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

GÉNÉRATION DE COLONNES POUR L'ORDONNANCEMENT D'UNE MACHINE À TRAITEMENT PAR FOURNÉES

D. DASTE, C. GUÉRET et C. LAHLOU

École des Mines de Nantes / IRCCyN
La Chantrerie – 4, rue Alfred Kastler – BP 20722
F-44307 Nantes CEDEX 3 – France
denis.daste@emn.fr, christelle.gueret@emn.fr et chams.lahlou@emn.fr

RÉSUMÉ : *Cet article présente un problème d'ordonnancement de tâches de différentes tailles sur une machine à traitement par fournées afin de minimiser le plus grand retard algébrique des tâches. Une machine à traitement par fournées est une machine pouvant exécuter simultanément un ensemble de tâches dont la somme des tailles ne dépasse pas la capacité de la machine. Ce problème, noté* $1|p-batch, non-identical, b<n|L_{\max}$*, est NP-Difficile au sens fort. Une approche par génération de colonnes est proposée. Les expériences numériques montrent qu'elle fournit une très bonne borne inférieure en un temps relativement court.*

MOTS-CLÉS : *génération de colonnes, ordonnancement, machine à traitement par fournée, borne inférieure*

1. INTRODUCTION

Nous nous intéressons au problème d'ordonnancer un nombre fini de tâches de différentes tailles en utilisant une machine à traitement par fournées. Une fournée est un groupe de tâches qui sont traitées en une même opération. Toutes les tâches regroupées commencent et se terminent alors à la même date. Les problèmes associés à ce type de ressources sont appelés *problèmes de fournées (batch scheduling)*.

Les *problèmes de fournées* se rencontrent dans l'industrie (chimie, pharmacie, aéronautique et électronique), où l'utilisation de fours, de séchoirs ou encore d'autoclaves est fréquente.

La première étude sur les *problèmes de fournées* semble être la publication de l'article d'Ikura et Gimble, 1986, qui présente un algorithme exact en $\mathcal{O}(n^2)$ pour le problème avec des tâches de même durée, des tailles identiques et des dates de disponibilité, avec pour objectif la minimisation de la durée totale (*makespan*).

Par la suite, quelques résultats de complexité ont été démontrés, principalement dans les articles de Brucker *et al.*, 1998, Lee *et al.*, 1992, Li and Lee, 1997, Baptiste, 2000, Cheng *et al.*, 2001 et Liu, 2003, pour des problèmes concernant des tâches de même taille et un critère de performance relatif aux dates échues. Des heuristiques (Perez *et al.*, 2000; Wang et Uzsoy, 2002; Uzsoy, 1995), des algorithmes génétiques (Wang et Uzsoy, 2002) et des méthodes exactes (Mehta et Uzsoy, 1998; Wang et Uzsoy, 2002; Webster et Baker, 1995) ont été proposés pour ces problèmes.

Quelques travaux traitent des *problèmes de fournées* concernant des tâches de tailles différentes, mais ces articles portent uniquement sur les problèmes avec un objectif en relation avec les dates de fin des tâches ($\sum C_j$, $\sum w_j C_j$, $C_{\max}$). Des heuristiques (Dobson et Nambimadom, 2001; Zhang *et al.*, 2001; Ghazvini et Dupont, 1998; Chang et Wang, 2004; Li *et al.*, 2005), des algorithmes génétiques (Damodaran *et al.*, 2006), des algorithmes de recuit simulé (Melouk *et al.*, 2004), et des méthodes exates (Azizoglu et Webster, 2000 et 2001; Dupont et Dhaenens-Flipo, 2002) ont été proposés pour résoudre ces problèmes.

Pour une synthèse plus poussée sur les *problèmes de fournées*, les lecteurs intéressés pourront se reporter à l'article de Potts et Kovalyov, 2000.

À notre connaissance, le problème avec des tâches de durées et de tailles différentes, et la minimisation du retard algébrique n'a jamais été étudié, et personne n'a encore proposé d'approche par génération de colonnes pour ce genre de problèmes.

Cet article est organisé de la façon suivante : la première section décrit le problème considéré. Dans la deuxième section, une méthode de génération de colonnes est proposée afin de trouver une borne inférieure pour ce problème. Le protocole expérimental et des résultats sont présentés et analysés dans la section 4. Enfin, nous terminons par quelques remarques sur les résultats obtenus et présentons des perspectives de recherche.

