Belgian Journal of Operations Research, Statistics and Computer Science, Vol 28, n° 1

SOGESCI - B.V.W.B. - PRIZE
1986

L'ORDONNANCEMENT SUR UNE MACHINE AVEC DES CONTRAINTES DE DELAI

R. PETERS
Université Catholique de Louvain, Belgium

ABSTRACT

In this paper we deal with a one-machine scheduling problem to minimize the weighted sum of completion times, subject to deadline constraints. The aim of this study is to present the problem as a mixed integer problem, soluble by a mathematical programming system. Special attention has been paid to finding a formulation of which the linear relaxation is offering a lower bound close to the optimal value, which is the essential difficulty with that kind of problems. A lower bounding procedure using dominance properties and valid inequalities is used in a branch and bound algorithm. The algorithm is tested on problems with up to 30 jobs.

1. Enoncé du problème

Commençons par préciser l'énoncé du "problème d'ordonnancement sur 1 machine avec des contraintes de délai" qui fera l'objet de cette étude . Les symboles, choisis pour représenter les paramètres, proviennent de l'initial des termes anglais : "J" de "job", "p" de "processing time", "d" de deadline" et "w" de "weight"

Soient n travaux $J_1,\ldots,J_n$ devant être exécutés sans interruption sur 1 machine .

Chacun des travaux J_j $(j=1,\ldots,n)$

- requiert une durée de réalisation $p_j \geq 0$ connue ;
- doit être achevé dans un délai $d_j \geq 0$ connu ;
- a un poids d'urgence $w_j \geq 0$ connu .

La machine

- exécute un seul travail à la fois ;
- est prête pour exécuter les travaux $J_1,\ldots,J_n$ dès l'instant initial ;
- ne nécessite pas de périodes d'inactivité entre les exécutions des travaux .

L'objectif est de trouver un ordonnancement des travaux $J_1,\ldots,J_n$ qui minimise la somme pondérée de leur instant de démarrage t_j, soit $\sum_{k=1}^{n} w_k t_k$, tout en respectant les délais limites d_j .

2.Introduction

2.1. Problème sans contraintes de délai

Si on considère le problème sans contraintes de délai, il faut et il suffit d'ordonner les travaux selon un ordre non décroissant des $\frac{p_j}{w_j}$ (ce qui demande O (n log n) opérations) pour obtenir une séquence optimale ; cette séquence est connue dans la littérature comme la "weighted shortest processing time sequence" ou plus brièvement la "WSPT sequence".

L'algorithme découle du théorème suivant dû à Smith .

Théorème : En l'absence de contraintes de délai, toute séquence de travaux J_j est optimum ssi elle est ordonnée selon un ordre non décroissant des $\frac{p_j}{w_j}$.

Preuve :

Les solutions optimales ne se trouvent que parmi les séquences de travaux J_j ordonnés selon un ordre non décroissant des $\frac{p_j}{w_j}$.

Supposons, en effet, qu'il existe une séquence de travaux S optimale, pour laquelle il existe un k tel que

$$\frac{p_k}{w_k} > \frac{p_{k+1}}{w_{k+1}}$$

c-à-d tel que

$$p_k \, w_{k+1} > p_{k+1} \, w_k \qquad (1)$$

Si nous permutons dans ce shéma, le $k^{\text{ème}}$ travail avec le $k+1^{\text{ème}}$ travail, la valeur de l'objectif passe de W à

$$W' = W + (w_k\, p_{k+1} - w_{k+1}\, p_k). \qquad (2)$$

Nous concluons de (1) et (2) que

$W' < W$,

ce qui est contradictoire avec l'hypothèse d'optimalité de S.

De plus, tout ordonnancement des travaux J_j selon un ordre non décroissant des $\frac{p_j}{w_j}$ admet une même valeur pour l'objectif ; les permutations de deux travaux "adjacents" avec même $\frac{p_j}{w_j}$ ne modifient pas l'objectif .

□

2.2. Problème avec des contraintes de délai

Alors que le problème sans contraintes de délai admettait un algorithme efficace, le problème avec des contraintes de délai se complique fortement . Lenstra, Rinnooy Kan et Brucken montrent qu'il est NP - complet . Il y a donc peu d'espoir de trouver un algorithme polynomial pour le résoudre .

Smith, Burns et Miyazaki présentent des heuristiques fournissant "rapidement" au problème une solution réalisable "presque" optimale .

Smith procède de la manière suivante . Il commence par fixer le dernier travail J_j de la séquence . Il le choisit parmi les travaux dont les délais sont au moins aussi grands que le temps de réalisation total et avec $\frac{p_j}{w_j}$ aussi grand que possible . Ensuite, ce travail est écarté du problème . La procédure est répétée jusqu'à ce que tous les travaux soient classés . Plusieurs auteurs se sont penchés sur le caractère d'optimalité de cette heuristique . Lemmons montre par un exemple, que contrairement à ce que prétend Smith, la séquence obtenue n'est pas nécessairement optimale . Burns prouve que l'heuristique de Smith ne fournit une séquence optimale que par rapport à l'ensemble des séquences réalisables différant par une permutation de deux travaux adjacents . Potts et Van Wassenhove établissent des conditions suffisantes d'optimalité pour la séquence obtenue par l'heuristique de Smith : la séquence est optimale si, pour tout $i,j=1,\ldots,n$, $p_i \leq p_j$ entraîne que $w_i \geq w_j$ ou si les travaux J_j s'y ordonnent selon un ordre non decroissant des $\frac{p_j}{w_j}$.

Burns propose une heuristique "2-échange" visant à améliorer la solution qui ordonne les travaux dans un ordre non décroissant des délais . Elle fournit une séquence optimale par rapport à l'ensemble des séquences réalisables différant par une permutation de deux travaux .

Miyazaki élabore une nouvelle heuristique qui tente à améliorer celle de Smith . Cette nouvelle heuristique cherche à réordonner certaines parties de la séquence trouvée par l'algorithme de Smith, parties qui contiennent au moins un travail J_i s'achèvant après le délai du travail qui le précède .

Bansal, Potts - Van Wassenhove et Posner présentent, quant à eux, des algorithmes "branch and bound" fournissant des solutions optimales au problème . Ils utilisent, en outre, des propriétés de dominance qui leur permettent éventuellement de fixer, à priori, la place de certains travaux dans la séquence et de réaliser des taillages lors du "branching". (Ces propriétés seront précisées et complétées plus loin).

Bansal prend comme borne inférieure la valeur de la solution optimale du problème sans contraintes de délai .

Potts et Van Wassenhove utilisent une borne inférieure obtenue en faisant une relaxation Lagrangienne des contraintes de délai et en choisissant les multiplicateurs parmi des valeurs qui permettent à l'heuristique de Smith de fournir des solutions optimales aux relaxations linéaires ("multiplier adjustment method").

