PROBLEMES D'ORDONNANCEMENT À DURÉES ÉGALES

J. CARLIER

Dans cet article, nous étudions le problème de l'ordonnancement sur une machine de tâches de même durées disponibles et devant être achevées à des dates différentes.

Ce problème a longtemps été ouvert.

Dans ce papier, nous présentons une méthode pour le résoudre.

Nous montrons également déterminer un ordonnancement ayant un nombre minimal de tâches en retard.

Enfin nous généralisons à plusieurs machines identiques.

1. INTRODUCTION

Dans cet article, nous considérons l'ordonnancement de n tâches; une tâche i est disponible à la date a_i , doit être achevée à la date b_i et a une durée D (les durées sont égales) (1). Nous supposons en outre que toutes les tâches doivent passer sur une même machine; en conséquence, deux tâches devront s'exécuter dans des intervalles de temps différents.

Ce problème a été résolu par B.Simons /6/ -- /10//11/ et par J. Carlier /2/; cette dernière méthode, que nous présentons, est basée - sur une relation de préordre que nous definissons et dont nous étudions les propriétés. Nous montrons comment l'utiliser pour résoudre ce problème.

Ensuite, nous étendons la méthode au problème de la minimisation du nombre de tâches en retard.

Enfin, nous généralisons au cas de plusieurs machines identiques.

2. RELATION DE PREORDRE

Dans ce paragraphe, nous démontrons deux propositions qui nous servirons à justifier les algorithmes polynomiaux permettant de traiter les problèmes à durées égales.

I désignera un ensemble fini sur lequel est défini un préordre total ρ (rappelons qu'un préordre total ρ est une relation réflexive et transitive telle que pour toute paire d'éléments de I , on ait $i \rho j$ ou $j \rho i$

Dans la suite C et E sont deux parties de I de cardinaux respectifs p et r.

2.1. Définitions.

On dit que C est préférable à E et on note -- C P E si le cardinal de C est plus grand que le cardinal de E et s'il existe des ordres - $\{c_1, c_2, \ldots, c_p\}$ et $\{e_1, e_2, \ldots, e_r\}$ de C et E tels que:
$c_1 \rho e_1$, $c_2 \rho e_2, \ldots, c_r \rho e_r$.

- Jacques Carlier - Institut de Programmation - Université Paris VI - 75012 Paris

- Article rebut el Desembre del 1981

Exemples

$I = \{1,2,\ldots,10\}$, $C = \{2,4,7,8,10\}$
$E = \{3,6,7,9\}$, $\rho = <$; nous ordonnons C et E: $c_1 = 2$, $c_2 = 4$, $c_3 = 7$, $c_4 = 8$, $c_5 = 10$; --- $e_1 = 3$, $e_2 = 6$, $e_3 = 7$, $e_4 = 9$; on a $p = 5$ et $r = 4$, alors $p > r$; de plus: $c_1 \rho e_1, c_2 \rho e_2, c_3 \rho e_3, c_4 \rho e_4$, donc C P E.

2.2. Propriétés caractéristiques

Nous énonçons maintenant deux propriétés équivalentes à la définition et montrons que P -- est un préordre sur $\mathscr{P}(I)$.

Propriété 1

Ordonnons C et E de telle sorte que $c_{i_1} \rho c_{i_2} \rho \ldots \rho c_{i_p}$ et $e_{j_1} \rho e_{j_2} \rho \ldots \rho e_{j_r}$ (cela est toujours possible car ρ est un préordre total).

Pour que C soit préférable à E , il -- faut et il suffit que $p > r$ et ----- $c_{i_1} \rho e_{j_1}, c_{i_2} \rho e_{j_2}, \ldots, c_{i_r} \rho e_{j_r}$.

Démonstration

La condition est suffisante (prendre les ordres $\{c_{i_1}, c_{i_2}, \ldots, c_{i_p}\}$ et $\{e_{j_1}, e_{j_2}, \ldots \ldots e_{j_r}\}$).

Montrons par induction sur r qu'elle est également nécessaire; soient C et E deux parties de I telles que C soit préférable à E ; on a par définition deux ordres $\{c_1, c_2, \ldots, c_p\}$ - et $\{e_1, e_2, \ldots, e_r\}$ tels que: $c_1 \rho e_1$, ----- $c_2 \rho e_2, \ldots, c_r \rho e_r$; en particulier $c_{j_1} \rho e_{j_1}$; envisageons trois cas:

-1 - $i_1 = j_1$

On conclut en appliquant l'hypothèse d'in-- duction à $C - \{c_{i_1}\}$ et à $E - \{e_{j_1}\}$.

