7e Conférence Francophone de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 - Paris - France
«Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination»

RÉDUCTIONS DE DOMAINE LAGRANGIENNES POUR LE PROBLÈME DE MINIMISATION DES PÉNALITES D'AVANCE ET DE RETARD PONDERÉES SUR UNE MACHINE

B. DETIENNE, L. PÉRIDY, E. PINSON and D. RIVREAU

Institut de Mathématiques Appliquées
44, rue Rabelais - BP 10808 - 49008 ANGERS CEDEX 01
boris.detienne@uco.fr, laurent.peridy@uco.fr, eric.pinson@uco.fr, david.rivreau@uco.fr

RÉSUMÉ : *Cet article présente de nouvelles règles d'élimination pour le problème de minimisation des pénalités d'avance et de retard sur une machine, avec ou sans dates de disponibilité. Ces règles, basées sur une décomposition lagrangienne, permettent de réduire considérablement les fenêtres d'exécution des tâches, et, ainsi, d'utiliser efficacement d'autres règles d'élimination classiques. Les expérimentations montrent que des instances comportant jusqu'à* 70 *tâches sans date de disponibilité, et* 40 *tâches avec dates de disponibilité, peuvent être résolues optimalement en une heure, en intégrant ces propriétés dans un branch-and-bound.*

MOTS-CLÉS : *Ordonnancement, Juste-à-temps, Règles d'élimination, Relaxation lagrangienne, Méthode exacte*

1. INTRODUCTION

Cet article est dédié à un problème d'ordonnancement particulier, qui se trouve au cœur de la problématique du juste-à-temps, puisqu'il en modélise le cas le plus simple tout en en intégrant les caractéristiques principales. Cette politique industrielle a vu le jour avec la nécessité, de plus en plus pressante car poussée par l'élargissement de la concurrence, de satisfaire les exigences temporelles du client, tout en minimisant les coûts de stockage ou en évitant les pertes de produits périssables.

Notre problème, noté $1|r_j|\sum_j \alpha_j E_j+\beta_j T_j$ dans la notation standard, peut se définir ainsi : on dispose d'un ensemble de tâches J devant être réalisées à l'aide d'un processeur (ou machine) unique, pouvant traiter une seule tâche à la fois. La réalisation de chaque tâche $j \in J$ nécessite p_j unités de temps, à réaliser sans préemption, et après la date de disponibilité r_j. Chaque tâche j possède par ailleurs une date échue d_j, date de fin d'exécution idéale. Si la fin de j survient avant (resp. après) d_j, alors un coût est impacté à j, égal à α_j (resp. β_j) fois le nombre d'unités de temps d'avance (resp. de retard). En notant E_j l'avance (resp. T_j le retard) de la tâche j, le coût total de l'ordonnancement est égal à $\sum_{j\in J} \alpha_j E_j+\beta_j T_j$. Ces quantités, caractérisant le problème, sont entières.

Un état de l'art des études menées sur ce problème et ses variantes peut être trouvé dans (Baker & Scudder 1990) ou (Kanet & Sridharan 2000). Le cas particulier de $1|r_j|\sum_j \alpha_j E_j+\beta_j T_j$, dans lequel toutes les pénalités d'avance et les dates de disponibilité sont nulles ($1||\sum_j w_j T_j$), est NP-difficile (Lenstra, Kan & Brucker 1977) ; le problème général est donc clairement lui aussi NP-difficile.
Il existe quelques cas particuliers polynomiaux, comportant souvent une date échue commune à toutes les tâches, pour certains desquels Hassin et Shani (Hassin & Shani 2005) recensent des algorithmes, comme par exemple pour les cas à date échue commune, sans date de disponibilité et à coûts unitaires, ou encore dans lesquels les coûts d'avance sont tous égaux, de même que les coûts de retard.
Pour le cas général étudié ici, une étude numérique de diverses bornes inférieures est présentée dans (Kedad-Sidhoum, Rios Solis & Sourd 2004), dans laquelle on peut relever l'excellente qualité des bornes basées sur la formalisation indexée sur le temps du problème (Sousa 1989).

Une propriété intéressante du problème a été utilisée dans de nombreuses approches, heuristiques comme exactes : optimiser le problème, relativement à une séquence fixe des tâches, est polynomial (voir par exemple (Hendel & Sourd 2007)).Une grande partie

de la complexité du problème est alors reportée sur la recherche d'une séquence optimale ou de bonne qualité.

La résolution exacte de $1|r_j|\sum_j \alpha_j E_j + \beta_j T_j$ étant difficile, même pour des instances de petite taille, un certain nombre de travaux ont été consacrés à des cas particuliers ou à des méthodes heuristiques. On peut par exemple citer (Fry, Armstrong & Blackstone 1987, Lee & Choi 1995, Bülbül, Kaminsky & Yano 2006).

Des approches permettant de résoudre exactement les problèmes comportant moins de 30 tâches sont proposées par Tanaka et al. (Tanaka, Sasaki & Araki 2003) et Valente et Alves (Valente & Alves 2005), tandis que Sourd et Kedad-Sidhoum (Sourd & Kedad-Sidhoum 2003) proposent un branch-and-bound permettant de résoudre des instances de taille atteignant 30 tâches.
Avec plus de succès, Sourd et Kedad Sidhoum (Sourd & Kedad-Sidhoum 2005) proposent plus récemment un branch-and-bound basé sur une relaxation lagrangienne des contraintes de ressources dans la formalisation indexée sur le temps et de nouvelles règles de dominance permettant de résoudre des instances comportant jusqu'à 50 tâches. Yau et al. (Yau, Pan & Shi 2006), se basant sur une borne inférieure d'une formalisation par affectation, présentent, eux, une hybridation mêlant programmation dynamique et branch-and-bound capable de traiter optimalement des instances de même taille.

Cet article présente tout d'abord une formalisation du problème, puis une décomposition lagrangienne s'appuyant dessus. De nouvelles règles d'élimination basées sur cette décomposition et permettant de restreindre de manière importante les fenêtres d'exécution des tâches sont ensuite décrites. Une méthode arborescente exploitant ces règles et capable de résoudre optimalement en moins d'une heure des instances jusqu'à 70 tâches sans dates de disponibilité, et 40 tâches avec dates de disponibilité, est détaillée, suivie des résultats numériques obtenus.