Posner propose comme borne inférieure la valeur de la solution optimale du problème où la suspension d'un travail est possible pour en intercaler un ou plusieurs autres . Il prouve que sa borne est plus ou aussi forte que celle de Bansal et Potts - Van Wassenhove .

3. Formulation du problème

Nous allons proposer, pour le problème, une formulation avec les variables $\{D_{ij}\}_{i,j=1,\ldots,n;\ i\neq j}$, où D_{ij} vaut 1 si le travail J_i précède le travail J_j, o sinon . Mais avant, nous allons construire et analyser celle relative au problème sans contraintes de délai . Bien que cette dernière n'ait aucun intérêt pratique (vu l'existence de la "WSPT rule"), cela à l'avantage d'introduire et de motiver la formulation pour notre problème .

3.1. Problème sans contraintes de délai

Soient les variables

$$D_{ij} = \begin{cases} 1 \text{ si le travail } J_i \text{ précède le travail } J_j \\ \\ \text{o sinon} \end{cases} \qquad (i,j=1,\ldots,n;\ i\neq j)$$

t_j = instant de démarrage du travail J_j $(j=1,\ldots,n)$

Le probleme sans contraintes de délai peut se formuler de la manière suivante :

$$\text{minimiser} \quad \sum_{k=1}^{n} w_k t_k \tag{o}$$

sous les contraintes

$$t_j - \sum_{\substack{k=1 \\ k\neq j}}^{n} p_k D_{kj} = o \quad (j=1,\ldots,n) \tag{1}$$

$$D_{ij} + D_{ji} = 1 \qquad (i,j=1,\ldots,n\ ;\ i<j) \tag{2}$$

$$D_{ij}+D_{jk}+D_{ki} \leq 2 \qquad (i,j,k=1,\ldots,n;\ j<i,\ j<k,\ i\neq k) \tag{3}$$

$$t_j \geq o\ ;\ D_{ij} \in \{o,1\} \quad (i,j=1,\ldots,n;\ i\neq j) \tag{4}$$

La contrainte (1) définit les variables t_j . La contrainte (3) impose que tout travail J_i se fasse,soit avant,soit après le travail J_j . La contrainte (3) assure la transitivité de l'ordre: si le travail J_i précède le travail J_j et si le travail J_j précède le travail J_k, alors le travail J_i doit précéder le travail J_k .

Propriété

La relaxation linéaire du problème sans contraintes de transitivité fournit les solutions optimales aux variables $\{t_j\}_{j=1,\ldots,n}$ pour le problème mixte avec les contraintes de transitivité .

Preuve

Soient les deux problèmes

$$(P0)\quad \left|\begin{array}{l} \min \sum_{j=1}^{n} w_j\, t_j \\ t_j - \sum_{\substack{i=1 \\ \neq j}}^{n} p_i\, D_{ij} = 0 \quad (j=1,\ldots,n) \\ D_{ij} + D_{ji} = 1 \quad (i,j = 1,\ldots,n; i<j) \\ D_{ij} + D_{jk} + D_{ki} \leq 2 \quad (i,j,k = 1,\ldots,n\ ;\ j<i, j<k,\ i \neq k) \\ t_j \geq 0\ ;\ D_{ij} \in \{0,1\} \quad (i,j = 1,\ldots,n\ ;\ i \neq j) \end{array}\right.$$

$$(P1)\quad \left|\begin{array}{l} \min \sum_{j=1}^{n} w_j\, t_j \\ t_j - \sum_{\substack{i=1 \\ \neq j}}^{n} p_i\, D_{ij} = 0 \quad (j=1,\ldots,n) \\ D_{ij} + D_{ji} = 1 \quad (i,j = 1,\ldots,n; i<j) \\ t_j \geq 0\ ;\ D_{ij} \geq 0 \quad (i,j = 1,\ldots,n; i \neq j) \end{array}\right.$$

Pour prouver la propriété, il suffit de montrer que toute solution optimale au problème (P0) reste optimale pour le problème (P1), $\forall\ (w_1,\ldots,w_n) \in \mathbb{R}_+^n$.

Soit $\left\{t_j^*\right\}_{j=1,\ldots,n}$ une solutions optimale du problème (P0)

Supposons que la numérotation des travaux soit telle que

$$(t_1^*, t_2^*, \ldots, t_j^*, \ldots, t_n^*) = (0, p_1, \ldots, \sum_{k=1}^{j-1} p_k, \ldots, \sum_{k=1}^{n-1} p_k)$$

Pour le problème (P1), la solution $\left\{t_j^*\right\}_{j=1,\ldots,n}$ est réalisable et donne à l'objectif la valeur

$$Z_{LP} = \sum_{j=1}^{n} w_j \, t_j^*$$

Construisons le dual du problème (P1) :

$$(D1)\quad \left|\begin{array}{lll} \max \sum_{\substack{i,j=1 \\ i<j}}^{n} v_{ij} & & \\ u_j & \leqslant w_j & (j=1,\ldots,n) \\ -p_i u_j + v_{ij} & \leqslant 0 & (i,j = 1,\ldots,n \ ; \ i<j) \\ -p_j u_i + v_{ij} & \leqslant 0 & (i,j = 1,\ldots,n \ ; \ i>j) \end{array}\right.$$

La valeur Z_{LP} est finie, pour tout $(w_1,\ldots,w_n)$ appartenant à $\mathbb{R}_+^n$. De plus, pour le problème (D1) et pour tout $(w_1,\ldots,w_n)$ appartenant à $\mathbb{R}_+^n$,

la solution $(u_j^*, v_{ij}^*, v_{ji}^*)_{i,j=1,..,n}$

où $u_j^* := w_j$

$v_{ij}^* := \min (p_i w_j, p_j w_i)$

$= p_i w_j$ si $i<j$ (car $\frac{p_i}{w_i} \leqslant \frac{p_j}{w_j}$, d'après la WSPT rule)

est réalisable,

et donne à l'objectif la valeur

$$W_{LP} = \sum_{\substack{i,j=1 \\ i<j}}^{n} p_i \, w_j$$

$$= \sum_{j=2}^{n} w_j \sum_{i=1}^{j-1} p_i$$

$$= Z_{LP}$$

Donc la solution $\left\{ t_j^* \right\}$ j=1,...,n est réalisable et optimale pour le problème (P1).