-2 - $i_1 > r$

On a $c_{i_1} \rho c_{j_1}$, et , $c_{j_1} \rho e_{j_1}$; donc par-- transitivité $c_{i_1} \rho e_{j_1}$; on se ramène au -- cas précédent en remplaçant la relation ----- $c_{j_1} \rho e_{j_1}$ par $c_{i_1} \rho e_{j_1}$, c'est à dire en permutant c_{i_1} et c_{j_1} dans l'ordre $\{c_1, c_2, \ldots, c_p\}$

-3- $i_1 < r$ et $i_1 \neq j_1$

On a: $c_{i_1} \rho c_{j_1}$, $c_{j_1} \rho e_{j_1}$, $e_{j_1} \rho e_{i_1}$; --- donc on peut remplacer les deux relations --- $c_{i_1} \rho e_{i_1}$ et $c_{j_1} \rho e_{j_1}$ par $c_{i_1} \rho e_{j_1}$ et ----- $c_{j_1} \rho e_{i_1}$; c'est à dire en permutant c_{i_1} et c_{j_1} dans l'ordre $\{c_1, c_2, \ldots, c_p\}$. On con-- clut en appliquant l'hypothèse d'induction à $C - \{c_{i_1}\}$ et $E - \{e_{j_1}\}$.

Corollaire

P est un préordre sur $\mathscr{P}(I)$.

Démonstration

P est évidemment réflexive; la transitivité - résulte de la propriété 1.

Propriété 2

Pour que C soit préférable à E ,il faut et il suffit que $(C - C \cap E)$ soit préférable à $(E - C \cap E)$.

Démonstration

La condition est évidemment suffisante; montrons par induction sur r qu'elle est également nécessaire; soient donc C et E deux parties de I telles que C soit préférable à E ; d'après la propriété 1, on peut indicer C et E de telle sorte que:
$c_1 \rho c_2 \rho \ldots \rho c_p$, $e_1 \rho e_2 \rho \ldots \rho e_r$, ------ $c_1 \rho e_1 \ldots c_r \rho e_r$.

Envisageons trois cas:

-1 - $c_1 \in C \cap E$

Alors $c_1 = e_l$; si $l = 1$, on applique l'hypo-- thèse d'induction à $C - \{c_1\}$ et à $E - \{e_l\}$ -- sinon: on a $c_l \rho e_l$, $e_l = c_1$ et $c_1 \rho e_1$;donc $c_l \rho e_1$, par transitivité; on remplace les - deux relations $c_1 \rho e_1$ et $c_l \rho e_l$ par $c_1 \rho e_l$ et $c_l \rho e_1$ avant d'appliquer l'hypothèse d'induction à $C - \{c_1\}$ et $E - \{e_l\}$.

-2 - $c_1 \notin C \cap E$ et $e_1 \notin C \cap E$

Alors $e_l = e_m$; si $m > r$, on remplace $c_1 \rho e_1$ par $c_m \rho e_1$ et on applique l'hypothèse d'induction a $C - \{c_m\}$ et $E - \{e_1\}$; - sinon $m \leq r$, on remplace $c_1 \rho e_1$, et $c_m \rho e_m$, par $c_m \rho e_1$ et $c_1 \rho e_m$, et, on applique l'hypothèse d'induction a $C - \{c_m\}$ $E - \{e_1\}$.

-3 - $c_1 \notin C \cap E$ et $e_1 \notin C \cap E$

On applique l'hypothèse d'induction à $C - \{c_1\}$ et $E - \{e_1\}$.

2.3. PROPOSITIONS

Proposition 1

Si C P E, $u \notin C$, $v \notin E$, et $u \rho w \; \forall w \in \bar{C}$, alors $\{u\} \cup C$ est préférable à $\{v\} \cup E$.

Démonstration

D'après la proprieté 2, on a: $c_1 = e_1 \ldots c_s = e_s$, $c_{s+1} \rho e_{s+1}, \ldots, c_r \rho e_r$; posons

$C' = \{c_1, \ldots, c_s\} = \{e_1, \ldots, e_s\} = E \cap C$

$C'' = \{c_{s+1} \ldots c_r\}$

$C''' = \{c_{r+1} \ldots c_p\}$. Envisageons quatre cas:

-1 - $v \in \bar{C}$

Par hypothèse $u \rho v$, donc $(\{u\} \cup C)$ P $(\{v\} \cup E)$.

-2 - $v \in C'$

C'est impossible puisque $v \notin E$.

-3 - $v \in C''$

$v = c_l$ et $c_l \rho e_l$; comme $e_l \notin C$, on a $u \rho e_l$; on remplace $c_l \rho e_l$ par $c_l \rho v$ et $u \rho e_l$; donc $(\{u\} \cup C)$ P $(\{v\} \cup E)$.

-4 - $v \in C'''$

On ajoute $v \rho v$; donc $(\{u\} \cup C)$ P $(\{v\} \cup E)$.

Proposition 1 bis

Si C P E, $u \notin C$, $v \notin E$, et $\exists$ $v_o : v_o \rho v$; $u \rho w$ pour tout w de $\bar{C}$ satisfaisant $v_o \rho w$; alors $\{u\} \cup C$ est préférable à $\{v\} \cup E$.