Formalisation

Introduisons le modèle mathématique sur lequel l'ensemble des travaux présentés dans la suite s'appuie, dérivé de la formalisation indexée sur le temps augmentée d'une contrainte redondante jouant un rôle important dans l'établissement de certaines règles présentées dans cet article.
Formellement, notons :
- $T = [\![0, \tau]\!]$: horizon de planification
- $P = \sum_{j\in J} p_j$: somme des durées des tâches
- $\forall j \in J, D_j = [\![ect_j, lct_j]\!]$: domaine de fin d'exécution de la tâche j
- $\forall j \in J, \forall t \in D_j, c_{jt}$: coût associé à la tâche j si elle s'achève en t
- $\forall j \in J, \forall t \in D_j, \forall \theta \in T$, $a_\theta^{jt} = \begin{cases} 1 \text{ si } t \in [\![\theta - p_j, \theta - 1]\!] \\ 0 \text{ sinon} \end{cases}$: indique si la tâche j est en cours d'exécution en θ lorsqu'elle s'achève en t

On peut alors modéliser le problème sous la forme du programme linéaire en nombres entiers suivant :

$$
\begin{array}{rll}
(ETTIS^D) : \min & \sum_{j\in J}\sum_{t\in D_j} c_{jt}\cdot x_{jt} & \\
s.c. \forall j \in J & \sum_{t\in D_j} x_{jt} & = 1 \quad (1)\\
\forall \theta \in T & \sum_{j\in J}\sum_{t\in D_j} a_\theta^{jt} x_{jt} & \leq y_\theta \quad (2)\\
& \sum_{\theta\in T} y_\theta & = P \quad (3)\\
\forall j \in J, \forall t \in D_j & x_{jt} \in \{0,1\} & \quad (4)\\
\forall \theta \in T & y_\theta \in \{0,1\} & \quad (5)
\end{array}
$$

Dans ce modèle, la variable de décision x_{jt} est égale à 1 si et seulement si la tâche j s'achève en t, et y_θ est égal à 1 si et seulement si une tâche est en cours d'exécution en θ.

2. DÉCOMPOSITION LAGRANGIENNE

Les réductions de domaine présentées dans cet article sont basées sur une décomposition lagrangienne, basée sur le modèle précédent et que nous décrivons dans cette section.

En dualisant le jeu de contraintes de ressources (2) et en lui associant un vecteur de multiplicateurs de Lagrange positif v, on obtient la fonction de Lagrange suivante :

$$
\begin{array}{rl}
\mathcal{L}^D(x,y,v) = & \sum_{j\in J}\sum_{t\in D_j}\left(c_{jt} + \sum_{\theta=t-p_j}^{t-1} v_\theta\right)\cdot x_{jt} \\
& \qquad - \sum_{\theta\in T} v_\theta \cdot y_\theta \\
s.c. & (1)\wedge(3)\wedge(4)\wedge(5)
\end{array}
$$

Pour toute distribution de v, la valeur de la fonction duale $L^D(v) = \min_{x,y}\mathcal{L}^D(x,y,v)$ constitue une borne inférieure de l'optimum de $(ETTIS^D)$. Le calcul de cette fonction est par ailleurs décomposable en sous-problèmes indépendants :
$L^D(v) = \min_{x,y}\mathcal{L}^D(x,y,v) = \sum_{j\in J} SP_x^{Dj}(v) - SP_y^D(v)$.
En notant $\forall j \in J, \forall t \in D_j, \tilde{c}_{jt} = c_{jt} + \sum_{\theta=t-p_j}^{t-1} v_\theta$ le coût réduit de la variable x_{jt}, le j^{eme} sous-problèmes

s'exprime, pour toute tâche $j \in J$:

$$SP_x^{Dj}(v) = \min \sum_{t \in D_j} \tilde{c}_{jt} \cdot x_{jt}$$
$$s.c. \quad (1) \wedge (4)$$

Cette première composante se résout simplement en sélectionnant l'occurence de chaque tâche possédant le coût régularisé le plus faible. Le dernier sous-problème s'écrit ainsi :

$$SP_y^D(v) = \max \sum_{\theta \in T} v_\theta \cdot y_\theta$$
$$s.c. \quad (3) \wedge (5)$$

Sa résolution consiste à sélectionner les instants associés aux P multiplicateurs de valeurs les plus élevées. Cette étape peut se réaliser par un simple tri en $O(\tau \log \tau)$.

La fonction duale L^D peut en pratique être optimisée grâce à une procédure de sous-gradients (Held, Wolfe & Crowder 1974).

3. RÉDUCTIONS DE DOMAINES LAGRANGIENNES

Nous présentons dans cette section un ensemble de règles d'élimination visant à réduire les domaines d'exécution des tâches en exploitant l'information collectée lors de la relaxation lagrangienne.
Ce type de techniques peut être vu comme un cas d'utilisation du *shaving* (Carlier & Pinson 1994, Martin & Schmoys 1996, Carlier, Péridy, Pinson & Rivreau 2004). Le shaving, ou *opérations globales*, consiste à effectuer un test de consistance d'un ensemble de solutions sous une condition particulière. Si cet ensemble se révèle être vide, alors la condition est trivialement respectée par toutes les solutions de l'ensemble initial.

Dans le cas présent, le test de consistance s'effectue au travers du calcul implicite de bornes lagrangiennes. Cette idée a déjà été utilisée dans (Peridy, Pinson & Rivreau 2003), pour le problème $1|r_j|\sum_j w_j U_j$, afin de tester l'hypothèse de retard ou d'avance d'un job particulier. L'approche peut aussi être vue comme une généralisation des *Constraint Programming Based Lagrangean Relaxations*, formalisées pour la première fois dans (Sellmann & Fahle 2001) et étudiées plus avant dans (Sellmann 2004), qui, orientées propagation de contraintes, se focalisent comme (Peridy *et al.* 2003) sur la réduction du domaine d'une variable à la fois, selon le même principe.

Les travaux présentés ici s'en distinguent par l'utilisation intensive des multiplicateurs au sein d'une procédure de sous-gradients, et les configurations relatives aux hypothèses testées plus complexes.