2. DESCRIPTION DU PROBLÈME

Dans le problème auquel nous nous intéressons, un ensemble J de n tâches et une machine à traitement par fournée de capacité $b \in \mathbb{N}$ (modèle borné, *bounded model*) sont donnés. Chaque tâche j est caractérisée par un triplet de nombres entiers (d_j, s_j, p_j), où d_j est la date échue de la tâche j, s_j est l'encombrement de la tâche j sur la machine (taille de la tâche) et p_j est le temps d'exécution de la tâche j. Toutes les données sont déterministes et connues a priori.

La machine à traitement par fournées peut exécuter plusieurs tâches simultanément à condition que la somme des tailles des tâches n'excède pas sa capacité b. Le temps de processus d'une fournée est égal au plus long temps de processus des tâches appartenant à cette fournée. La date C_j de fin d'une tâche est la date de fin de la fournée à laquelle elle appartient. De plus, la machine et les tâches sont disponibles dès l'instant zéro, l'interruption du traitement d'une fournée n'est pas autorisée et on ne peut pas retirer ou ajouter une tâche en cours de traitement.

En utilisant la notation à trois champs $\alpha|\beta|\gamma$ de Graham *et al.*, 1979, ce problème est noté $1|p-batch, non-identical, b < n|L_{\max}$. Le 1 du premier champ indique que la machine est unique. Dans le second champ, $p-batch$ indique un problème de *Batch Processing* ou encore *parallel-batch*, c'est à dire de machine à traitement par fournées, $non-identical$ indique que les tâches sont de tailles s_j différentes, $b < n$ indique que la machine a une capacité fixée. Enfin, le dernier champ, $L_{\max}$, indique le critère à optimiser. $L_{\max} = \max_{j \in J}\{C_j - d_j\} = \max_{j \in J}\{L_j\}$ est le plus grand retard algébrique des tâches.

Le problème $1|p-batch, non-identical, b < n|L_{\max}$ est $\mathcal{NP}$-difficile *au sens fort* puisque Brucker *et al.*, 1998, ont prouvé que le même problème avec des tâches de même taille ($1|p-batch, b < n|L_{\max}$) était $\mathcal{NP}$-difficile *au sens fort*.

Il peut se modéliser de la façon suivante :

Notations

Paramètres

N	Nombre de tâches
K	Nombre de positions pour les fournées dans l'ordonnancement
b	Capacité de la machine
p_j	Durée opératoire de la tâche j
s_j	Taille de la tâche j
d_j	Date échue de la tâche j
$D_{\max}$	$\max_{j=1,\ldots,n}\{d_j\}$
m_j	Nombre entier tel que $m_j = D_{\max} - d_j$

Variables

x_{jq}	Variable binaire, égale à 1 si la tâche j est dans la fournée qui est en position q dans l'ordonnancement, 0 sinon
C_q	Date de fin de la fournée ordonnancée en position q
Δ_q	Date échue de la fournée ordonnancée en position q
$L_{\max}$	Plus grand retard algébrique

Modèle mathématique MM

$$\min L_{\max}$$

$$\sum_{q=1}^{K} x_{jq} = 1 \qquad \forall j = 1,\ldots,N \qquad (1)$$

$$\sum_{j=1}^{N} x_{jq} s_j \leq b \qquad \forall q = 1,\ldots,K \qquad (2)$$

$$x_{jq} p_j + C_{q-1} \leq C_q \qquad \forall j = 1,\ldots,N, \ \forall q = 1,\ldots,K \qquad (3)$$

$$C_0 = 0 \qquad (4)$$

$$\Delta_q \leq D_{\max} - x_{jq} m_j \qquad \forall j = 1,\ldots,N, \ \forall q = 1,\ldots,K \qquad (5)$$

$$C_q - \Delta_q \leq L_{\max} \qquad \forall q = 1,\ldots,K \qquad (6)$$

$$x_{jq} \in \{0,1\} \qquad \forall j = 1,\ldots,N, \ \forall q = 1,\ldots,K \qquad (7)$$