Proposition 2

Si C P E, alors $\bar{E}$ P $\bar{C}$.

Démonstration

Comme conséquence directe de la propriété 2,- on a:
$(C - C \cap E)$ P $(E - C \cap E)$. Or : $(C - C \cap E) = (\bar{E} - \bar{E} \cap \bar{C})$ et $(E - C \cap E) = (\bar{C} - \bar{E} \cap \bar{C})$; - donc :$(\bar{E} - \bar{E} \cap \bar{C})$ P $(\bar{C} - \bar{E} \cap \bar{C})$; on conclut: $\bar{E}$ P $\bar{C}$.

2.4. ORDRE

Théorème

Lorsque la relation de préordre est un ordre total, la relation de préférence P confère à $\mathcal{P}(I)$ une structure de treillis distributif. Nous noterons l'ordre large $\leq$, l'ordre strict $<$.

Démonstration

P est antisymétrique

Posons $C' = C - C \cap E$, $E' = E - C \cap E$; si C P E et E P C, on a d'après la propriété 2: C' P E' et E'P C';de C' P E',on déduit facilement que le plus petit élément de E' est strictement plus petit que le plus petit élément de C' ; de E' P C', on déduit la proposition inverse; il en résulte $E' = C' = \phi$ et $E = C$ (récurrence immédiate).

P est un ordre

P est un préordre antisymétrique, donc un ordre.

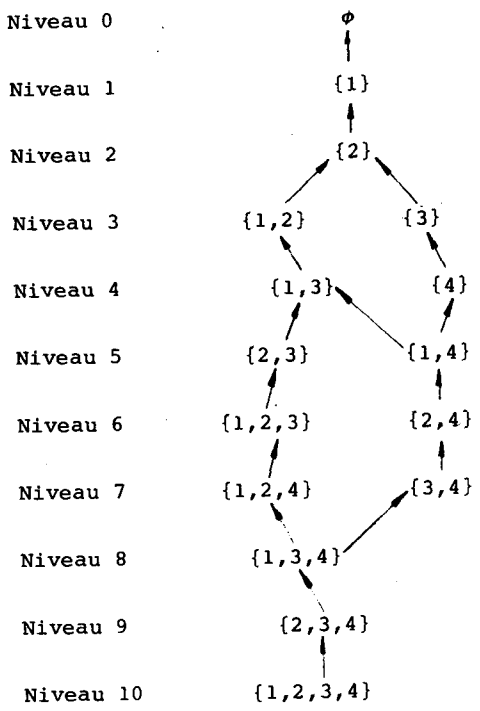
(℘(I),P) est un treillis

Soient E et F deux parties de I : E = $= \{e_1, e_2, \ldots, e_r\}$, F = $\{f_1, f_2, \ldots, f_p\}$; on peut supposer: $p \geq r$, $e_1 < e_2 < \ldots < e_r$ et $f_1 < f_2 \ldots < f_p$. Nous allons montrer que $E \wedge F = \{\inf(e_1, f_1), \ldots, \inf(e_r, f_r), f_{r+1}, \ldots, f_p\}$ et $E \vee F = \{\sup(e_1, f_1), \ldots, \sup(e_r, f_r)\}$. Il est immédiat de vérifier que $(E \wedge F)$ P E et $(E \wedge F)$ P F; il reste donc à montrer que $E \wedge F$ est le plus grand des minorants. Soit $G = \{g_1, g_2, \ldots, g_q\}$ avec G P E, G P F et $g_1 < g_2 < \ldots < g_q$; pour G P E, la propriété 1 s'écrit: $g_1 \leq e_1 \ldots g_r \leq e_r$; pour G P F, la propriété 1 s'écrit: $g_1 \leq f_1, \ldots, g_p \leq f_p$ donc $g_1 \leq \inf(e_1, f_1) \ldots g_r \leq \inf(e_r, f_r)$, $g_{r+1} \leq f_{r+1}, \ldots, g_p \leq f_p$. Ce qui montre que G P(E F) donc E F est le plus grand des minorants. On prouve de même que E F est le plus petit des majorants. ((I), P) est un treillis.

Le treillis est distributif

Pour trois réels $e, f,$ et g, on a:
$(\max(\min(e,f),g) = \min(\max(e,g),\max(f,g))$
de ceci, il résulte la distributivité de $\vee$ sur $\wedge$.

Relation ≥



Un cas particulier intéressant est celui où $\rho = \geq$ et $I = \{1,2,\ldots,n\}$. Le diagramme de Hasse de la relation d'ordre est alors un graphe à couches; les successeurs d'un sommet $\{e_1, e_2, \ldots, e_r\}$ sont les sommets:

- $\{e_1, e_2, \ldots, e_{i-1}, e_i{}^{-1}, \ldots, e_r\}$ pour $e_i{}^{-1} \neq e_{i-1}$ et $e_i \neq 1$.
- $\{e_2, \ldots, e_r\}$ si $e_1 = 1$

Figure 1 $|I| = 4$

3. DETERMINATION D'UN ORDONNANCEMENT REALISABLE

Soit I l'ensemble des tâches; notre but est de trouver une solution, c'est à dire un ordonnancement $\mathcal{T} = (t_i)_{i \in I}$ respectant les contraintes:

(i) $t_i \geq a_i$ .L'opération i ne peut débuter avant la date a_i.