Dans cette section, nous nous plaçons dans le cas où nous disposons :
- d'une borne supérieure UB de l'optimum du problème
- d'une distribution $\bar{v}$ de v
- de la solution optimale du sous-problème lagrangien correspondant. On notera, en particulier, $\forall j \in J, \mu_j = SP_x^{Dj}(\bar{v}) = \min_{t \in D_j} \tilde{c}_{jt}$, coût réduit de la variable sélectionnée pour la tâche j
- de la valeur $L^D(\bar{v})$ de la fonction de Lagrange correspondante

On introduit les notations suivantes :
- $r_{max} = \max_{j \in J} r_j$
- $d_{max} = \max_{j \in J} d_j$
- $\zeta_{max} = \sup(r_{max}, d_{max})$
- $d_{min} = \min_{j \in J} d_j$

Par ailleurs, on notera S^D l'ensemble des solutions vérifiant les contraintes de $(ETTIS^D)$ et de coût inférieur ou égal à UB. Pour chaque tâche j, on définit les domaines de fin d'exécution de manière plus précise : un instant t appartient au domaine de fin d'exécution D_j si et seulement s'il existe au moins une solution de S^D dans laquelle j s'achève en t.

Cette première proposition permet de tester l'hypothèse qu'une tâche s'achève en un instant de temps donné.

Proposition 1 *Soit $j \in J$ et $\theta \in T$.*
Si $L^D(\bar{v}) - \mu_j + \tilde{c}_{j\theta} > UB$ alors $\theta \notin D_j$.

Preuve. Considérons l'ensemble des solutions de coût inférieur ou égal à UB dans lesquelles j s'achève en θ, i.e. $S^{D'}$, avec $\forall i \in J - \{j\}, D'_i = D_i$ et $D'_j = \{\theta\}$. On a donc, si $L^{D'}(\bar{v}) > UB$, $S^{D'} = \emptyset$. Or,

$$\begin{aligned} L^{D'}(\bar{v}) &= \sum_{i \in J} SP_x^{D'i}(\bar{v}) + SP_y^{D'}(\bar{v}) \\ &= \sum_{i \in J-\{j\}} SP_x^{Di}(\bar{v}) + SP_y^{D}(\bar{v}) + SP_x^{D'j}(\bar{v}) \\ &= L^D(\bar{v}) - SP_x^{Dj}(\bar{v}) + SP_x^{D'j}(\bar{v}) \\ &= L^D(\bar{v}) - \mu_j + \tilde{c}_{j\theta} \end{aligned}$$

D'où, si $L^D(\bar{v}) - \mu_j + \tilde{c}_{j\theta} > UB$, alors $S^{D'} = \emptyset$, et donc $\theta \notin D_j$. ■

La proposition suivante permet de tester l'arbitrage d'une disjonction (cf. 4.2).

Proposition 2 *Soient $i \in J$, $j \in J - \{i\}$. On note :*
- $l_i = \inf(lct_i, lct_j - p_j)$
- $e_j = \sup(ect_j, ect_i + p_j)$

– $f_i = \min\{\tilde{c}_{jt} | t \in [\![ect_i, l_i]\!]\}$
– $f_j = \min\{\tilde{c}_{jt} | t \in [\![e_j, lct_j]\!]\}$
Si $L^D(\bar{v}) - \mu_i - \mu_j + f_i + f_j > UB$ *alors* j *est exécutée avant* i *dans toute solution de* S^D.

Preuve. Considérons l'ensemble des solutions de coût inférieur ou égal à UB dans lesquelles i est exécutée avant j, i.e. $S^{D'}$, avec $\forall k \in J - \{i,j\}, D'_k = D_k$, $D'_j = [\![ect_i, l_i]\!]$ et $D'_i = [\![e_j, lct_j]\!]$.
On a donc, si $L^{D'}(\bar{v}) > UB$, $S^{D'} = \emptyset$. Or,

$$\begin{aligned} L^{D'}(\bar{v}) &= \sum_{k \in J} SP_x^{D'k}(\bar{v}) + SP_y^{D'}(\bar{v}) \\ &= \sum_{k \in J-\{i,j\}} SP_x^{Dk}(\bar{v}) + SP_y^{D}(\bar{v}) \\ &\quad + SP_x^{D'i}(\bar{v}) + SP_x^{D'j}(\bar{v}) \\ &= L^D(\bar{v}) - SP_x^{Di}(\bar{v}) - SP_x^{Dj}(\bar{v}) \\ &\quad + SP_x^{D'i}(\bar{v}) + SP_x^{D'j}(\bar{v}) \\ &= L^D(\bar{v}) - \mu_i - \mu_j + f_i + f_j \end{aligned}$$

D'où, si $L^D(\bar{v}) - \mu_i - \mu_j + f_i + f_j > UB$, alors $S^{D'} = \emptyset$, et donc, dans toute solution de S^D, i n'est pas exécutée avant j. Ce qui implique que j est exécutée avant i dans toute solution de S^D. ■

Les deux propositions suivantes permettent l'établissement d'une troisième règle, testant l'hypothèse de démarrage de l'ordonnancement en un instant donné.

Proposition 3 *Il existe au moins une solution optimale sans temps mort entre deux tâches dans les intervalles* $[\![r_{max}, d_{min}]\!]$ *et* $[\![\zeta_{max}, \tau]\!]$.

Preuve. Dans l'intervalle $[\![r_{max}, d_{min}]\!]$, il suffit de vérifier que retarder d'une unité de temps la tâche précédant un tel temps mort engendre une solution de coût au plus égal à celui de la solution courante. Le raisonnement symétrique s'applique indépendamment dans $[\![\zeta_{max}, \tau]\!]$. ■

Proposition 4 *Soit* $t \in T$. *On note :*

– $P = \sum_{j \in J} p_j$
– $Y^-(t) = \begin{cases} [\![t, d_{min}]\!] & si\ t \leq d_{min} \\ \emptyset & sinon \end{cases}$
– $Y^+(t) = \begin{cases} [\![\zeta_{max}, t+P-1]\!] & si\ t+P-1 \geq \zeta_{max} \\ \emptyset & sinon \end{cases}$

S'il existe une solution optimale dont la première tâche débute en t, *alors il existe une solution optimale dans laquelle une tâche est en cours d'exécution durant chaque instant de* $Y^-(t) \cup Y^+(t)$.