$$C_q \geq 0 \qquad \forall q = 1,\ldots,K \qquad (8)$$

$$\Delta_q \geq 0 \qquad \forall q = 1,\ldots,K \qquad (9)$$

L'objectif de ce problème est de minimiser le plus grand retard algébrique. La contrainte (1) impose que chaque tâche doit être affectée à exactement une position. La contrainte (2) oblige chaque fournée de tâches à ne pas excéder la capacité b de la machine. La contrainte (3) signifie qu'une fournée ne peut démarrer avant la fin de celle qui la précède. La contrainte (4) représente la date de début de l'ordonnancement. La contrainte (5) calcule la date échue de chaque fournée (plus petite date échue des tâches contenues dans cette fournée). En effet, si la tâche j appartient à la fournée qui est en position q dans l'ordonnancement, la contrainte s'écrit $\Delta_q \leq D_{\max} - m_j$, soit $\Delta_q \leq D_{\max} - (D_{\max} - d_j)$, ou encore $\Delta_q \leq d_j$). Et si la tâche j n'appartient pas à cette fournée, nous avons $\Delta_q \leq D_{\max}$. La contrainte (6) indique que le retard algébrique de

l'ordonnancement est supérieur au retard algébrique de chaque fournée. La contrainte (7) définit x_{jq} comme étant une variable binaire. Les contraintes (8) et (9) sont des contraintes de positivité.
Des tests ont montré que ce modèle ne permet pas de résoudre des problèmes de taille 20 en temps raisonnable (moins d'une heure). De plus, la relaxation linéaire fournit une borne inférieure de piètre qualité. Nous proposons donc dans la section suivante une approche par génération de colonnes pour obtenir de meilleures bornes inférieures.

3. GÉNÉRATION DE COLONNES

3.1. Principe général

La technique de génération de colonnes est apparue en 1960 (Dantzig et Wolf, 1960). Elle est particulièrement performante pour résoudre des programmes linéaires continus, appelés programmes maîtres (PM), ayant un nombre si élevé de variables qu'il est impossible de générer explicitement toutes les colonnes de la matrice des contraintes. Pour une description générale de la méthode on consultera Barnhart *et al.*, 1998.

Le principe de la méthode est le suivant : dans le PM, le nombre de colonnes intéressantes (susceptibles de composer de bonnes solutions) est relativement restreint. La génération de colonnes commence donc par résoudre le PM réduit à quelques colonnes intéressantes. Le Programme Linéaire (PL) obtenu, appelé Programme Maître Restreint (PMR), est résolu par l'algorithme du simplexe. Comme il est peu probable que ce PMR donne directement l'optimum du problème initial, il faut alors générer de nouvelles colonnes. Ces colonnes sont générées en résolvant ce que l'on appelle le Sous-Problème du PM (SP). Elles sont calculées à partir des variables duales de la solution courante et sont ajoutées au PMR. Ce dernier est alors de nouveau résolu avec le simplexe, puis de nouvelles colonnes sont ajoutées. Le processus est répété jusqu'à ce que l'on ne trouve plus de nouvelles colonnes améliorantes. On est alors à l'optimum du PM, ce qui nous donne une borne inférieure du problème entier modélisé par le modèle mathématique MM.

3.2. Le programme maître

Dans notre cas, une colonne du PM correspond à une fournée. Le PM a alors pour objectif d'affecter une fournée à une position dans l'ordonnancement en s'assurant que chaque tâche est ordonnancée exactement une fois, et de façon à minimiser le retard algébrique maximal des tâches. Voici sa formulation :

Modèle mathématique pour la génération de colonnes [MMGC]

$$\min L_{\max}$$

$$\sum_{q=1}^{K}\sum_{f=1}^{F}(a_{jf})x_{fq} \geq 1 \qquad \forall j = 1,\dots,N \quad (1)$$

$$C_q - C_{q-1} - \sum_{f=1}^{F}(p^f)x_{fq} \geq 0 \qquad \forall q = 1,\dots,K \quad (2)$$

$$-\sum_{f=1}^{F} x_{fq} \geq -1 \qquad \forall q = 1,\dots,K \quad (3)$$

$$L_{\max} + \Delta_q - C_q \geq 0 \qquad \forall q = 1,\dots,K \quad (4)$$

$$D_{\max} - \Delta_q - \sum_{f=1}^{F}(m^f)x_{fq} \geq 0 \qquad \forall q = 1,\dots,K \quad (5)$$