□

3.2. Problème avec des contraintes de délai

Nous définissons, comme pour le problème sans contraintes de délai, les variables :

$$D_{ij} = \begin{cases} 1 \text{ si le travail } J_i \text{ précède le travail } J_j \\ \\ 0 \text{ sinon} \end{cases} \quad (i,j=1,\ldots,n;\ i \neq j)$$

t_j = instant de démarrage du travail J_j $(j=1,\ldots,n)$

Le problème avec les contraintes de délai peut alors naïvement se formuler de la manière suivante :

$$\text{minimiser} \quad \sum_{k=1}^{n} w_k t_k \qquad (o)$$

sous les contraintes

$$t_j - \sum_{k=1}^{n} p_k D_{kj} = o \qquad (j=1,\ldots,n) \qquad (1)$$

$$D_{ij} + D_{ji} = 1 \qquad (i,j,=1,\ldots,n; i<j) \qquad (2)$$

$$D_{ij} + D_{jk} + D_{ki} \leq 2 \qquad (i,j,k=1,\ldots,n;\ j<i, j<k,\ i \neq k) \qquad (3)$$

$$\sum_{\substack{k=1 \\ \neq j}}^{n} p_k D_{kj} \leq d_j - p_j \qquad (j=1,\ldots,n) \qquad (4)$$

$$t_j \geq o;\ D_{ij} \in \{o,1\} \qquad (i,j=1,\ldots,n.i \neq j) \qquad (5)$$

où la contrainte (4) impose trivialement que le travail J_j soit achevé avant d_j .

4. Amélioration de la formulation

Empiriquement, on constate que la relaxation linéaire de la formulation du problème avec des contraintes de délai, présentée ci-avant, fournit généralement de faibles bornes inférieures, même avec des problèmes de petite taille (c-à-d n petit).

Géométriquement, ceci signifie que, dans l'espace des $\{p_{ij}\}_{\substack{i,j=1,\ldots,n \\ i \neq j}}$, le polyèdre décrit par la relaxation linéaire n'est pas assez proche de l'enveloppe convexe des solutions .

Pour s'en rapprocher, nous allons :

1°) incorporer une phase initiale qui, par une simple analyse des données, va fixer la valeur de certaines variables et renforcer les contraintes de délai ;

2°) renforcer les contraintes de délai ;

3°) rajouter des inégalités valables .

Cela permettra de trouver plus rapidement, par l'algorithme "branch and bound", une solution optimale au problème :

- la relaxation linéaire de la formulation fournit une plus forte borne inférieure ;
- l'algorithme "branch and bound" génère plus rapidement des solutions entières .

4.1. Utilisation d'une phase initiale

La phase initiale, qui sera présentée, reprend, en l'améliorant, celle à laquelle a abouti la littérature spécialisée (Posner) . Elle repose essentiellement, sur trois propriétés de "dominance", c-à-d trois propriétés qui spécifient, sous certaines conditions, des contraintes de "précédence" entre travaux, respectées par au moins une solution optimale .

4.1.1. Propriétés de dominance

La première propriété de dominance qui sera présentée est due à Potts et Van Wassenhove . La deuxième constitue une amélioration d'une propriété connue depuis Bansal . La troisième, dans sa forme complète, est mise au point par Posner .

Propriété 1

Soient deux travaux J_i et J_j, i,j=1,...,n et $i \neq j$.

> Si $p_i \leq p_j$, $w_i \geq w_j$ et $d_i \leq d_j$
> alors il existe une séquence optimale dans laquelle le travail J_i précède le travail J_j .

Preuve

Soit une séquence réalisable dans laquelle le travail J_j précède le travail J_i, la paire de travaux (J_j, J_i) étant telle que $p_i \leq p_j$, $w_i \geq w_j$ et $d_i \leq d_j$.

Si on y permute les places de J_i et J_j, la solution reste réalisable, vu que $p_i \leq p_j$ et $d_i \leq d_j$. De plus :
le travail J_j voit sa contribution au coût augmenter de

$$p_i\, w_j + \sum_{k=j+1}^{i-1} p_k\, w_j \ ;$$

le travail J_i voit sa contribution au coût diminuer de

$$p_j\, w_i + \sum_{k=j+1}^{i-1} p_k\, w_i \ ;$$

les travaux exécutés entre J_j et J_i voient leur contribution au coût diminuer de

$$\sum_{k=j+1}^{i-1} (p_j - p_i)\, w_k \ ;$$

le coût total diminue donc de

$$(p_j w_i - p_i w_j) + p_k \sum_{k=j+1}^{i-1} (w_i - w_j) + (p_j - p_i) \sum_{k=j+1}^{i-1} w_k$$

qui est non négatif, vu que $p_i \leq p_j$ et $w_i \geq w_j$.

□

Propriété 2

Supposons que les travaux soient indicés dans un ordre non décroissant de leur délai .

Supposons qu'on ait imposé à la séquence optimale de vérifier certaines relations de "précédence" entre les travaux (à l'aide de propriétés de dominance, par exemple) .

Soit B_i l'ensemble des indices des prédécesseurs du travail J_i qui ont été fixés, pour tout i=1,...,n .

Soit le travail J_j où $2 \leq j \leq n$

S'il existe un q, $1 \leq q < j$, tel que

$$d_q \leq p_j + \sum_{k \leq q} p_k + \sum_{\substack{k > q \\ k \in (UB_i)_{i>q} \cup B_j}} p_k + \sum_{\substack{k > q \\ k \notin (UB_i)_{i>q} \cup B_j}} p_{kq}$$

où $p_{kq} = \max (o, d_q - d_k + p_k)$

alors, dans toute séquence réalisable respectant les contraintes de "précédence", les travaux $J_1,...,J_q$ sont ordonnés avant le travail J_j .

Preuve

Soit q, $1 \leq q < j$, tel que

$$d_q \leq p_j + \sum_{k \leq q} p_k + \sum_{\substack{k > q \\ k \in (UB_i)_{i>q} \cup B_j}} p_k + \sum_{\substack{k > q \\ k \notin (UB_i)_{i>q} \cup B_j}} p_{kq}$$

où $p_{kq} = \max (o, d_q - d_k + p_k)$

Soit une séquence réalisable respectant les contraintes de "précédence" et où le travail J_j s'exécute avant un des travaux $J_1, ...,J_q$.

Au moment d_q, les travaux des ensembles suivants doivent être achevés :

$$\left\{J_i : i \leq q\right\} \cup \left\{J_i : i > q \text{ et } i \in \left\{\bigcup_{i=1,...,q} B_i\right\}\right\} \cup \left\{J_j\right\} \cup \left\{J_i : i > q \text{ et } i \in B_j\right\}$$

De plus, chaque travail J_k de

$$\left\{ J_i \; : \; i > q \quad \text{et} \quad i \notin \left(\bigcup_{i=1,\ldots,q} B_i \right) \cup B_j \right\}$$

doit au moins être commencé en $d_k - p_k$, d'où p_{kq} unités de temps avant d_q .