Preuve. Evident d'après la proposition 3, en remarquant qu'un ordonnancement réalisable débutant en t ne peut s'achever avant l'instant $t + P - 1$. ■

Proposition 5 *Soit* $t \in [\![0, |T| - P + 1]\!]$. *Relativement à* t, *introduisons les notations :*
– $Y^-(t) = \begin{cases} [\![t, d_{min}]\!] & si\ t \leq d_{min} \\ \emptyset & sinon \end{cases}$
– $Y^+(t) = \begin{cases} [\![\zeta_{max}, t+P-1]\!] & si\ t+P-1 \geq \zeta_{max} \\ \emptyset & sinon \end{cases}$
– $Y^0(t) = T - (Y^-(t) \cup Y^+(t) \cup [\![0, t-1]\!])$
– $r = P - (d_{min} - t)^+ - (t + P - \zeta_{max})^+$
– $Y^*(t) = \{\underset{\theta \in Y^0(t)}{\text{argmax}}\ {}_s\bar{v}_\theta | s \leq r\}$ *: r instants de multiplicateurs de Lagrange les plus élevés dans* $Y^0(t)$
– $SP'_y(\bar{v}, t) = \sum_{\theta \in Y^-(t) \cup Y^+(t) \cup Y^*(t)} \bar{v}_\theta$

Si $SP_x^D(\bar{v}) + SP'_y(\bar{v}, t) > UB$ *alors aucune solution optimale ne débute en* t.

Preuve. Supposons l'existence d'une solution optimale débutant en t.
Alors, d'après la proposition 4, il existe une solution optimale dans laquelle tous les instants de $Y^-(t) \cup Y^+(t)$ sont occupés. Modélisons cette configuration grâce à sa formalisation en programme linéaire en nombres entiers :

$$\begin{aligned} (ETTIS^D)' : min \sum_{j \in J} \sum_{t \in D_j} c_{jt} \cdot x_{jt} & \\ s.c. \forall j \in J \sum_{t \in D_j} x_{jt} &= 1 \\ \forall \theta \in T \sum_{j \in J} \sum_{t \in D_j} a_\theta^{jt} x_{jt} &\leq y_\theta \quad (6) \\ \sum_{\theta \in T} y_\theta &= P \\ \forall \theta \in Y^-(t) \cup Y^+(t) y_\theta &= 1 \\ \forall j \in J, \forall t \in T x_{jt} &\in \{0,1\} \\ \forall \theta \in T y_\theta &\in \{0,1\} \end{aligned}$$

Considérons la relaxation lagrangienne de ce programme consistant à dualiser les contraintes (6). Il est alors évident que, pour toute distribution du vecteur v de multiplicateurs associés, $SP_x^D(v) + SP'_y(v,t)$ est une borne inférieure de l'optimum de $(ETTIS^D)'$. La solution optimale de $(ETTIS^D)'$ devant aussi être optimale pour $(ETTIS^D)$, $SP_x^D(v) + SP'_y(v,t) > UB$ induit une contradiction. ■

On peut grâce à cette proposition écrire un algorithme, de complexité temporelle $O(\tau)$, permettant de calculer un minorant du plus petit instant auquel une solution optimale peut commencer, à partir de l'optimum du sous-problème $SP_x(v)$ ainsi que l'ordre croissant des multiplicateurs de Lagrange.

On peut établir, de la même manière, le résultat ainsi que l'algorithme symétriques, permettant de tester l'hypothèse d'achèvement de l'ordonnancement en un instant donné.

4. BRANCH-AND-BOUND

L'ensemble de ces règles d'élimination a fait l'objet de tests au sein d'une méthode de résolution exacte, que nous décrivons ici, et dans laquelle elles sont utilisées conjointement avec d'autres règles de dominance et d'élimination classiques. La méthode développée est une procédure de séparation-évaluation en profondeur. Bien entendu, la borne inférieure sur laquelle nous nous sommes basés est la relaxation lagrangienne présentée dans la section 2.

4.1 Intégration des réductions de domaine

Les différentes règles d'élimination ont été exploitées en les intégrant à une procédure de sous-gradient (Held et al. 1974) optimisant la fonction duale correspondante. Cette routine, fournissant bien sûr, entre autres, une borne inférieure, exécute, à partir d'un grand nombre de vecteurs de multiplicateurs générés par le processus de convergence, un sous-ensemble ou la totalité des différentes règles, selon des critères empiriques, jusqu'à stabilisation des fenêtres des jobs.

La première règle est appliquée, pour toute tâche j et le multiplicateur de Lagrange $\bar{v}$ courant, en recherchant le plus petit (resp. plus grand) instant θ de D_j pour lequel la proposition 1 n'est pas active. D_j, codé comme un simple intervalle, voit sa borne correspondante ajustée à θ. Bien que $\bar{v}$ ne soit ni optimal ni même proche de l'optimum sous l'hypothèse $D_j = \{\theta\}$, l'utilisation d'un grand nombre de distributions de multiplicateurs différentes rend l'élimination efficace. La seconde règle, issue de la proposition 2, est contrôlée selon le même principe pour toute disjonction non arbitrée et telle que l'optimum du sous-problème lagrangien courant soit incompatible avec l'arbitrage testé (condition *sine qua non* pour que la borne calculée sous l'hypothèse testée soit différente de la borne courante). La troisième règle, issue de la proposition 5 permet éventuellement, toujours selon le même principe, de réduire l'horizon temporel.

Au cours de cette procédure, des bornes supérieures lagrangiennes, dont le calcul est détaillé plus bas (4.5), sont aussi dérivées.

Guidés par les résultats expérimentaux, nous avons choisi d'appliquer les réductions de manière intensive, en chaque noeud de l'exploration, cette technique se révèlant plus efficace qu'une utilisation ponctuelle, comme, par exemple, à la racine de l'arbre exclusivement.

4.2 Autres règles de dominance et d'élimination

De manière assez classique pour un problème disjonctif, le Branch-and-Bound décrit ici utilise activement la notion d'arbitrage de disjonctions (Carlier 1975). D'une part, l'arbitrage (admissible) de toutes les disjonctions revient à définir un séquencement des tâches, dérivable dans notre cas en une solution optimale relativement à la séquence. Rechercher une solution réalisable revient donc à rechercher un arbitrage de toutes les disjonctions. D'autre part, à chaque arbitrage d'une disjonction ou d'un groupe de disjonctions correspond un ajustement des fenêtres d'exécution des tâches concernées, permettant la propagation de l'information en termes de domaines réalisables.

