MPRO - Examen d'ordonnancement 2014

Christophe Picouleau, David Savourey

13 janvier 2014

L'examen dure 2h30. Les notes de cours sont autorisées. Les livres ne sont pas autorisés. La clarté de rédaction sera prise en compte dans la notation.

1 Ordonnancements actifs et LO-actifs

Dans tout cet exercice, on se place dans le cadre de problèmes d'ordonnancement à une machine, et l'on suppose que les tâches sont soumises à des dates de disponibilité. Si l'on note F la fonction à minimiser, alors on travaille sur le problème $1|r_i|F$.

Un ordonnancement est semi-actif si aucune tâche ne peut être exécutée plus tôt sans changer l'ordre d'exécution.

Un ordonnancement actif est un ordonnancement où l'on ne peut pas avancer l'exécution d'une tâche sans retarder une autre tâche.

Question 1.1. *Quel lien y a-t-il entre semi-actif et actif ? Les deux sont-ils équivalents ? L'un implique-t-il l'autre ? Justifier votre réponse, si besoin à l'aide d'exemples.*

Question 1.2. *Rappeler la définition d'un critère régulier. Donner un exemple de critère régulier, et un exemple de critère non régulier.*

Question 1.3. *Montrer que la classe des ordonnancements actifs est dominante pour tout critère régulier.*

Question 1.4. *Un ordonnancement non-actif peut-il être optimal pour le problème $1|r_i|\Sigma C_i$? Un ordonnancement non-actif peut-il être optimal pour le problème $1|r_i|\Sigma T_i$? Justifier vos réponses, si besoin à l'aide d'exemple.*

On souhaite maintenant étudier l'opération locale suivante : échanger deux tâches adjacentes dans une séquence. En voici un exemple : soit une séquence σ et i et j deux tâches consécutives (i avant j dans σ). On introduit les notations suivantes :

- Δ : la date à laquelle la machine est libre avant d'exécuter i ;
- $C_k(t)$: la date de fin au plus tôt de la tâche k si elle est exécutée dès que possible à partir de la date t, en respectant sa date de disponibilité ;
- $F_k(t)$: le coût de la tâche k si elle termine son exécution à la date $C_k(t)$.

La Figure 1 illustre la situation dans laquelle nous nous plaçons.

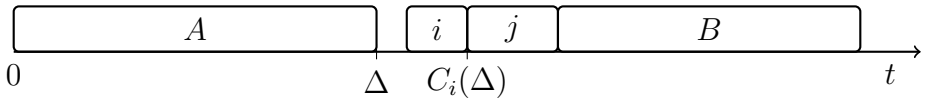

Figure 1 - La séquence σ

Sans connaître les tâches situées après j dans la séquence σ (les tâches de B sur la figure), on peut néanmoins affirmer qu'il est nécessairement intéressant d'échanger i et j si les deux conditions suivantes sont vérifiées :

- (a) l'échange ne peut pas retarder les tâches de B ;
- (b) la somme des coûts des tâches i et j après échange est plus petite qu'avant échange.

Question 1.5. *Caractériser la condition (a) à l'aide des notations introduites.*

Question 1.6. *Caractériser la condition (b) à l'aide des notations introduites.*

On dit qu'un ordonnancement actif pour lequel il n'existe pas de tel échange local améliorant est LO-actif (pour Locally Optimal).

Question 1.7. *Montrer que la classe des ordonnancement LO-actifs est dominante pour les critères réguliers.*

2 Ordonnancement d'un graphe biparti avec petits délais de communication

Le problème consiste à ordonnancer (sans duplication) un graphe de tâches $G = (A \cup B, <)$ sur un nombre non limité de processeurs identiques. G est biparti, c'est-à-dire : $A \cap B = \emptyset$ et $\forall(\mathrm{a}, \mathrm{b}) \in <, \mathrm{a} \in \mathrm{A}$ et $\mathrm{b} \in \mathrm{B}$. Les durées d'exécution des tâches sont supérieures aux délais de communication : $\min_{i \in A \cup B}\{p_i\} \geq \max_{(a,b) \in <}\{c_{ab}\}$. L'objectif est de déterminer un ordonnancement de durée minimale.

Dans un ordonnancement, la date de début d'exécution d'une tâche i est notée t_i et le processeur exécutant i est noté π_i.

Question 2.1. *Montrer qu'il existe un ordonnancement optimal tel que au plus deux tâches sont exécutées sur le même processeur.*

Question 2.2. *Montrer qu'il existe un ordonnancement optimal tel que si deux tâches a et b sont exécutées sur le même processeur alors $(a, b) \in <$.*

Nous rappelons qu'un *couplage* $\mathcal{C}$ de G est un ensemble d'arcs de $<$ tel que pour toute paire d'arcs $(a, b) \in \mathcal{C}, (a', b') \in \mathcal{C}, a \neq a', b \neq b'$. Un couplage $\mathcal{C}$ est *maximal* si et seulement si pour tout arc $(a, b) \notin \mathcal{C}$ il existe $(a, b') \in \mathcal{C}$ ou $(\mathrm{a}', \mathrm{b}) \in \mathcal{C}$, c'est-à-dire $\mathcal{C} \cup \{(a, b)\}$ n'est pas un couplage.

Question 2.3. *Montrer qu'il existe un ordonnancement optimal tel que les arcs de G correspondant aux couples de tâches exécutées sur un même processeur forment un couplage maximal de G.*

Nous notons $\theta_{ab} = p_a + p_b + c_{ab}, (a, b) \in <$. Nous indiçons les arcs de G de sorte que $\theta_1 \geq \theta_2 \geq \cdots \geq \theta_m$. Soit l'algorithme suivant qui donne un couplage maximal de G :

$\mathcal{C} \leftarrow \emptyset$
pour $i = 1$ à m faire
  si l'arc i n'est pas adjacent à un arc de $\mathcal{C}$ alors $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$

Question 2.4. *Montrer qu'il existe un ordonnancement optimal tel que $\pi_a = \pi_b$ pour tout arc $(a, b) \in \mathcal{C}$.*

Question 2.5. *Donner un algorithme polynomial qui calcule un ordonnancement optimal.*