(ii) $t_i + D \leq b_i$ .L'operation i doit être achevée à la date b_i

(iii) $t_i < t_j \Rightarrow t_j \geq t_i + D$. Deux opérations ne peuvent être exécutées simultanément.

Nous allons présenter l'algorithme et l'illustrer à l'aide d'un exemple à 6 tâches dont les données sont les suivantes: I = 1,2,3,4,5,6, ; $a_1 = 0$, $a_2 = 2$, $a_3 = 7$,

$a_4 = 9$, $a_5 = 10$, $a_6 = 24$; $b_1 = 32$, $b_2 = 35$
$b_3 = 22$, $b_4 = 20$, $b_5 = 23$, $b_6 = 30$, D = 5.

3.1. Définitions

Nous donnons pour commencer les définitions d'ordonnancement partiel, de délai, d'ordonnancement actif et d'ordonnancement x-actif.

Un ordonnancement partiel d'un sous-ensemble U de I est un ensemble de dates $\mathcal{T} = (t_u)_{u \in U}$ tel que:

(i) $t_u \geq a_u$

(ii) $t_u + D \leq b_u$

(iii) $t_u < t_{u'} \Rightarrow t_{u'} \geq t_u + D$

Le délai d'un ordonnancement partiel $\mathcal{T}$ est la date où la machine devient disponible: $y = \max_{u \in U} (t_u + D)$.

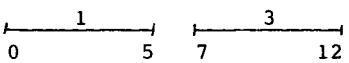
Un ordonnancement actif est un ordonnancement partiel satisfaisant la condition *(iv)*:

(iv) il n'existe pas de tâche w de $\bar{U}$ avec $b_w < y$

Un ordonnancement x-actif est un ordonnancement actif de delai inférieur à x.

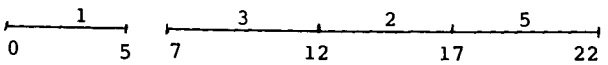
Dans la suite de ce paragraphe, nous nous interesserons uniquement aux ordonnancements actifs; nous noterons $T(x)$ l'ensemble des ordonnancements x-actifs et $T(\infty)$ l'ensemble des ordonnancements actifs.

Exemple 1



Dans ce premier exemple; U = {1,3} , $t_1 = 0$ $t_3 = 7$; le délai de l'ordonnancement est 12. Il est actif car il n'y a pas de tâches w de $\bar{U}$ avec $b_w < 12$. L'ordonnancement est x-actif pour $x \geq 12$.

Exemple 2



Dans ce deuxième exemple, U = {1,2,3,5 } , $t_1 = 0$, $t_3 = 7$, $t_2 = 12$, $t_5 = 17$. Le delai est 22. Cet ordonnancement n'est pas actif car b_4 est strictement inférieur à 22.

3.2. Principe de l'algorithme

Nous allons définir un préordre sur l'ensemble des ordonnancements actifs tel que les éléments maximaux pour ce préordre soient, lorsqu'elles existent les solutions du problème.

Nous construisons en cours d'algorithme pour chaque valeur de x un element $\mathcal{T}^x$ maximal parmi les ordonnancements x-actifs, donc pour x suffisamment grand, nous aurons les éléments maximaux du préordre.

3.3. Préordre

Nous définissons sur I la relation ρ par : $i \rho j$ si $b_i \leq b_j$. A ce préordre sur I est associé le préordre P sur $\mathcal{P}(I)$ (Cf 2), U P V si:

- $p = |U| \geq r = |V|$;
- $b_{u_1} \leq b_{v_1}, \ldots, b_{u_r} \leq b_{v_r}$; des que:
  $b_{u_1} \leq b_{u_2} \leq \ldots \leq b_{u_p}$ et
  $b_{v_1} \leq b_{v_2} \leq \ldots \leq b_{v_r}$.

Par abus d'écriture, nous écrirons $\mathcal{T}$ P $\mathcal{T}'$ si l'ensemble des tâches U ordonnancées par $\mathcal{T}$ est préférable à l'ensemble V des tâches ordonnancées par $\mathcal{T}'$ .P est alors un préordre sur l'ensemble des ordonnancements actifs.

Intuitivement $\mathcal{T}$ est préférable à $\mathcal{T}'$ si le nombre de tâches ordonnancées par $\mathcal{T}$ est plus grand que le nombre de tâches ordonnancées par $\mathcal{T}'$ et si ces tâches sont plus urgentes.

Théorème

S'il existe des solutions au problème, l'ensemble de ces solutions est l'ensemble des éléments maximaux de $T(\infty)$.