Un certain nombre de règles, déduisant, à partir des fenêtres d'exécution, des arbitrages de disjonctions ainsi que les ajustements associés, peuvent être trouvées dans la littérature. Ces règles s'avèrent très puissantes dès lors que les tâches possèdent des fenêtres d'exécution suffisament serrées. Les procédures de réduction présentées plus haut permettent leur application efficace. Nous présentons maintenant quelques-unes de ces règles, utilisables directement à notre problème.

Proposition 6 Arbitrages triviaux sur disjonctions (Carlier 1975)
Soient i et j deux tâches distinctes de J.
Si $ect_i + p_j > lct_j$ alors $j \rightarrow i$ dans toute solution.

Proposition 7 Ajustements sur disjonctions
Soient i et j deux tâches distinctes de J.
Si $i \rightarrow j$ alors $ect_j \geq ect_i + p_j$ et $lct_i \leq lct_j - p_j$

Carlier et Pinson proposent dans (Carlier & Pinson 1994) un algorithme en $O(n \log n)$ calculant l'ensemble des arbitrages triviaux sur disjonctions et les ajustements correspondants.

Proposition 8 Arb. sur ensembles ascendants et descendants (Carlier & Pinson 1989, Carlier & Pinson 1990)
Soient $j \in J$ et $I \subseteq J - \{j\}$.
Si $\min_{i \in I \cup \{j\}} \{ect_i - p_i\} + p_j + \sum_{i \in I} p_i > \max_{i \in I} lct_i$,
alors $\forall i \in I, i \rightarrow j$ dans toute solution.
Si $\min_{i \in I} \{ect_i - p_i\} + p_j + \sum_{i \in I} p_i > \max_{i \in I \cup \{j\}} lct_i$,
alors $\forall i \in I, j \rightarrow i$ dans toute solution.

Proposition 9 Aj. sur ensembles ascendants et descendants (Carlier & Pinson 1989)
Soient $j \in J$ et $I \subseteq J - \{i\}$.
Si $\min_{i \in I \cup \{j\}} \{ect_i - p_i\} + p_j + \sum_{i \in I} p_i > \max_{i \in I} lct_i$
alors $ect_j \geq \max_{I' \subseteq I} \left\{ \min_{i \in I'} r_i + \sum_{i \in I'} p_i \right\} + p_j$.
Si $\min_{i \in I} \{ect_i - p_i\} + p_j + \sum_{i \in I} p_i > \max_{i \in I \cup \{j\}} lct_i$
alors $lct_j \leq \min_{I' \subseteq I} \left\{ \max_{i \in I'} d_i - \sum_{i \in I'} p_i \right\}$.

Un algorithme de complexité $O(n \log n)$ permettant les arbitrages et les ajustements sur ensembles ascendants/descendants est lui aussi décrit dans (Carlier & Pinson 1994).

La règle de dominance suivante, basée sur une propriété des solutions optimales aux extrémités de l'ordonnancement, est aussi intégrée à notre méthode de résolution.

Proposition 10 *Il existe au moins une solution optimale dans laquelle :*
- *$i \rightarrow j$, pour toute tâche i exécutée intégralement dans $[\![r_{max}, d_{min}]\!]$ et pour toute tâche j différente de i telle que $\frac{\alpha_i}{p_i} > \frac{\alpha_j}{p_j}$*
- *$k \rightarrow l$, pour toute tâche l exécutée intégralement dans $[\![\zeta_{max}, \tau]\!]$ et pour toute tâche k différente de l telle que $\frac{\beta_l}{p_l} > \frac{\beta_k}{p_k}$*

Preuve. Concernant le positionnement relatif de i et j, on peut distinguer trois cas de figure :
- j est exécutée intégralement dans $[\![r_{max}, d_{min}]\!]$. Il suffit alors d'exprimer la différence de coût entre les solutions correspondant aux permutations de deux tâches consécutives pour vérifier la propriété, par transitivité.
- j est exécutée partiellement dans $[\![r_{max}, d_{min}]\!]$. j est alors nécessairement la dernière tâche débutant dans $[\![r_{max}, d_{min}]\!]$, et donc $i \rightarrow j$.
- j est exécutée intégralement dans $]\!]d_{min}, \tau]\!]$. On a alors trivialement $i \rightarrow j$.

Indépendamment, le raisonnement symétrique vaut pour les tâches k et l. La propriété est donc vérifiée dans tous les cas. ■

4.3 Schéma de séparation

Nous avons opté pour un schéma de séparation hybride, comportant deux types de décision. L'exploration des solutions est restreinte à leur séquence, puisqu'on peut facilement calculer la solution optimale relativement à celle-ci.

Globalement, un noeud de l'arbre de recherche est caractérisé par une séquence de tâches *en entrée* de l'ordonnancement, représentant les premières tâches exécutées, une séquence *en sortie*, représentant les dernières tâches, et, pour chaque tâche, un domaine de fin d'exécution. L'ensemble des prédécesseurs et successeurs de chaque tâche, relativement aux disjonctions arbitrées, est lui aussi conservé en chaque noeud.

Le type de décision prise en chaque noeud, par défaut, consiste à choisir une tâche à ajouter à la séquence en cours de construction, en entrée ou en sortie. Cette tâche est choisie selon le nombre d'entrées et sorties possibles, ainsi qu'une évaluation par défaut associée à la décision. Cette dernière est calculée comme mentionné dans la preuve de la proposition 2.

Relativement à une tâche, les réductions lagrangiennes détectent, dans certains cas, des domaines de fin d'exécution admissibles regroupés en intervalles, séparés par une portion de l'horizon relativement importante dans laquelle la tâche ne peut s'achever sans conduire à une solution sub-optimale.
Les domaines de fin d'exécution réalisables de chaque tâche étant traités comme de simples intervalles dans notre implémentation, ces cas conduisent à considérer des tâches aux fenêtres d'exécution très larges, rendant les règles d'élimination inefficaces sur elles ; difficulté se propageant de surcroît à l'ensemble des tâches. Pour éviter ces comportements, on sépare dans ce cas le domaine d'exécution de la tâche en deux sous-ensembles, auxquels on fait correspondre deux fils du noeud courant.

4.4 Initialisation

La phase d'initialisation du branch-and-bound remplit trois rôles : calculer une borne inférieure à la racine de l'arbre de recherche, calculer une borne supérieure, et réduire au maximum les domaines de fin d'exécution des tâches, dans le but commun de réduire le combinatoire nécessaire à l'énumération implicite de toutes les solutions.