$$x_{fq} \in \{0,1\} \qquad \forall f = 1,\dots,F, \quad \forall q = 1,\dots,K \quad (6)$$

$$C_q \geq 0 \qquad \forall q = 1,\dots,K \quad (7)$$

$$\Delta_q \geq 0 \qquad \forall q = 1,\dots,K \quad (8)$$

Notations

Paramètres

N	Nombre de tâches
K	Nombre maximal de positions pour les fournées dans l'ordonnancement
F	Nombre de fournées générées
$D_{\max}$	$\max_{j=1,\dots,n}\{d_j\}$
p^f	Durée opératoire de la fournée f
d_j	Date échue de la tâche j
m^f	Nombre entier définit par $m^f = \max_{j\in f}\{m_j\}$
a_{jf}	Valeur binaire valant 1 si la tâche j est dans la fournée f et 0 sinon

Variables

C_q	Date de fin de la fournée ordonnancée en position q
Δ_q	Date échue de la fournée ordonnancée en position q
$L_{\max}$	Plus grand retard algébrique
x_{fq}	Variable binaire, égale à 1 si la fournée f est en position q dans l'ordonnancement 0 sinon

L'objectif de ce problème est de minimiser le plus grand retard algébrique. La contrainte (1) impose que chaque tâche doit être affectée à au moins une position. La contrainte (2) indique qu'une fournée ne peut démarrer avant que celle qui la précède soit terminée. La contrainte (3) interdit d'affecter plus d'une fournée à chaque position. La contrainte (4) impose que le retard algébrique de l'ordonnancement soit supérieur au retard algébrique de chaque fournée. La contrainte (5) calcule la date échue de chaque fournée. Il s'agit de la plus petite date échue des tâches contenues dans cette fournée. La contrainte (6) définit x_{fq} comme étant une variable binaire. Les contraintes (7) et (8) sont des contraintes de

positivité.

La relaxation linéaire obtenue en relâchant la contrainte (6) du modèle MMGC nous donne le programme maître PM.

3.3. Solution initiale

Pour initialiser la génération de colonnes, il est nécessaire de disposer d'une solution initiale réalisable. Une première possibilité est d'utiliser la solution composée de N fournées, chacune contenant une seule tâche. Il est également possible d'utiliser une solution calculée par une bonne heuristique. Les résultats des tests ont montré que la solution initiale calculée à partir d'une bonne heuristique était plus performante que la solution composée des N fournées.

3.4. Sous-Problème

À chaque itération, l'objectif du sous-problème est de déterminer une colonne (fournée) qui améliore la solution courante du PMR. Ceci revient à déterminer une fournée qui respecte la capacité de la machine et qui soit de coût réduit négatif. Il s'agit donc de résoudre, pour chaque position q possible dans l'ordonnancement, le problème modélisé de la façon suivante, où $(\alpha_q, \beta_q, \gamma_q, \delta_q)$ sont les valeurs optimales des variables duales de PMR associées à la position q :

Notations

Paramètres

N	Nombre de tâches
s_j	Taille de la tâche j
p_j	Durée opératoire de la tâche j
b	Capacité de la machine
d_j	Date échue de la tâche j
$D_{\max}$	$\max_{j=1,\ldots,n}\{d_j\}$
m_j	Nombre entier tel que $m_j = D_{\max} - d_j$

Variables

z_j	Variable binaire valant 1 si la tâche j appartient à la fournée, et 0 sinon
p^f	Variable correspondant à la durée opératoire de la fournée f
m^f	$m^f = \max_{j \in f}\{m_j\}$

Sous-Problème [SPq]

$$\min \quad -\sum_{j=1}^{N}(\alpha_j)z_j + (\beta_q)p^f + (\gamma_q) + (\delta_q)m^f$$

$$(p_j)z_j \leq p^f \qquad \forall j = 1,\ldots,N \quad (1)$$

$$\sum_{j=1}^{N}(s_j)z_j \leq b \qquad (2)$$

$$(m_j)z_j \leq m^f \qquad \forall j = 1,\ldots,N \quad (3)$$

$$z_j \in \{0,1\} \qquad \forall j = 1,\ldots,N \quad (4)$$

Dans ce modèle, z_j est une variable valant 1 si la tâche j appartient à la fournée et 0 sinon. L'objectif de ce problème est de trouver une fournée qui respecte la capacité de la machine (contrainte (2)) et dont le coût associé soit le plus petit possible et négatif. La contrainte (1) permet de définir la durée opératoire d'une fournée comme étant la plus longue durée opératoire des tâches se trouvant dans la fournée. La contrainte (3) calcule la date échue d'une fournée (la plus petite date échue des tâches composant la fournée). La contrainte (4) définit z_j comme étant une variable binaire.