Donc, avant d_q, la machine doit réaliser du travail exigeant au moins

$$p_j + \sum_{k \leq q} p_k + \sum_{\substack{k > q \\ k \in (UB_i) \cup B_j \\ i>q}} p_k + \sum_{\substack{k > q \\ k \notin (UB_i) \cup B_j \\ i>q}} p_{kq}$$

unités de "temps-machine", ce qui est impossible .

□

Remarque

Jusqu'à ce jour, cette propriété de dominance n'a été présentée, dans la littérature spécialisée, que sous une forme simplifiée, sans les termes $\sum_{\substack{k > q \\ k \in (UB_i) \cup B_j \\ i>q}} p_k$ et $\sum_{\substack{k > q \\ k \notin (UB_i) \cup B_j \\ i>q}} p_{kq}$.

Propriété 3

Supposons que les travaux soient indicés dans un ordre non décroissant de leur délai .

Supposons qu'on ait imposé à la séquence optimale de vérifier certaines relations de "précédence" liant des travaux à des travaux de délai plus élevé (à l'aide de propriétés de dominance, par exemple) . Soit B_i l'ensemble des indices des prédécesseurs du travail J_i qui ont été fixés, pour tout i=1,...,n.

Soit le travail J_j où $2 \leq j \leq n$.

S'il existe un q, $1 \leq q < j$, tel que

$$\frac{p_j}{w_j} \geq \frac{p_i}{w_i} \quad \text{ou } i \in B_j, \quad \forall i = 1,\ldots,q$$

alors il existe une séquence optimale, respectant les contraintes de "précédence", telle que les travaux $J_1,\ldots, J_q$ soient ordonnés avant le travail J_j .

Preuve

La propriété est évidente, d'après le critère d'optimalité, établi par Smith, pour le problème sans contraintes .

□

4.1.2. Algorithme

Tout d'abord, chaque délai d_j qui excède le temps de réalisation total des travaux est réduit à ce dernier :

$$d_j := \min (d_j, \sum_{k=1}^{n} p_k) \qquad (j=1,\ldots,n).$$

Ensuite, on utilise les propriétés de dominance pour fixer certaines variables D_{ij} à 1 .

Quand une nouvelle relation de "précédence" $D_{ij} = 1$ est trouvée par notre procédure, il est évident que le travail J_i et tous ses prédécesseurs déjà fixés doivent être les prédécesseurs du travail J_j et de tous ses successeurs déjà fixés . Ces relations additionnelles doivent être prises en compte aussitôt qu'un "$D_{ij} = 1$" a été fixé .

Après quoi les valeurs des délais peuvent être réadaptées:

$$d_j := \min (d_j, \sum_{k=1}^{n} p_k - \sum_{k \in A_j} p_k) \qquad (j=1,\ldots,n)$$

où A_j est l'ensemble des indices des successeurs du travail J_j qui ont pu être déterminés par l'application des propriétés de dominance;

$$d_j := \min (d_j, d_i - p_i - \sum_{k \in A_j \cap B_i} p_k) \quad (j,i = 1,\ldots,n \; ; \; i \in A_j)$$

où B_i est l'ensemble des indices des prédécesseurs du travail J_i qui ont pu être déterminés par l'application des propriétés de dominance .

Cette réduction des valeurs des délais permet de trouver, par les propriétés de dominance, de nouvelles relations de "précédence" entre les travaux . La procédure est donc répétée jusqu'à ce qu'aucune relation de "précédence" nouvelle ne puisse plus être trouvée .

4.1.3. Remarques

La phase initiale a un double effet bénéfique :

1°) elle réduit la taille du problème en fixant la valeur de certaines variables ;

2°) elle renforce les contraintes de délai, en réduisant les valeurs des délais limites .

Il est important de faire la distinction entre les propriétés de dominance 1 et 3, d'une part, et la propriété de dominance 2, d'autre part . La propriété de dominance 2 exploite uniquement la présence de délais limites dans le problème . En d'autres termes, elle est exploitable dans tout autre problème d'ordonnancement sur une machine avec des délais limites à ne pas dépasser . Par contre, les propriétés de dominance 1 et 3 sont propres à notre problème particulier.

4.2. Reformulation des contraintes de délai

Nous allons renforcer les contraintes de délai :

$$\sum_{\substack{k=1 \\ k \neq j}}^{n} p_k \, D_{kj} \leq d_j - p_j \qquad (j = 1,\ldots,n)$$

qui, rappelons-le, expriment que, dans toute solution réalisable, le travail J_j doit pouvoir s'exécuter avant d_j, pour tout $j=1,\ldots,n$.

Par clarté, nous allons progressivement (en 2 étapes) renforcer ces contraintes .

4.2.1. 1ère étape dans la reformulation

Nous allons exprimer que, dans toute solution réalisable, tous les travaux J_k de délai d'achèvement non supérieur à d_j doivent pouvoir s'exécuter avant d_j, pour tout $j=1,\ldots,n$.

Proposition

Supposons que les travaux soient indicés dans un ordre non décroissant de leur délai .

$$\sum_{k > j} p_k \, D_{kj} \leq d_j - \sum_{k \leq j} p_k$$

est une inégalité valable, $\forall$ $j=1,\ldots,n-1$.

Preuve

Soit D^*_{ij} une solution réalisable

L'ensemble des travaux qui doivent être achevés au moment d_j est :

$$\left\{J_i : i \leq j\right\} \quad \cup \quad \left\{J_i : i > j \text{ et } D^*_{ij} = 1\right\}$$

Donc, le temps de réalisation de ces travaux doit être inférieur ou égal à d_j :

$$\sum_{k \leq j} p_k + \sum_{k > j} p_k \, D^*_{kj} \leq d_j$$

□

4.2.2. 2ème étape dans la reformulation

Nous allons exprimer que, dans toute solution réalisable, tous les travaux J_k de délai de commencement non supérieur à d_j doivent pouvoir s'exécuter, pendant au moins $\min(p_k, d_j-(d_k-p_k))$ unités de temps, avant d_j, pour tout $j=1,\ldots,n$.

Proposition

Supposons que les travaux soient indicés dans un ordre non décroissant de leur délai .

$$\sum_{k>j} (p_k - p_{kj})\, D_{kj} \leq d_j - \sum_{k\leq j} p_k - \sum_{k>j} p_{kj}$$

où $p_{kj} := \max(0, d_j - d_k + p_k)$

est une inégalité valable, $\forall\ j=1,\ldots,n-1$.

Preuve

Soit D^*_{ij} une solution réalisable .

L'ensemble des travaux qui doivent être achevés au moment d_j est :

$$\left\{J_i,\ i \leq j\right\} \quad \cup \quad \left\{J_i,\ i>j \quad \text{et } D^*_{ij}=1\right\}$$

De plus, chaque travail J_k de $\left\{J_i : i>j \text{ et } D^*_{ij}=0\right\}$ doit au moins être commencé en $d_k - p_k$, donc $\max(0, d_j-(d_k-p_k))$ unités de temps avant d_j .