La méthode d'initialisation que nous avons développée consiste en deux exécutions de la procédure de sous-gradient modifiée (cf. 4.1). La première exécution permet d'obtenir une bonne borne supérieure et d'effectuer une première réduction des domaines. La seconde exécution, partant de multiplicateurs de Lagrange réinitialisés, permet d'effectuer une réduction des domaines plus forte car elle dispose dès les premières itérations d'une borne supérieure serrée.

4.5 Bornes supérieures

L'efficacité d'un Branch-And-Bound est, en général, beaucoup améliorée par le calcul en chaque noeud de solutions heuristiques de bonne qualité. Dans ce cas particulier, la connaissance d'une borne supérieure est d'autant plus importante qu'elle conditionne l'efficacité des réductions lagrangiennes employées.

La méthode que nous avons développée en utilise deux, basées toutes les deux sur une distribution v des multiplicateurs de Lagrange. Leur principe commun est de dériver, à partir de v et des choix déjà opérés, un séquencement des tâches.
Cette séquence est ensuite calée optimalement. Selon le contexte (début de l'exploration, amélioration de la borne), la séquence sert éventuellement de solution initiale à un algorithme de descente simple, dont le voisinage consiste en l'ensemble des solutions obtenues par une permutation de deux jobs ou par l'extraction/insertion d'un job.
Aucun effort de développement n'a été consacré, dans notre implémentation, à l'optimisation de cette phase. On peut cependant noter qu'une implémentation efficace est possible (Hendel & Sourd 2006).

Les deux méthodes employées pour générer la séquence initiale sont maintenant détaillées.

4.5.1 Séquence optimale du problème lagrangien

L'optimum du problème lagrangien ne fournit pas une solution réalisable, mais fournit néanmoins une séquence qui peut être utilisée comme explicité ci-dessus.
Elle est simplement donnée par l'ordre des instants de fin d'exécution sélectionnés par le sous-problème en x, en différant éventuellement les tâches dont tous les prédécesseurs ne sont pas encore séquencés lors de la construction de la séquence.

4.5.2 Meilleurs enchaînements stricts

On désigne par *enchaînement strict* de deux tâches i et j l'exécution consécutive de i et j sans temps mort. Partant de la supposition que le nombre de temps morts dans une bonne solution est faible, cette heuristique est basée sur la proposition suivante :

Proposition 11 *Soient $i \in J$ et $j \in J - \{i\}$. Considérons l'ensemble S' des solutions dans lesquelles i et j sont exécutées consécutivement et sans temps mort. Posons par ailleurs $a = \sup(ect_i, ect_j - p_j)$ et $b = \inf(lct_j, lct_i + p_j)$, et supposons $a \leq b$ (dans le cas contraire, $S' = \emptyset$).*
Alors, en notant $z(s)$ le coût de la solution s, on a : $\forall s \in S', \forall \bar{v} \in \mathbb{R}^{\tau}, z(s) \geq L^D(\bar{v}) - \mu_i - \mu_j + \min_{t \in [\![a,b]\!]} \{\tilde{c}_{i,t-p_j} + \tilde{c}_{j,t}\}$.

Preuve. On peut aisément vérifier que la contrainte d'exécution consécutive sans temps mort des tâches i et j peut se modéliser en les remplaçant dans le problème initial par une tâche k unique, dont la durée est $p_i + p_j$, le coût d'exécution est $c_{i,t-p_j} + c_{j,t}$, et le domaine de fin d'exécution est $[\![a, b]\!]$.
Il est alors évident que les sous-problèmes lagrangiens relatifs aux autres tâches conservent le même optimum, et que l'optimum de celui correspondant à k est égal à $\min_{t \in [\![a,b]\!]} \{\tilde{c}_{i,t-p_j} + \tilde{c}_{j,t}\}$.
En remplaçant la partie non commune dans l'expression de $L^D(\bar{v})$, on obtient le résultat. ■

On utilise ce résultat en construisant la séquence itérativement de manière gloutonne, en choisissant à chaque fois la tâche dont tous les prédécesseurs sont déjà séquencés et d'évaluation la plus faible relativement à la dernière tâche de la séquence partielle.

5. RÉSULTATS NUMÉRIQUES

L'ensemble des tests reportés ici a été effectué sur un PC équipé de 2 Go de RAM et d'un microprocesseur Intel Pentium D930 cadencé à 3 GHz, sous MS Windows XP, avec des implémentations mono-thread.

5.1 Instances de test

Les différentes expérimentations numériques que nous avons menées ont été effectuées sur des instances générées aléatoirement pour les problèmes sans date de disponibilité, et sur celles proposées dans (Bülbül et al. 2006) pour les problèmes avec dates de disponibilité.

Les instances aléatoires ont été créées comme décrit dans (Sourd & Kedad-Sidhoum 2005) : les dates de disponibilité sont toutes fixées à 0. Le générateur prend en entrée trois paramètres : n, le nombre de tâches, R, facteur contrôlant la dispersion des dates échues (*Range Factor*), et T, facteur gérant la valeur moyenne des dates échues (*Tardiness Factor*).

On tire d'abord aléatoirement la durée de chaque tâche, selon une loi uniforme, dans $[\![10, 100[\![$. Les dates échues sont ensuite tirées elles aussi uniformément dans $[\![d_{min}, d_{min} + R \cdot P]\!]$, avec $d_{min} = \sup(0, P(T - R/2))$ et $P = \sum_j p_j$. Les coûts d'avance

et de retard sont tirés uniformément entre 1 et 5. 5 instances ont été générées pour chacune des combinaison des paramètres suivants : $n \in \{30, 50, 60, 70\}$, $T \in \{0.2, 0.5, 0.8\}$ et $R \in \{0.2, 0.5, 0.8\}$.

5.2 Réductions lagrangiennes

Nous exposons dans cette section les résultats obtenus grâce à notre implémentation des règles d'élimination présentées dans cet article, appliquées aux instances sans date de disponibilité. Nous reportons tout d'abord les temps de calcul nécessaires à la procédure d'initialisation (cf. 4.4), dans la table 1. Comme

Nombre de tâches	Temps moyen (s.)
30	2,9
50	15,4
60	32,2
70	67,3

Tab. 1 – Temps moyen de calcul pour les réductions initiales

on peut le constater, le temps de calcul est doublé pour chaque dizaine de tâches supplémentaires. Cependant, il reste très acceptable, même pour les instances à 70 tâches.

