MPRO - Examen d'ordonnancement 2015/2016

Christophe Picouleau, David Savourey

11 janvier 2016

L'examen dure 3h. Les notes de cours sont autorisées. Les livres ne sont pas autorisés. La clarté de rédaction sera prise en compte dans la notation.

1 Le problème $1|r_i, p_i = p| \sum w_i U_i$

[*Ne paniquez pas si vous trouvez cette partie difficile, j'en tiendrai compte dans la notation.* **May the force be with you.**]

On souhaite résoudre le problème d'ordonnancement à une machine suivant : on dispose de n jobs qui sont tous de durée égale, notée p. Chaque job est soumis à une date de disponibilité r_i et à une date d'échéance d_i. On souhaite minimiser la somme des poids des jobs en retard ($\sum w_i U_i$).

On supposera, sans perte de généralité, que l'on a renuméroté les jobs de sorte que $d_1 \leq d_2 \leq \ldots \leq d_n$.

Soit $T = \{r_j + lp \mid j = 1, \ldots, n \ ; l = 0, \ldots, n-1\}$.

Question 1.1. *Combien l'ensemble T contient-il de valeurs au maximum ?*

Question 1.2. *Montrer qu'il existe un ordonnancement optimal où tous les jobs commencent à des dates qui appartiennent à l'ensemble T.*

Dans toute la suite, nous considérons que des ordonnancements qui sont tels que tous les jobs commencent à des dates qui appartiennent à l'ensemble T.

Dès lors qu'un job est en retard, son coût est fixe (c'est son poids). Le problème revient donc à trouver un sous-ensemble de poids maximal de jobs que l'on peut ordonnancer à l'heure. Dans la suite, nous ne considérerons que les jobs à l'heure dans les ordonnancements : quand nous parlerons d'un job ordonnancé, cela impliquera qu'il est à l'heure. D'autre part, nous chercherons à maximiser la somme des poids des jobs à l'heure (plutôt que de minimiser la somme des poids des jobs en retard, mais c'est évidemment équivalent).

Question 1.3. *Soit O un ordonnancement réalisable. Soient $i < j$. Supposons que j commence à la date s, qu'il est ordonnancé avant i et que $r_i \leq s$. Montrer que l'on peut permuter i et j pour former un nouvel ordonnancement réalisable de même coût que O.*

Question 1.4. *Supposons qu'il existe un ordonnancement optimal où le job de date d'échéance maximale (ici n) est à l'heure. Montrer qu'il existe un ordonnancement optimal où :*

- *le job n commence à la date s' ;*
- *les jobs qui sont ordonnancés avant n ont une date de disponibilité strictement inférieure à s' ;*
- *les jobs qui sont ordonnancés après n ont une date de disponibilité supérieure ou égale à s'.*

Ce constat est pour nous très intéressant, car il nous permet de décomposer notre problème.

Afin de détailler cette décomposition, nous devons définir une version paramétrée de notre problème. *[Soyez bien attentif ici, c'est un peu technique mais pas compliqué.]*

On note tout d'abord $U_k(s, e)$ l'ensemble des jobs j tels que $1 \leq j \leq k$ et $s \leq r_j < e$.

On note ensuite $P_k(s, e)$ le problème qui consiste à trouver un ordonnancement optimal en ne considérant que les jobs de $U_k(s, e)$ et où l'on ne peut ordonnancer des jobs qu'entre les dates $s + p$ et e. Dans cette notation, le problème initial peut s'écrire $P_n(min(T) - p, max(T))$.

On note $W_k^*(s, e)$ la valeur optimale du problème $P_k(s, e)$. Lorsqu'aucun des jobs de $U_k(s, e)$ ne peut être ordonnancé entre $s+p$ et e, ou que l'ensemble des jobs possibles est vide ($k = 0$), on aura simplement $W_k^*(s, e) = 0$.

Nous rappelons ici que nous ne considérons que les jobs à l'heure dans les ordonnancements que nous recherchons. Autrement dit, un ordonnancement

optimal pour le problème $P_k(s, e)$ est composé d'un sous-ensemble de jobs de $U_k(s, e)$ qui sont à l'heure.

Revenons-en à nos moutons. Supposons que nous devions résoudre le problème $P_k(s, e)$. Nous devons considérer deux cas, suivant que le job dont la date d'échéance est la plus grande (donc k), est en retard (cas (a)) ou à l'heure (cas (b)).

Dans le cas (a), cela revient à résoudre le problème en retirant le job k du problème.

Question 1.5. *En supposant que l'on se trouve dans le cas (a), exprimer $W_k^*(s, e)$ en fonction d'un ou plusieurs autres $W_{\_}^*(\_, \_)$.*

Dans le cas (b), nous allons utiliser ce que nous avons démontré à la question 1.4. Comme nous travaillons désormais sur le problème $P_k(s, e)$, il nous faut simplement considérer que le job de date d'échéance maximale est désormais k et non plus n : si k commence à la date s', on sait que l'on peut considérer que :

- tous les jobs avant k auront une date de disponibilité strictement inférieure à s' ;
- tous les jobs après k auront une date de disponibilité supérieure ou égale à s'.

Question 1.6. *En supposant que l'on se trouve dans le cas (b), exprimer $W_k^*(s, e)$ en fonction d'un ou plusieurs autres $W_{\_}^*(\_, \_)$ (en utilisant par exemple s') et de w_k.*

Nous avons exprimé le cas (b) en fonction s', que nous ne connaissons pas. Il nous faut donc tester toutes les valeurs possibles pour s' dans le calcul de $W_k^*(s, e)$, et retenir la plus avantageuse. Toutefois, nous savons que :

- s' doit appartenir à T ;
- $s' \in [s + p, e]$;
- s' est plus grand que r_k ;
- s' est plus petit que $d_k - p$.

Question 1.7. *Combien y a-t-il de valeurs possibles pour s' ? Nous demandons ici un ordre de grandeur.*

Question 1.8. *Lors de la résolution du problème $P_k(s, e)$, dans quel cas est-on certain que seul le cas (b) est possible ?*

Question 1.9. *Proposer un schéma de programmation dynamique permettant de calculer toutes les valeurs de* $W_k^*(s, e)$, $k \in 1, 2, \ldots, n$ *et* $s, e \in T$.

Question 1.10. *En fonction du schéma proposé, quelle est la complexité de calculer* $W_n^*(min(T) - p, max(T))$ *?*