Donc, avant d_j, la machine doit réaliser du travail exigeant au moins

$$\sum_{k\leq j} p_k + \sum_{k>j} (p_k - \max(0, d_j-(d_k-p_k)))\, D_{kj} + \sum_{k>j} \max(0, d_j-d_k-p_k)$$

unités de "temps-machine" .

□

4.3. Addition d'inégalités valables

Nous allons proposer deux nouvelles classes d'inégalités valables . Jointes à notre formulation, elles vont rapprocher de l'enveloppe convexe des solutions, le polyhèdre décrit par la relaxation linéaire .

4.3.1. 1ère classe d'inégalités valables

Nous allons exprimer les contraintes de délai, en tenant compte du fait que si un travail J_i précède un des travaux $J_1,\ldots, J_j$, il se réalise avant le délai d_j .

Proposition

Supposons que les travaux soient indicés dans un ordre non décroissant de leur délai .

$$\sum_{k>j} (p_k - p_{kj}) \max \left\{ D_{kq} \right\}_{q \leq j} \leq d_j - \sum_{k>j} p_k - \sum_{k>j} p_{kj}$$

où $p_{kj} = \max (o, d_j - d_k + p_k)$

est une inégalité valable, $\forall\, j = 1,\ldots,n-1$.

Preuve

Soit D^*_{ij} une solution réalisable .

L'ensemble des travaux qui doivent être achevés au moment d_j est :

$$\left\{ J_i : i \leq j \right\} \quad \cup \quad \left\{ J_i : i > j \text{ et } \max \left\{ D^*_{kq} \right\}_{q \leq j} = 1 \right\}$$

De plus, chaque travail J_k de $\left\{J_i : i > j \text{ et } \max\left\{D^*_{kq}\right\}_{q \leq j} = o\right.$ doit au moins être commencé en $d_k - p_k$, donc $\max(o, d_j-(d_k-p_k))$ unités de temps avant d_j .

Donc, avant d_j, la machine doit réaliser du travail exigeant au moins

$$\overline{\sum_{k \leq j}} \; p_k$$

$$+ \overline{\sum_{k > j}} \; (p_k - \max(o, d_j-(d_k-p_k))) \; \max\left\{D_{kq}\right\}_{q \leq j} + \overline{\sum_{k > j}} \; \max(o, d_j-(d_k-p_k))$$

unités de "temps-machine".

□

4.3.2. 2ème classe d'inégalités valables

Soient les contraintes liant les variables t_j aux variables D_{ij}, les contraintes de délai et les inégalités valables du paragraphe précédant . Si nous les considérons comme des <u>contraintes de type "sac-à-dos"</u> (c-à-d du type $\sum_{k=1}^{m} a_k x_k \leq b$ avec $b \geq 0$, $a_i \geq 0$ et $x_i \in \{0,1\}$, i=1,...,m), il est possible de générer de nouvelles inégalités valables .

Trouver des inégalités valables à partir de contraintes "sac-à-dos" est, en effet, chose connue . La proposition suivante rappelle, comment, dans leur forme la plus élémentaire, elles peuvent se construire (pour avoir plus d'explications et pour connaître les versions améliorées, le lecteur pourra consulter, par exemple, l'ouvrage de Nemhauser et Wolsey) .

<u>Proposition</u>

Soit $S = \left\{ (x_1,\ldots,x_m) : \sum_{k=1}^{m} a_k x_k \leq b \text{ et } x_i \in \{0,1\} \text{ , } i=1,\ldots,m \right\}$

avec $b \geq 0$ et $a_i \geq 0$, i =1,...,m.

Soit $C = \left\{ E \subseteq \{1,\ldots,m\} : \sum_{k \in E} a_k > b \right\}$.

La relation $\sum_{k \in E} x_k \leq |C| - 1$ est une inégalité valable pour S, $\forall E \in C$.

5. Problèmes d'implémentation

Un algorithme "branch and bound" basé sur la formulation à laquelle on a abouti (avec comme borne inférieure initiale la solution de la relaxation linéaire) est, en pratique, inutilisable, dès que le nombre de travaux est plus ou poins élevé : le nombre de variables 0-1, le nombre de contraintes et le nombre d'inégalités valables proposées foisonnent, rendant le temps de résolution excessif ou la place mémoire nécessaire incompatible avec ce que permettent les logiciels commerciaux .

Sans même prendre en compte les contraintes générées dans la phase initiale et les inégalités valables proposées au paragraphe 4.3., on a besoin pour résoudre un problème avec 4 travaux, de 12 variables 0-1 et de 21 contraintes; avec 10 travaux, de 90 variables 0-1 et de 304 contraintes ; avec 50 travaux, de 2450 variables 0-1 et de 40 524 contraintes; ...

La formulation à laquelle nous avons abouti s'exprime, en effet, avec :

$n\ (n-1)$	variables 0-1	D_{ij}
n	variables continues	t_j
n	contraintes définissent les	t_j
$\frac{n\ (n-1)}{2}$	"	liant chaque D_{ij}, avec $i<j$, à D_{ji}
$\frac{n\ (n-1)(n-2)}{3}$	"	de transitivité
$n-1$	"	de délai

plus toutes les contraintes générées, dans la phase initiale
plus toutes les inégalités valables du paragraphe 4.3.

L'objet de ce chapître est de proposer une façon d'utiliser efficacement la formulation, compte tenu des limitations informatiques . Tout d'abord, on mentionnera comment se servir d'un minimum de variables 0-1 . Ensuite, on développera une heuristique de génération de contraintes de transitivité, spécialement encombrantes, vu leur nombre en n^3. Cette heuristique permettra de n'incorporer à la formulation que les contraintes de transitivité "nécessaires" . Enfin, on exposera des heuristiques de génération de coupes, destinées à sélectionner "judicieusement" les inégalités valables à ajouter à la formulation . On aboutira à une stratégie de résolution qui part d'une formulation avec au pire $\frac{n(n-1)}{2}$ variables 0-1, avec n variables continues et avec 2 n contraintes .

5.1.Réduction du nombre de variables 0-1

Deux moyens permettent de réduire le nombre de variables 0-1 .

Tout d'abord, on peut substituer chacune des variables D_{ij}, avec $i < j$, par $1 - D_{ji}$, et éliminer les contraintes $D_{ij} + D_{ji} = 1$ $(i,j = 1,\ldots,n\ ,\ i < j)$. Ceci permet de supprimer la moitié des variables 0-1, et de retirer $\frac{n(n-1)}{2}$ contraintes à la formulation .