Nombre de tâches	Var. réduites	Disj. Arbitrées
30	91%	79%
50	91%	78%
60	90%	76%
70	90%	76%
Moyenne	91%	77%

Tab. 2 – Part des disjonctions arbitrées et des variables éliminées en sortie des réductions initiales

Une des premières constations qu'on peut faire, relativement à la performance en termes de réduction du problème, est que cette procédure seule permet de résoudre optimalement 20% des instances à 30 tâches, en fournissant la preuve d'optimalité.
Cette optimalité peut être prouvée, durant la procédure, lorsque toutes les disjonctions sont arbitrées, ou tous les domaines des variables réduits à des singletons, ou lorsque le domaine d'une variable est réduit à l'ensemble vide (les réductions sont effectuées relativement à une borne supérieure cible égale au coût de la meilleure solution trouvée moins un). On peut observer que la part de variables éliminées et de disjonctions fixées est quasiment fixe selon la taille des instances. Cette remarque doit être tempérée par le fait que leur nombre initial augmente respectivement de manière linéaire et quadratique en fonction du nombre de tâches. Le tableau 3 récapitule les performances des bornes calculées durant la procédure, déclinées selon les différentes valeurs du *Range Factor*. L'écart entre les deux bornes est très faible en

R	Déviation moyenne
0,2	0,30%
0,5	1,00%
0,8	2,99%
Moyenne	1,43%

Tab. 3 – Déviation des bornes supérieure et inférieure en sortie des réductions initiales

moyenne pour les cas où $R = 0.2$ et plus important pour les instances dotées d'un *Range Factor* plus élevé (une explication de ce phénomène est donnée plus bas).

Nombre de tâches	Nombre d'instances où Heur = Opt
30	34 / 45
50	28 / 45
60	21 / 45
70	17 / 45
Total	100 / 180

Tab. 4 – Nombre d'instances pour lesquelles la solution heuristique est optimale en sortie des réductions initiales

On peut voir dans le tableau 4 que la solution heuristique en sortie de la procédure d'initialisation est optimale (pas nécessairement prouvée comme telle cependant) pour les trois quart des instances à 30 tâches, et plus du tiers pour les instances à 70 tâches.

Ces résultats montrent que les nouvelles règles développées permettent un pré-traitement efficace pour le problème considéré. En effet, elles permettent d'obtenir rapidement des bornes inférieure et supérieure de bonne qualité, et d'éliminer une grande part des variables du problème.

5.3 Branch-And-Bound

La limite de temps d'exécution imposée est de 1 heure. Les données exposées dans cette section correspondent à une implémentation dans laquelle les réductions lagrangienne issues de la proposition 5 sont désactivées. En effet, les réductions brutales de l'horizon qu'elles provoquent rendent la convergence de la procédure de sous-gradients difficile à contrôler, engendrant parfois son arrêt prématuré.

5.3.1 Instances sans date de disponibilité

Le tableau 5 rapporte la part d'instances résolues par le branch-and-bound, ainsi que le temps moyen nécessaire à leurs résolutions.

Nombre de tâches	% optimal	Tps moyen (s.)
30	100%	13,2
50	98%	358,2
60	84%	510,7
70	60%	635,2
Total	86%	343,8

TAB. 5 – Part d'instances résolues optimalement et temps moyen de calcul

Ces résultats montrent que les réductions de domaine lagrangiennes constituent un outil compétitif de résolution pour ce problème, puisque, à notre connaissance, les méthodes publiées actuellement ne rapportent des succès que jusqu'à 50 tâches. Le tableau 6 donne une indication de l'impact de la configuration des instances sur l'efficacité de notre méthode. Il fournit le temps moyen de résolution des problèmes selon les paramètres T et R. On voit nettement, dans ce tableau,

	R			
T	0,2	0,5	0,8	Moyenne
0,2	100%	90%	65%	85%
0,5	100%	90%	70%	87%
0,8	100%	85%	70%	85%
Moyenne	100%	88%	68%	86%

TAB. 6 – Part d'instances résolues optimalement selon les paramètres de génération

que les instances dont les dates échues sont peu dispersées ($R = 0.2$) sont résolues plus facilement que les autres. Cela s'explique par le fait que les règles de dominances aux extrémités de l'ordonnancement (proposition 10) sont plus souvent appliquées, l'intervalle dans lequel elles ne sont pas valides étant déterminé par $[\![d_{min}, d_{max}]\!]$. La même explication vaut pour les résultats moins bons obtenus pour une valeur de $R = 0.8$.

On peut par ailleurs noter que la valeur du *Tardiness Factor* ne semble pas influer sur l'efficacité du branch-and-bound.

5.3.2 Instances avec dates de disponibilité

Comme mentionné plus haut, le jeu d'instances employé pour cette catégorie de problèmes est celui proposé dans (Bülbül et al. 2006).

Ces instances sont générées de manière semblable à celles utilisées plus haut. Dépendant de plus de paramètres, leurs caractéristiques comprennent en outre des durées minimum et maximum, entre lesquelles sont tirées uniformément les durées des tâches. Les dates échues sont générées de manière similaire à précédemment. La date de disponibilité de chaque tâche est tirée aléatoirement dans $[\![0, P]\!]$ (ce qui implique que certaines instances peuvent satisfaire $r_{max} > d_{max}$). Les pénalités d'avance et de retard sont tirées uniformément entre deux valeurs minimale et maximale, paramètres du générateur. La combinaison de ces paramètres donne un jeu de 300 instances par nombre de tâches.

L'ensemble des instances à 20 tâches est résolu optimalement, en un temps moyen inférieur à 2 secondes, 90% des instances sont résolues en moins de 5 secondes et la totalité en moins de 30 secondes. La méthode développée permet par ailleurs de résoudre 92, 7% des instances comportant 40 tâches en moins d'une heure et en un temps moyen de 280 secondes, dont 85% en moins de 1000 secondes et 72% en moins de 300 secondes.

Les performances dégradées de l'algorithme, par rapport au cas sans date de disponibilité, s'expliquent par la réduction de la portion au début de l'horizon où les règles de dominance peuvent s'appliquer, enrayant la déduction d'arbitrages de disjonctions en entrée de la séquence.