Démonstration

Toute solution $\mathcal{T} = (t_u)_{u \in U}$ est un ordonnancement actif puisque U = I, $\bar{U} = \phi$;c'est un élément maximal pour le préordre P puisque U = I. Réciproquement, tout ordonnancement maximal $\mathcal{T}' = (t_v)_{v \in V}$ est préférable en particulier à une solution $\mathcal{T} = (t_u)_{u \in U}$ on a V P U, U = I , donc V = I; $\mathcal{T}'$ est donc une so-

lution.

3.4. Algorithme

Dans l'algorithme ci-dessous :

- K_x est l'ensemble des tâches qui peuvent être ordonnancées avant la date x : $K_x = \{i \in I / a_i + D \leq x\}$;
- U_x est l'ensemble des tâches ordonnancées par $\mathcal{T}^x$.
- $H = K_x - U_{x-D}$ est l'ensemble des tâches non ordonnancées par $\mathcal{T}^{x-D}$, qui peuvent être ordonnancées avant la date x.
- m est la tâche la plus urgente de H.
- $[\mathcal{T}^{x-D}+m]$ est l'ordonnancement obtenu à partir de $\mathcal{T}^{x-D}$ en exécutant l'opération m le plus tôt possible c'est à dire à la date $t_m = max(a_m, y)$ où y est le délai de $\mathcal{T}^{x-D}$.

Algorithme

(i) Initialisations

$\mathcal{T}^x = \phi$ pour: $x = \inf_{i \in I} a_i$ à $x = [(\inf_{i \in I} a_i) + D - 1]$.

(ii) Calcul de $\mathcal{T}^x$

Soient $K_x = \{i \in I / a_i + D \leq x\}$, $H = K_x - U_{x-D}$; si, $H \neq \phi$; soit m : $b_m = \min_{h \in H} b_h$.

Si $H = \phi$, poser : $\mathcal{T}^x = \mathcal{T}^{x-D}$; et aller en *(iii)*.

Si $[\mathcal{T}^{x-D}+m]$ est actif, poser $\mathcal{T}^x = [\mathcal{T}^{x-D}+m]$; et aller en *(iii)*.

Sinon $[\mathcal{T}^{x-D}+m]$ n'est pas actif; poser $\mathcal{T}^x = \mathcal{T}^{x-1}$.

(iii) Incrémentation de x

Si $x \neq \max_{i \in I} b_i$, poser $x = x+1$; et aller en *(ii)*

Si $U_x \neq I$, il n'y a pas de solution sinon $\mathcal{T}^x$ est une solution; l'écrire.

3.5. Traitement de l'exemple

- $x = 0,1,2,3,4$: $\mathcal{T}^x = \phi$; $U_x = \phi$.

- $x = 5$

$K_5 = \{i \in I / a_i + D \leq 5\} = \{1\}$; $U_{x-D} = \phi$; $H = K_5 - U_0 = \{1\}$; $m = 1$; $[\mathcal{T}^0 + m]$ est actif donc $\mathcal{T}^5 = [\mathcal{T}^0 + m]$.

- $x = 6$

Voir $x = 5$

- $x = 7,8,9$

$\mathcal{T}^7 = \mathcal{T}^8 = \mathcal{T}^9 = \mathcal{T}^5$ (car 1 est plus urgente que 2).

- $x = 10$

$K_{10} = \{1,2\}$; $U_{10-5} = U_5 = \{1\}$; $H = K_{10} - U_5 = \{2\}$; $m = 2$; $[\mathcal{T}^{x-D}+m]$ est actif; d'où $\mathcal{T}^{10} = [\mathcal{T}^{x-D}+m]$.

- $x = 11$

$\mathcal{T}^{11} = \mathcal{T}^{10}$

- $x = 12$

$K_{12} = \{1,2,3\}$; $U_{12-5} = U_7 = \{1\}$; $H = K_{12} - U_7 = \{2,3\}$; $m = 3$ (car $b_2 = 35$ et $b_3 = 22$); $[\mathcal{T}^7+3]$ est actif, d'où :

$\mathcal{T}^{12} = [\mathcal{T}^7+3]$.

etc... Les différents ordonnancements construits par l'algorithme sont rapportés sur la figure 2, une solution du problème est $\mathcal{T}^{34}$.

3.6. Justification de l'algorithme

Nous supposerons que $\inf_{i \in I} (a_i+D)$ est inférieure à $\inf_{i \in I} b_i$; sinon, il est évident que le problème n'a pas de solution.

Théorème

Pour tout x, de $x = \inf_{i \in I} a_i$ à $x = \sup_{i \in I} b_i$ l'ordonnancement $\mathcal{T}^x$, construit par l'algorithme, est x-actif et est préférable à tous les ordonnancements x-actifs.