Ensuite, on peut remplacer par leur valeur toutes les variables restantes D_{ij} , avec $i > j$, qui ont pu être déterminées lors de la phase initiale (voir paragraphe 4.1) . En général,cela permet d'éliminer beaucoup de variables 0-1, vu la force des propriétés de dominance .

5.2. Génération des contraintes de transitivité

Les contraintes de transitivité alourdissent fort la formulation, puisque leur nombre est de l'ordre de n^3 . Utiliser une heuristique de génération de contraintes permet d'appliquer les programmes d'optimisation sur des problèmes avec moins de contraintes .

Pour déterminer un minorant du coût des solutions réalisables, on va procéder de la manière suivante . On omet initialement les contraintes de transitivité et on résoud la relaxation linéaire . S'il n'y a pas de conflit de transitivité dans la solution, alors cette dernière fixe la borne inférieure . Sinon, on ajoute à la formulation certaines contraintes de transitivité violées, judicieusement choisies ; on répète le processus de résolution d'une relaxation linéaire et d'adjonction de contraintes violées, jusqu'à l'obtention d'une solution compatible avec toutes les contraintes de transitivité ; cette solution fournit la borne inférieure cherchée .

Une fois calculée la borne inférieure, on poursuit la résolution de façon similaire par "branch and bound" . On commence par appliquer l'algorithme "branch and bound" à la formulation avec, comme seules contraintes de transitivité, celles générées par la recherche de la borne inférieure . La solution obtenue est réalisable, et optimale, si elle respecte toutes les contraintes de transitivité . Sinon, on ajoute à la formulation certaines contraintes de transitivité violées, judicieusement choisies; on répète le processus de résolution par "branch and bound" et d'adjonction de contraintes de transitivité, jusqu'à l'obtention d'une solution compatible avec toutes les containtes de transitivité ; cette solution est réalisable, et optimale .

Quant à la détermination des contraintes de transitivité qui sont violées par la solution $\left\{D^{*}_{ij}\right\}_{\substack{i,j=1,\ldots,n \\ i \neq j}}$ une façon, plus rationnelle que l'exploration et la vérification de toutes les contraintes de transitivité, consiste à procéder comme suit . On commence par réindicer les travaux J_j dans un ordre non décroissant du nombre de travaux qui les précèdent, soit $\sum_{\substack{k=1 \\ k \neq j}}^{n} D^{*}_{kj}$; puis, "dans cette nouvelle numérotation", on détermine les contraintes de transitivité violées, c-à-d les relations $D^{*}_{i',j'}+D^{*}_{j',k'}+D^{*}_{k',i'} > 2$ ($j' < i'$, $j' < k'$, $k' \neq i'$), où " * " signifie que l'indice est pris dans la nouvelle numérotation . Ce travail de recherche se trouve allégé, si on ne parcourt que les contraintes $D_{i',j'}+D_{j',k'}+D_{k',i'} \leq 2$ ($j' < i', j' < k', k' \neq i'$) pour lesquelles $D^{*}_{i',j'} \neq 0$. Si $D^{*}_{i',j'} = 0$, les contraintes $D_{i',j'}+D_{j',k'}+D_{k',i'} \leq 2$ ($j' < i', j' < k', k' \neq i'$) ne sont certainement pas violées, de sorte qu'il est, de plus, inutile de les inspecter . Si la séquence à examiner est "proche" de la séquence optimale, ce qu'on espère, beaucoup de variables $D^{*}_{i',j'}$, avec $j' < i'$, sont nulles, ce qui justifie l'emploi de cette stratégie .

Afin de ne pas générer trop de contraintes mutuellement redondantes, on veillera à ne jamais générer au cours d'une même itération deux contraintes de transitivité ayant en commun une variable D_{ij} (sous la forme D_{ij} ou D_{ji}) .

L'heuristique adoptée, pour chercher les contraintes de transitivité à générer, peut donc se formaliser comme suit .

Donnée : D (I,J) représentent les relations d'ordre de la solution à examiner ;

Etape 1 : Réindicer les travaux J_j dans un ordre non décroissant de $\sum_{k=1, k\neq j}^{n} D(K,J)$;soit Z(I,J) les relations d'ordre, de la solution à examiner, dans la nouvelle numérotation ;

Etape 2 :/Initialisation/

```
POUR        I,J=1,...,n
        REPETER     VU (I,J)=0 ;
```

/Recherche des contraintes de transitivités violées/

```
POUR        I=N,...,1
        REPETER POUR J=1,...,I-1
                    REPETER  SI (Z(I,J).EQ.0)
                             ALORS passer à une nouvelle itération en J ;
                             SI  (VU(I,J).EQ.1)  ou (VU(J,I).EQ.1)
                             ALORS passer à une nouvelle itération en J ;
                             POUR K=J+1,..., I-1 et K=I+1,...,N
                                  REPETER  SI    (VU(K,J).EQ.1) ou (VU(J,K).EQ.1)
                                                                 ou
                                           (VU(K,I).EQ.1) ou (VU(I,K).EQ.1)
                                           ALORS passer à une nouvelle itération en K ;
                                           SI     Z(I,J) + Z(J,K) + Z(K,I) > 2
                                           ALORS signaler la contrainte de transitivité
                                                 violée ;
                                                 VU(I,J)=1 ; VU(J,K)= ; VU(K,I)=1 ;
```

5.3. Génération de coupes

Considérer la formulation avec toutes les inégalités valables du paragraphe 4,3 est impensable, vu le nombre de ces dernières . Il est donc bon d'utiliser des stratégies heuristiques de génération de coupes, c-à-d des procédures qui répètent les trois étapes suivantes :

1°) chercher une solution réalisable à la relaxation linéaire du problème ;

2°) choisir judicieusement certaines inégalités valables violées ;

3°) rajouter à la formulation du problème les inégalités valables sélectionnées .

Pour les coupes "sac-à-dos", ce travail est réalisé par des logiciels, comme "MPSARX" qui est en option avec le système de programmation mathématique VM D1.20 de SCICONIC .

Pour les autres coupes, il est rationnel d'adopter une stratégie qui rajoute, à chaque itération au maximum une coupe violée par contrainte de délai ; on choisit, si elle existe, la coupe qui viole le plus la solution "en cours".

6. L'algorithme

Synthétisons sous forme d'un algorithme les résultats auquels nous avons abouti . Nous le présentons sous une forme qui facilite son implémentation sur ordinateur .