5.3.3 Impact des différentes règles

Règles	{A}	{A,E}	{A,1}	{A,E,1}	{A,E,1,2}
% optimal	49%	49%	87%	93%	96%
Tps moyen	12,5	10,4	6,7	4,5	4,3

TAB. 7 – Performance de l'algorithme selon les règles utilisées

Le tableau 7 montre les résultats obtenus par le branch-and-bound sur l'ensemble des instances à 30 tâches, sous une limite de temps de 30 secondes, selon les différentes règles appliquées. L'application de la proposition 1 (1 dans le tableau) est nettement avantageuse par rapport à l'utilisation des seuls arbitrages sur disjonctions et ensembles (A) et des règles de dominance aux extrémités de l'ordonnancement (E). Par ailleurs, l'usage de la proposition 2 (2) permet de résoudre des instances supplémentaires.

CONCLUSION

Cet article présente de nouvelles règles d'élimination pour le problème de minimisation des pénalités d'avance et de retard pondérées sur une machine avec dates de disponibilité, ainsi qu'une méthode de résolution exacte les exploitant. Ces règles s'avèrent être des outils puissants de réduction des variables de ce problème. En effet, le branch-and-bound développé surpasse les méthodes publiées à ce jour à notre connaissance. Elles ont en outre l'avantage d'ouvrir la porte à l'utilisation d'autres règles classique en ordonnancement, basées sur les fenêtres d'exécution des tâches.

On peut imaginer de nombreux prolongements à ces travaux, comme l'exploitation d'autres tests de consistance : on peut en effet facilement tester l'hypothèse de la présence de temps morts dans une solution optimale par une légère modification de la propriété 5. Il serait par ailleurs intéressant de tester différents schémas de séparation pour ce problème, et d'appliquer des méthodes similaires à d'autres problèmes d'optimisation combinatoire, comme $Pm|r_j|\sum_j \alpha_j E_j + \beta_j T_j$.

REFERENCES

Baker, K. R. & Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties : a review, *Operations Research* **38**(1) : 22–36.

Bülbül, K., Kaminsky, P. & Yano, C. (2006). Preemption in single machine earliness/tardiness scheduling, *Journal of scheduling, A paraître* .

Carlier, J. (1975). Thèse de troisième cycle.

Carlier, J. & Pinson, E. (1989). An algorithm for solving the job-shop problem, *Management Science* **35**(2) : 164–176.

Carlier, J. & Pinson, E. (1990). A practical use of jackson's preemptive schedule for solving the job shop problem, *Annals of Operations Research* **26**(1-4) : 269–287.

Carlier, J. & Pinson, E. (1994). Adjustments of heads and tails for the job-shop problem, *European Journal of Operational Research* **78** : 146–161.

Carlier, J., Péridy, L., Pinson, E. & Rivreau, D. (2004). *Handbook of scheduling : Algorithms, Models, and Performance Analysis*, CRC Press, chapter Chapter IV : Elimination rules for job-shop scheduling problem : overview and extensions, pp. 1–19.

Fry, T. D., Armstrong, R. D. & Blackstone, J. H. (1987). Minimizing weighted absolute deviation in single machine scheduling, *IIE Transactions* **19** : 445–450.

Hassin, R. & Shani, M. (2005). Machine scheduling with earliness, tardiness and non-execution penalties, *Computers and Operations Research* **32**(3) : 683–705.

Held, M., Wolfe, P. & Crowder, H. (1974). Validation of subgradient optimization, *Mathematical Programming* **6** : 62–88.

Hendel, Y. & Sourd, F. (2006). Efficient neighborhood search for the one-machine earliness-tardiness scheduling problem, *European Journal of Operational Research* **173**(1) : 108–119.

Hendel, Y. & Sourd, F. (2007). An improved earliness-tardiness timing algorithm, *Computers & Operations Research* **34**(10) : 2931–2938.

Kanet, J. J. & Sridharan, V. (2000). Scheduling with inserted idle time : Problem taxonomy and literature review, *Operations Research* **48**(1) : 99–110.

Kedad-Sidhoum, S., Rios Solis, Y. A. & Sourd, F. (2004). Lower bounds for the earliness-tardiness scheduling problem on parallel machines, *in* A. Oulamara & M. Portmann (eds), *Proceedings of the 9th International Conference on Project Management and Scheduling*, pp. 210–213.

Lee, C. Y. & Choi, J. Y. (1995). A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights, *Comput. Oper. Res.* **22**(8) : 857–869.

Lenstra, J., Kan, A. R. & Brucker, P. (1977). Complexity of machine scheduling problems, *Annals of Discrete Mathematics* **1** : 343–362.

Martin, P. & Schmoys, D. B. (1996). A new approach to computing optimal schedules for the job-shop scheduling problem, *Proceedings of the 5th International IPCO Conference*, pp. 389–403.

Peridy, L., Pinson, E. & Rivreau, D. (2003). Using short-term memory to minimize the weighted number of late jobs on a single machine, *European Journal of Operational Research* **148**(0) : 591–603.

Sellmann, M. (2004). Theoretical foundations of cp-based lagrangian relaxation., *CP*, pp. 634–647.

Sellmann, M. & Fahle, T. (2001). Cp-based lagrangian relaxation for a multimedia application. **URL:** *citeseer.ist.psu.edu/sellmann01cpbased.html*

Sourd, F. & Kedad-Sidhoum, S. (2003). The one machine scheduling with earliness and tardiness penalties, *Journal of Scheduling* **6**(6) : 533–549.

Sourd, F. & Kedad-Sidhoum, S. (2005). An efficient algorithm for the earliness-tardiness scheduling problem, *Technical report*, LIP6.

Sousa, J. (1989). *Time-indexed Formulation of Non-preemptive Single Machine Scheduling Problems*, PhD thesis, Université Catholique de Louvain.

Tanaka, S., Sasaki, T. & Araki, M. (2003). A branch-and-bound algorithm for the single-machine weighted earliness-tardiness scheduling problem with job-independent weights, *IEEE International Conference on Systems, Man and Cybernetics*, pp. 1571–1577.

Valente, J. M. S. & Alves, R. A. F. S. (2005). An exact approach to early/tardy scheduling with release dates, *Computers and Operations Research* **32**(11) : 2905–2917.

Yau, H., Pan, Y. & Shi, L. (2006). New solution approaches to the general single machine earliness-tardiness problem, *IEEE Transactions on Automation Science and Engineering*.