_{i\in\pi', i\neq k | d_i^A \geq d_i^C} d_i^C$.
On obtient l'inégalité à démontrer.

Cela nous propose une nouvelle born inférieur
$b_2 = t_A^\pi + \max_{k\in\pi'}(d_k^A + d_k^B + d_k^C + \sum_{i\in\pi', i\neq k | d_i^A \leq d_i^C} d_i^A + \sum_{i\in\pi', i\neq k | d_i^A > d_i^C} d_i^C)$.

3. De même manière, on se place sur la machine B et considère à nouveau les $(|\pi| + k)$ premières tâches.
Le temps pour ces $(|\pi| + k)$ tâches traitées sur la machine B est $t_B^\pi + \sum_{i=\pi+1}^{k-1} d_i^B + d_k^B$.
La tâche k doit passer sur la dernière machine C, ce qui prend au moins un temps d_k^C.
Après la tâche k, les tâches qu'il nous reste doivent passer sur la machine C, ce qui prend au moins un temps $\sum_{i>k} d_i^C$.

On a $P \geq (t_B^\pi + \sum_{i=\pi+1}^{k-1} d_i^B + d_k^B) + d_k^C + \sum_{i>k} d_i^C$.
On effectue à nouveau une remplacement comme avant pour obtenir une nouvelle born inférieur qui est
$b_3 = t_B^\pi + \max_{k\in\pi'}(d_k^B + d_k^C + \sum_{i\in\pi', i\neq k | d_i^B \leq d_i^C} d_i^B + \sum_{i\in\pi', i\neq k | d_i^B > d_i^C} d_i^C)$.
Ou encore mieux, $b_4 = max\{b_1, b_2, b_3\}$.

4. Dans les réponses 2 et 3, on a eu des minorations avant d'effectuer des remplacements. Pour améliorer la méthode arborescente, on peut modifier ces minorations.
$b_2' = t_A^\pi + \max_{k\in\pi'}(d_k^A + d_k^B + d_k^C + \sum_{i\in\pi', i<k} d_i^A + \sum_{i\in\pi', i>k} d_i^C)$.
$b_3' = t_B^\pi + \max_{k\in\pi'}(d_k^A + d_k^B + d_k^C + \sum_{i\in\pi', i<k} d_i^B + \sum_{i\in\pi', i>k} d_i^C)$.
La nouvelle born inférieur sera $b_5 = max\{b_2', b_3'\}$.