Phase 1 : phase initiale

1.1. On réindice les travaux dans un ordre non décroissant de leur délai ;

Si la séquence obtenue en classant les travaux dans l'ordre croissant de leur indice ne respecte pas les contraintes de délai, l'arrêt s'impose : le problème est irréalisable ;

1.2. Chaque délai d_j qui excède le temps de réalisation total des travaux est réduit à ce dernier :

$$d_j := \min (d_j, \sum_{k=1}^{n} p_k) ;$$

1.3. Répéter

1.3.1. déterminer des relations de précédence, en utilisant les propriétés de dominance du paragraphe 4.1.1. et en tenant compte des contraintes de transitivité (détails voir paragraphe 4.1.2) ;

1.3.2. déterminer, pour les temps d'achèvement des travaux dans la séquence optimale recherchée, des majorants qui sont plus petits que les délais donnés dans l'énoncé et qui dorénavant les remplaceront :

pour tout $j=1,\ldots,n$:

$$d_j := \min\left(d_j, \sum_{k=1}^{n} p_k - \sum_{k \in A_j} p_k\right)$$

où A_j est l'ensemble des successeurs du travail J_j qui ont pu être déterminés par l'application des propriétés de dominance;

pour tout $j,i=1,\ldots,n$, $i \in A_j$:

$$d_j := \min\left(d_j, d_i - p_i - \sum_{k \in A_j \cap B_i} p_k\right)$$

où B_i est l'ensemble des prédécesseurs du travail J_i qui ont pu être déterminés par l'application des propriétés de dominance;

tant que des relations de précédence s'obtiennent à l'étape 1.3.1..

Phase 2 : calcul de la borne inférieure

La borne inférieure considérée est donnée par la solution du problème linéaire qui se construit, itérativement, comme suit .

Répéter

2.1. répéter (*)

considérer le problème linéaire

$$\min \sum_{j=1}^{n} w_j t_j$$

$$t_j + \sum_{k<j} p_k D_{jk} - \sum_{k>j} p_k D_{kj} = \sum_{k<j} p_k \quad (j=1,\ldots,n)$$

$$\sum_{k>j} (p_k - p_{kj}) D_{kj} \leq d_j - \sum_{k>j} p_k - \sum_{k>j} p_{kj} \qquad (j=1,\ldots,n)$$

où $p_{kj} = \max(0, d_j - d_k + p_k)$

contraintes de transitivité générées aux itérations précédentes, initialement rien

coupes "sac-à-dos" générées aux itérations précédentes, initialement rien

coupes de type

$$\sum_{k>j} (p_k - p_{kj}) \max \{ D_{kq} \}_{q \leq j} \leq d_j - \sum_{k \leq j} p_k - \sum_{k>j} p_{kj} \quad (j=1,\ldots,n)$$

où $p_{kj} = \max(0, d_j - d_k + p_k)$

générées aux itérations précédentes, initialement rien

$t_j \geq 0$, $0 \leq D_{ij} \leq 1$ $\qquad (i,j=1,\ldots,n \; ; \; i > j)$;

2.1.1. résoudre le problème linéaire ;

2.1.2. générer, à l'aide de MPSARX (**), des coupes "sac-à-dos" violées par la solution ;

tant que des coupes "sac-à-dos" sont générées à l'étape 2.1.2.

2.2. générer des contraintes de transitivité violées par la solution, d'après les indications données au paragraphe 5.2. ;

2.3. générer des coupes du type

$$\sum_{k>j} (p_k - p_{kj}) \max\left\{D_{kq} \mid q \leq j\right\} \leq d_j - \sum_{k \leq j} p_k - \sum_{k>j} p_{kj} \qquad (j=1,\ldots,n)$$

où $p_{kj} = \max(0, d_j - d_k + p_k)$

tant que des contraintes de transitivité ou des coupes sont générées à l'étape 2.2. ou 2.3. .

(*) Toute variable D_{ij} fixée par la phase initiale est à remplacer par sa valeur .

(**) Option du système de programmation mathématique VM D1.20 de SCICONIC ; implémenté par Van Roy et Wolsey en 1983 .

Phase 3 : recherche de la solution optimale

La solution optimale est la solution du problème mixte qui se construit, itérativement, comme suit .

Répéter (*)

considérer le problème mixte

$\min \sum_{j=1}^{n} w_j t_j$ $t_j + \sum_{k<j} p_k D_{jk} - \sum_{k>j} p_k D_{kj} = \sum_{k<j} p_k$ $(j=1,\ldots,n)$ $\sum_{k>j} (p_k - p_{kj}) D_{kj} \leq d_j - \sum_{k<j} p_k - \sum_{k<j} p_{kj}$ $(j=1,\ldots,n)$ où $p_{kj} = \max(0, d_j - d_k + p_k)$
contraintes de transitivité générées à l'étape 2
coupes "sac-à-dos" générées à l'étape 2
coupes de type $\sum_{k>j} (p_k - p_{kj}) \max \{D_{kq}\}_{q \leq j} \; d_j - \sum_{k>j} p_k - \sum_{k>j} p_{kj}$ $(j=1,\ldots,n)$ où $p_{kj} = \max(0, d_j - d_k + p_k)$ générées à l'étape 2
contraintes de transitivités générées à l'étape 3, aux itérations précédentes, initialement rien
$t_j \geq 0$, $D_{ij} \in \{0,1\}$ $(i,j=1,\ldots,n \; ; i > j)$

3.1. résoudre le problème mixte par "branch and bound" avec comme borne inférieure celle donnée par la relaxation linéaire ;

3.2. générer des contraintes de transitivité violées par la solution, d'après les indications données au paragraphe 5.2. ;

tant que des contraintes de transitivité sont générées à l'étape 3.2.

(*) Toute variable D_{ij} fixée par la phase initiale est à remplacer par sa valeur .

7. Essais numériques

Montrons la validité de notre modèle et de notre algorithme, en les appliquant à la résolution de quelques problèmes . Les résultats sont repris dans les tableaux I, IIa et IIb . Ils sont obtenus sans employer la phase initiale, laquelle ne pourrait qu'améliorer davantage les résultats .

Commentaire du tableau I

Le tableau I montre la qualité de la borne inférieure, d'après les contraintes de délai utilisées :

- la borne inférieure 1 est obtenue en se servant de

$$\overbrace{\sum_{k \neq j}} p_k \, D_{kj} \leq d_j - p_j$$

- la borne inférieure 2 est obtenue en se servant de

$$\overbrace{\sum_{k > j}} p_k \, D_{kj} \leq d_j - \overbrace{\sum_{k \leq j}} p_k$$

- la borne inférieure 3 est obtenue en se servant de

$$\overbrace{\sum_{k > j}} (p_k - p_{kj}) \, D_{kj} \leq d_j - \overbrace{\sum_{k \leq j}} p_k - \overbrace{\sum_{k > j}} p_{kj}$$

où $p_{kj} := \max(0, d_j - d_k + p_k)$

- la borne inférieure 4 est obtenue en se servant de

$$\overbrace{\sum_{k > j}} (p_k - p_{kj}) \max \left\{ D_{kq} \right\}_{q \leq j} \leq d_j - \overbrace{\sum_{k \leq j}} p_k - \overbrace{\sum_{k > j}} p_{kj}$$

Les valeurs de la relaxation linéaire du problème sans coupes générées par MPSARX ou avec celles-ci sont données respectivement en regard des options LP ou XLP du tableau .