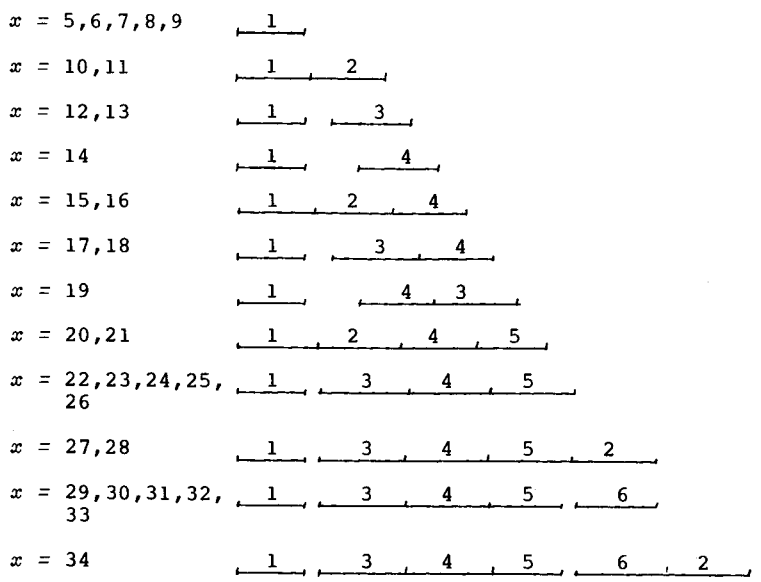


Figure 2: Ordonnancements construits par l'algorithme.

Démonstration

Les deux propriétés sont vraies pour $x < [\inf_{i \in E} a_i + D]$ car alors $K_x = \phi$ et donc $T(x) = \phi$.

Supposons les propriétés vraies pour $z \leq x-1$; nous allons montrer qu'elles restent vraies pour $\mathcal{T}^x$.

Si $H = \phi$, $K_x \subseteq U_{x-D}$; comme $K_{x-D} \subseteq K_x$ et $U_{x-D} \subseteq K_{x-D}$ on a $U_{x-D} \subseteq K_x$; donc $K_x = U_{x-D}$. Toutes les tâches x-exécutables sont ordonnancées dans $\mathcal{T}^{x-D}$; donc $\mathcal{T}^{x-D}$ est un element maximal de $T(x)$. De plus $\mathcal{T}^{x-D} \; T(x-D)$ et $T(x-D) \subseteq T(x)$. entraîne $\mathcal{T}^{x-D} \in T(x)$.

Si $H \neq \phi$, posons $\mathcal{T}' = [\mathcal{T}^{x-D}+m]$; soient $\mathcal{T}$ un élément quelconque de $T(x)$, l la dernière tâche ordonnancée par $\mathcal{T}$, $\mathcal{T}-l$ l'ordonnancement obtenu à partir de $\mathcal{T}$ en retirant la tâche l, U l'ensemble des tâches ordonnancées par $\mathcal{T}$.

Si $\mathcal{T}'$ est actif nous allons montrer que $\mathcal{T}' \; P \; \mathcal{T}$ en utilisant la proposition 1 démontréeau paragraphe précédent. Posons: $I = K_x$, $C = U_{x-D}$, $E = U - \{l\}$, $u = m$, $v = l$; les conditions de la proposition 1 sont réalisées : m est l'élément maximal de $H = [K_x - U_{x-D}]$ et $l \in E$; en conséquence $[U_{x-D}+m] \; P \; U$; $\mathcal{T}'$ est donc préférable à tout élément $\mathcal{T}$ de $T(x)$. Donc si $\mathcal{T}'$ est actif, il est x-actif et maximal parmi les x-actifs.

Si $\mathcal{T}'$ n'est pas actif, alors $b_m < x$; nous allons montrer qu'il n'existe pas d'ordonnancement de délai x exactement. S'il en existait un $\mathcal{T} = (t_u)_{u \in U}$, alors $\mathcal{T}^{x-D}$ serait préférable à $\mathcal{T} - \{l\}$ où l est la dernière tâche ordonnancée par $\mathcal{T}$; U_{x-D} serait préférable a $U - \{l\}$ entraînerait $(U - \{l\})^c$ préférable à $(U_{x-D})^c$; il existerait une tâche $g \in (U-\{l\})^c$ telle que $b_g \leq b_m$ donc $b_g < x$. Que $g = l$ ou $g \neq l$, $\mathcal{T}$ ne serait pas actif.

3.7. Evaluation de l'algorithme

Dans ce paragraphe, on montre comment modifier légèrement l'algorithme de façon à le rendre polynomial.

Corollaire 1

Il suffit de calculer au plus n^2 $\mathcal{T}^x$.

Démonstration

Le delai y des ordonnancements construits tel que :

$\exists \; i$ et k tels que : $a_i + k \times D = y \; (k = 1, n-1)$. Il suffit donc de ne donner à x que des valeurs parmi celles-là.

