MPRO - Examen d'ordonnancement 2013

Christophe Picouleau, David Savourey

14 janvier 2013

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 Une formulation 0-1

Nous nous proposons de travailler sur le problème $1|prec|\sum w_i C_i$. Ce problème est NP-difficile. Nous allons formuler ce problème en un programme linéaire en nombres entiers.

Tout d'abord, nous utiliserons les conventions suivantes dans la suite de cet exercice. Les jobs seront indicés de 1 à n. Nous noterons classiquement p_i la durée du job i, et w_i son poids. Par ailleurs, les précédences seront données par une matrice a à valeur dans $\{0,1\}$, où $a_{ij} = 1$ le job i doit précéder le job j dans tout ordonnancement réalisable.

Nous allons utiliser les variables binaires suivantes pour formuler notre problème en un PLNE :

$$x_{ij} = \begin{cases} 1 \text{ si le job } i \text{ précède le job } j, \\ 0 \text{ sinon} ; \end{cases}$$

où $i \in [1, n]$ et $j \in [1, n]$. Par ailleurs, nous imposerons que $\forall i \in [1, n], x_{ii} = 1$.

Pour qu'une affectation des variables x_{ij} corresponde à un ordonnancement réalisable, il faut que :

– la relation binaire induite par les x_{ij} définisse une relation d'ordre totale, c'est-à-dire que cette relation doit :

  - (1) être complète (“i précède j” ou bien “j précède i”) ;
  - (2) être sans circuit.
- (3) cet ordre respecte les précédences données en entrée par la matrice a.

Question 1.1. Soit une affectation des variables x_{ij}. Montrer que si la relation induite par les variables x_{ij} est transitive, alors elle est sans circuit.

Question 1.2. Formuler la quantité C_i (la date de fin du job i) comme une combinaison linéaire des variables x_{ij}. Vous pouvez bien sûr utiliser les données du problème.

Question 1.3. Proposer un PLNE sur les variables x_{ij} qui modélise le problème $1|prec|\sum w_iC_i$. Vous veillerez à écrire les contraintes de types (1), (2) et (3) exprimées plus haut.

Question 1.4. Afin de calculer des bornes inférieures, on relâche la contrainte interdisant les circuits. Proposer un algorithme polynomial qui résolve ce sous-problème. Justifier brièvement.

2 Règle de McNaughton

Nous souhaitons désormais étudier le problème $P|pmtn|C_{max}$. Les jobs seront indicés de 1 à n. Nous noterons p_i la durée du job i et C_i sa date de fin.

Question 2.1. Soit un problème à 3 machines avec les jobs suivants :

J_i	J_1	J_2	J_3	J_4	J_5
p_i	3	4	5	5	6

Donner une solution optimale pour cette instance. Dessiner un diagramme de Gantt de cette solution. Vous devrez justifier l'optimalité en utilisant un argument très simple. À partir de cet argument, proposer une borne inférieure valide (atteinte dans le cas de cette instance) pour le problème $P|pmtn|C_{max}$.

Question 2.2. Soit un problème à 4 machines avec les jobs suivants :

J_i	J_1	J_2	J_3	J_4	J_5	J_6	J_7
p_i	7	2	4	3	3	2	3

Donner une solution optimale pour cette instance. Dessiner un diagramme de Gantt de cette solution. Vous devrez justifier l'optimalité en utilisant un argument très simple. À partir de cet argument, proposer une borne inférieure valide (atteinte dans le cas de cette instance) pour le problème $P|pmtn|C_{max}$.

Question 2.3. À partir des deux exemples précédents, donner une formule analytique de la valeur optimale du problème $P|pmtn|C_{max}$. Justifier votre réponse. Enfin, proposer un algorithme en temps polynomial permettant de résoudre ce problème.

3 Ordonnancement avec communications

On considère un problème d'ordonnancement avec communications sous l'hypothèse *petits délais de communication*, i.e. $\max\{c_{ij}\} \leq \min\{p_i\}$.

On note $G = (T, A)$ le graphe de précédence et le *nombre de processeurs est illimité*.

Le critère à minimiser est la *durée de l'ordonnancement*, i.e. $\min_{\mathcal{O}} \max_{i \in T}\{C_i\}$ où $\mathcal{O}$ est l'ensemble des ordonnancements réalisables.

Deux tâches de précédence sont dites *indépendantes* s'il n'existe pas de chemin les reliant dans G.

Question 3.1. Montrer qu'il existe un ordonnancement optimal tel que pour toute paire de tâches indépendantes $\{i, j\}$, on a $\pi_i \neq \pi_j$.