Les problèmes traités sont énoncés en annexe . L'exemple avec les 4 travaux est celui que Posner a choisi pour illustrer son algorithme . Posner a obtenu 46 $\frac{1}{3}$ comme borne inférieure (à comparer à nos résultats) .

On peut voir, par ces quelques exemples, que la formulation initiale du chapître 3 (bornes inférieures 1,option LP) est mauvaise mais que chacune des modifications apportées au chapître 4 et testées ici améliore sensiblement la qualité de la borne .

On observe de plus, que pour ces petits problèmes, notre algorithme fournit, même sans la phase initiale, des bornes inférieures (bornes inférieures 4, option XLP) identiques aux valeurs optimales .

Commentaire des tableaux IIa et IIb

Les tableaux IIa et IIb donnent une idée de l'évolution de la borne au cours de la phase 2 de l'algorithme, du nombre d'inégalités générées,et du temps CPU requis pour générer les contraintes de transitivité et les coupes autres que celles trouvées par MPSARX (en utilisant les techniques préconisées aux paragraphes 5.2.et 5.3.)

Les problèmes traités sont énoncés en annexe .

Encore une fois, on peut remarquer la qualité des bornes inférieures atteintes par notre algorithme pour ces problèmes . De plus, l'augmentation de la valeur de la borne en cours d'algorithme et la taille des problèmes manipulés sont raisonnables de même que le temps mis pour générer les contraintes de transitivité et les coupes autres que celles ajoutées par MPSARX .

Tableau I : qualité de la borne inférieure

Nombre de travaux	Option	Borne inférieure 1	Borne inférieure 2	Borne inférieure 3	Borne inférieure 4	Optimun
4	LP	42.0	44.9	51.7	52.3	55.0
	X1P	45.0	55.0	55.0	55.0	
10	LP	309.8	326.0	339.8	344.5	364.0
	XLP	318.7	351.1	364.0	364.0	
12	LP	552.0	586.9	624.8	652.8	663.0
	XLP	566.6	637.3	663.0	663.0	
14	LP	724.2	731.6	739.8	744.0	755.0
	XLP	727.4	755.0	755.0	755.0	
15	LP	1098.7	1163.6	1250.1	1260.6	1321.0
	XLP	1100.9	1277.8	1321.0	1321.0	

Tableau IIa: qualité de l'algorithme

Nombre de travaux	Borne inférieure									Optimun
	initiale	1ème itér.	2ème itér.	3ème itér.	4ème itér.	5ème itér.	6ème itér.	7ème itér.	finale	
10	364.0	364.0	364.0	--	--	--	--	--	364.0	364.0
20	3293.0	3662.0	3838.0	3938.0	3947.0	3958.0	3964.0	3973.0	3973.0	3974.0
30	3136.0	3150.0	3162.0	3171.3	3193.7	3218.2	3227.2	3248.0	3319.5	3321.0

Tableau IIb : qualité de l'algorithme

Nombre de travaux	Nombre de lignes		Temps CPU moyen (*) pour générér des	
	initialement	après la 7ème itér.	contraintes de transitivité	coupes autres que celles trouvées par MPSARX
10	20	34	0.24 sec	0.22 sec
20	40	223	0.93 sec	1.16 sec
30	60	438	2.46 sec	2.50 sec

(*) Pour le MV 8000 ; les programmes exécutés sont présentés en annexe .

8. Conclusion

La relaxation linéaire de la formulation de notre problème semble fournir d'excellentes bornes inférieures, proches de la valeur optimale . Celles-ci paraissent meilleures que les bornes qu'il est possible d'atteindre avec les algorithmes proposés jusqu'à ce jour .

De plus, la technique utilisée pour implémenter la formulation est relativement économique en "place-mémoire" et en "temps-calcul", rendant possible la résolution de problèmes qui comportent un nombre assez élevé de travaux .

En augmentant le nombre de variables, il est possible de trouver des formulations dont la relaxation linéaire donne théoriquement des bornes aussi ou encore plus fortes (voir annexe). Cependant, la formulation s'alourdit de façon inopportune . La résolution en est rendue plus pénible et, compte tenue des performances limitées des moyens informatiques, elle n'est possible que pour de plus petits problèmes .

Notre étude ouvre la voie à d'autres recherches, qui traiteraient similairement des problèmes d'ordonnancement avec plusieurs machines, avec d'autres "fonctions-objectifs" et/ ou avec d'autres types de contrainte (instants avant lesquels les travaux ne peuvent être commencés, périodes d'indisponibilité des machines, durées d'inactivité des machines entre les exécutions de deux travaux, priorité dans l'exécution de travaux,...).

REFERENCES BIBLIOGRAPHIQUES

BAKER, K., Introduction to Sequencing and Scheduling, John Wiley & Sons, 1974 .

BANSAN, S.P., Single machine scheduling to minimize weighted sum of completion times with secondary criterion - A branch and bound approach, European Journal of Operational Research, 12, 1980, 177-181 .

BURNS, R.N., Scheduling to minimize the weighted sum of completion times with secondary criteria, Naval Research Logistics Quarterly, 23, 1976, 125-129 .

COFFMAN, E.G., Computer an job-shop scheduling theory, John Wiley & Sons, 1976 .

EMMONS, H.,A note on scheduling with dual criteria, Naval Research Logistics Quarterly, 22, 1975, 615-616 .

LENSTRA, J.K., RINNOOY KAN, A.H.G. and BRUCKER P., Complexity of machine scheduling problems, Ann. Discrete Math., 1, 1977, 343-362 .

NEMHAUSER, Q.L., and Wolsey, L.A., Integer and combinatorial optimization, John Wiley& Sons, 1987 .

MIYAZAKI, S., One machine scheduling with dual criteria, Journal of the Operations Research Society of Japan, 24, 1981, 37 - 50 .

POSNER, M.E., Minimizing weighted completion times with deadlines, Operations Research, 33, 1985, 562-574 .

POTTS, C.N., A lagrangean based branch and bound algorithm for single machine sequencing with precedence constraints to minimize total weighted completion time, Management Science, 31, 1985, 1300-1311.

POTTS, C.N., and VAN WASSENHOVE, L.N., An algorithm for single machine sequencing with deadlines to minimize total weighted completion time, European Journal of Operational Research, 12, 379-387 .

SMITH, W.E., Various optimizers for single-stage production, Naval Research Logistics Quarterly, 3, 1956, 59-66 .

WEISS, H.J., A greedy heuristic for single machine sequencing with precedence constraints, Management Science, 27, 1981, 1209 - 1216 .