Remarque

Pour $n = 2 \times k$, $D = k + 1$; $a_i = i$, $b_i = n\ D - i$, pour i égal 1 à k ; $a_{k+j} = D$, $b_{k+j} = (j+1) \times D + 1$ pour i egal $k+1$ à $2 \times k$; le nombre de $\mathcal{T}^x$ à calculer est supérieur à $n^2/4$.

Corollaire 2

Si on donne à x les valeurs du corollaire 1, l'algorithme est en $O(n^3)$.

Démonstration

Le calcul d'un $\mathcal{T}^x$ est en $O(n)$; la proposition en résulte car on calcule au plus n^2 $\mathcal{T}^x$.

Corollaire 3

Le nombre de $\mathcal{T}^x$ à calculer est au plus $\sup_{i\in I} b_i - \min_{i\in I} a_i$.

Corollaire 4

Le nombre de $\mathcal{T}^x$ à calculer est au plus $n\ (2\ D-1)$.

Démonstration

Cela est vérifié dès que l'on rajoute à l'algorithme le test: si $K_x = U_x$, alors poser: $x = \inf_{i\in I-K_x} a_i$.

Corollaire 5

Le nombre de $\mathcal{T}^x$ à conserver en cours d'algorithme est inférieure où égale à $\min(D,n)$.

Remarque

On peut programmer cet algorithme de façon à ce que la complexité soit $\max(O(n^2), O(n \log n \times \min(D,n)))$ et que le nombre de mémoires utilisé simultanément soit $O(n \times \min(D,n))$.

Pour cela, il faut "stocker" r ordonnancements partiels (r variable) codés selon la méthode décrite dans Carlier([3]).

4. MINIMISATION DU NOMBRE DE TACHES EN RETARD

Dans ce paragraphe, la fonction économique à minimiser est le nombre de tâches en retard, on note $T(x,k)$ l'ensemble des ordonnancements partiels de k tâches en un délai inférieure ou égal à x. Nous allons modifié l'algorithme de façon à déterminer un ordonnancement $\mathcal{T}^x_k$ maximal parmi les éléments de $T(x,k)$.

On remplace (ii) par $(ii)'$ (et on ajoute une boucle sur k).

$(ii)'$ Soient $H = K_x - U_{x-D}$, $H' = \{h \in H \ / \ b_h \geqslant x\}$

Si $H' = \phi$,poser: $\mathcal{T}^x_k = \mathcal{T}^{x-1}_k$.

Sinon, soit $m : b_m = \min_{h\in H'} b_h$, poser :

$\mathcal{T}^x_k = \mathcal{T}^{x-D}_{k-1} + \{m\}$.

Théorème

L'ordonnancement $\mathcal{T}^x_k$ est un élément maximal de $T(x,k)$.

Démonstration

Montrons que s'il existe un ordonnancement $\mathcal{T} = (t_u)_{u\in U}$ de délai x, alors H' est non vide et $\mathcal{T}^x_k$ est un élément maximal de $T(x,k)$.

On raisonne sur $U - \{l\}$ où l est la dernière tâche ordonnancée par $\mathcal{T}$, U_{x-D} est préférable à $U - \{l\}$ il en résulte que dans $U - \{l\}$ il y a au moins autant de tâches i telles que $b_i \geqslant x$ que dans U_{x-D}; une de ces tâches n'est pas exécuté dans U_{x-D} ; H'est donc non vide. Pour montrer que $\mathcal{T}^{x-D}_{k-1} + \{m\}$ est preférable à $\mathcal{T}$ on utilise la propriété 1 bis avec: $E = U - \{l\}$, $C = U_{x-D}$, $l = v$, $u = m$, v_0 tâche fictive avec $b_{v_0} = x$.

La complexité de cet algorithme sera en $O(n^3 \log n)$.

5. GENERALISATION AU PROBLEME A P MACHINES IDENTIQUES

On suppose maintenant qu'il y a p machines identiques. Nous allons voir que la méthode se généralise mais la complexité croit très vite avec p .

Nous appelerons délai d'un ordonnancement le p-uple $(y_1, y_2, \ldots, y_p)$ où les y_j sont les délais sur les machines; on supposera en outre $y_1 \leq y_2 \cdots \leq y_p$.

Un ordonnancement d'une partie U des taches sera $x = (x_1, \ldots x_p)$ actif si:

(i) $\not\exists j \leq x_j$

(ii) $\forall r \in U: t_r + D \leq b_r$

(iii) $\not\exists s \in \bar{U}$ $y_p > b_s$

On note $T(x_1, x_2, \ldots, x_p)$ l'ensemble des ordonnancements x-actifs.

Algorithme

On va construire $\mathcal{T}^{x_1, x_2, \ldots, x_p}$ préférable à tout autre élément de $T(x_1, x_2, \ldots \ldots, x_p)$ pour $x_1 \leq x_2 \leq \ldots \leq x_p$ et $|x_p - x_1| \leq D$.

(i) Initialisations

Soit $\alpha = \inf_{i \in I} a_i$; on initialise en calculant $\mathcal{T}^{x_1, x_2, \ldots, x_p}$ pour $x_p \leq \alpha + D$ (cela est facile puisqu'il y a au plus une tâche par machine).

(ii) Calcul de $\mathcal{T}^{x_1, x_2, \ldots, x_p}$

Soit $H = K_{x_p} - U_{x_p - D, x_1, \ldots, x_{p-1}}$

Si $H = \phi$, poser $\mathcal{T}^{x_1, x_2, \ldots, x_p} = \mathcal{T}^{x_1, \ldots, x_p - 1}$

Si H est non vide, soit m la tâche la plus urgente de H ; on pose $\mathcal{T}' = \mathcal{T}^{x_p - D, x_1, \ldots, x_{p-1}} + m$; si $\mathcal{T}'$ est x-actif, on pose

$$\mathcal{T}^{x_1, x_2, \ldots, x_p} = \mathcal{T}' \text{ sinon } \mathcal{T}^{x_1, x_2, \ldots, x_p} = \mathcal{T}^{x_1, \ldots, x_p - 1}.$$

Schéma de la démonstration de l'algorithme

Si H est vide, toutes les tâches exécutables sont ordonnancées; donc pas de difficulté.

Si H est non vide, la démonstration est analogue à celle pour une machine.

Technique de calcul de $\mathcal{T}^{x_1, x_2, \ldots, x_p}$

Il faut calculer $\mathcal{T}^{x_1, x_2, \ldots, x_p}$ pour $|x_p - x_1| \leq D$ et les x_i prenant les $O(n^2)$ valeurs (voir le corollaire 1); ordonnons ces valeurs dans le sens croissant: $z_1, z_2, \ldots$.

Supposons que l'on ait déterminé $\mathcal{T}^{x_1, x_2, \ldots, x_p}$ pour toute combinaison de p valeurs parmi les p premières de la suite: $z_1, z_2, \ldots, z_q$ (avec $|x_p - x_1| \leq D$); donnons maintenant à x_p la valeur z_{q+1} on peut alors évaluer toute combinaison de $p-1$ valeurs parmi les q premières: $x_1, x_2, \ldots, z_{q+1}$ avec $|z_{q+1} - x_i| \leq D$). Le nombre de x_i satisfaisant cette inégalité est au plus $2n - 1$; le nombre de ces combinaisons est donc de l'ordre de n^{p-1} ; au total, il y a au plus $O(n^{p+1})$ ordonnancements à calculer; l'algorithme coûte $O(n^{p+2})$.

Remarque : On peut généraliser également au cas où la fonction économique est la somme des retards.

REMERCIEMENTS

Je remercie J.K. Lenstra et A.H.G.Rinnooy Kan pour l'intérêt qu'ils ont porté à ce travail. Je remercie également B.Lemaire qui dirige mes recherches.

6. BIBLIOGRAPHIE

/1/ K.R. BAKER, "Introduction to Sequencing and Scheduling"., John Wiley and Sons - (1974).

/2/ J. CARLIER, "Travaux du Groupe AFCET de Programmation Combinatoire", EDF (1976).

/3/ J. CARLIER, "The One Machine Sequencing Problem", à paraître dans European Journal of Operational Reseach (1982).

/4/ E.G. COFFMAN, "Computer and Job-Shop -- Scheduling Theory", John Wiley and Sons New York (1976).

/5/ M.R. GAREY and D.S. JOHNSON, Computers and Intractability: A guide to the theory of NP Completeness, W.H. Freeman, -- San Francisco, California (1978).

/6/ M.R. GAREY, D.S. JOHNSON, B.B. SIMONS, and R.E. TARJAN, "Scheduling unit-time tasks with arbitrary release times and deadlines", SIAM J. Computer, 10 ----- (1981), pp 256-269.

/7/ R.L. GRAHAM, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN, "Optimization and - approximation in deterministic sequen-- cing and scheduling a survey Ann. Dis-- crete Math. 5,(1979).

/8/ J.K. LENSTRA, A.H.G. RINNOOY KAN, "Computational Complexity of Discrete Optimization Problem, "Ann-Discrete Math.-- 1979.

/9/ B.J. LAGEWEG, J.K. LENSTRA, A.H.G. RINNOOY KAN, "Minimizing maximum lateness on one machine: Computational experience and some application", Statistica -- Neerlandica, 39 (1976) pp. 25-41

/10/ B.B. SIMONS, "A fast algorithm for multiprocessor scheduling", 21st Annual --- Symposium on Fondation of Computer Science, IEEE Computer Society, Long Beach,-- California, (1980), pp 50-53.

/11/ B.B. SIMONS, "A fast algorithm for single processor scheduling", 19th Annual Symposium on Fondations of Computer --- Science, IEEE Computer Society, Long -- Beach, California,(1978) pp 246-252.

7. NOTA.

(1)
Le probleme est NP complet au sens fort lorsque les durées sont quelconques (Garey et -- Johnson /5/, Lenstra et Rinnooy Kan /8/.)