Une fois les différents sous-problèmes SP_q résolus, le meilleur couple (fournée,position) est sélectionné, mais seule la fournée (sans sa position) est ajoutée dans le PMR. En effet, il est possible qu'une fournée f associée à une position q générée à une itération donnée devienne intéressante pour une autre position dans les itérations suivantes. Ne pas obliger cette fournée à être ordonnancée en position q évite alors de générer plusieurs fois la même fournée.

Il faut noter que l'on peut, à chaque itération, ajouter une seule fournée (la fournée de plus petit coût réduit négatif trouvée) ou les p meilleures fournées parmi celles de coût réduit négatif trouvées. Des tests ont montré que, dans le dernier cas, $p = N$ donne les meilleurs résultats.

Nous proposons par la suite une méthode approchée pour ce sous-problème ainsi qu'une méthode exacte utilisée lorsque la méthode approchée ne trouve plus de solution de coût réduit négatif.

3.5. Méthode approchée de résolution du sous-problème

La méthode approchée que nous proposons est une heuristique gloutonne permettant de générer rapidement des fournées. Elle consiste à ajouter à chaque itération à la fournée courante la tâche qui augmente le moins le coût.

Son principe général est le suivant :

1. Initialiser la fournée f avec une seule tâche
2. Tant qu'il est possible d'ajouter une tâche à f
3. Ajouter la tâche qui augmente le moins le coût de f

Cet algorithme est exécuté N fois, une fois pour chaque tâche que l'on peut choisir à l'étape 1. Deux versions de cet algorithme sont proposées : dans un cas, seule la meilleure fournée rencontrée est conservée (algorithme noté GLO), dans un autre cas les

N meilleures fournées rencontrées sont conservées (algorithme noté GLO_N).

Si aucune fournée générée par cet algorithme n'a un coût réduit négatif, alors le sous-problème est résolu au moyen d'une méthode exacte.

3.6. Méthodes exactes de résolution du sous-problème

Deux méthodes exactes de résolution du sous-problème sont utilisées.

La première, notée PLNE, consiste à résoudre le programmation linéaire en nombres entiers du sous-problème à l'aide d'un solveur (Xpress-MP dans notre cas).

La seconde méthode consiste à énumèrer toutes les fournées, puis à les évaluer. Dans un cas, seule la meilleure fournée rencontrée est conservée (algorithme noté $ENUM$), dans un autre cas les N meilleures fournées générées sont conservées (algorithme noté $ENUM_N$).

L'algorithme $ENUM_N$ construit toutes les fournées contenant une seule tâche, puis toutes celles contenant deux tâches, et ainsi de suite. Pour éviter de construire plusieurs fois une même fournée, l'algorithme mémorise l'ordre dans lequel sont ajoutées les tâches dans une fournée. De plus, l'énumération de l'ensemble des fournées peut être accélérée en triant l'ensemble des tâches suivant l'ordre croissant des s_j. Plus précisément, l'algorithme procède de la façon suivante :

1. Soit $(j_1, j_2, \ldots, j_N)$ la liste des tâches triées par ordre de taille croissante
2. Initialiser une file F avec une fournée constituée de la première tâche
3. TantQue F n'est pas vide
4. Soit f la fournée en tête de file
5. Soit j_k la tâche ajoutée à f en dernier
6. Pour i allant de j_{k+1} à j_N
7. Si i peut être ajoutée à f
8. Mettre dans F la fournée composée de f et i
9. Sinon
10. sortir de la boucle Pour
11. FinPour
12. Enlever f de F
13. FinTantQue

On notera que c'est à l'étape 10 que l'énumération est accélérée en évitant de tester les tâches qui ne rentreraient pas dans la fournée, les tâches étant triées par ordre de taille croissante à l'étape 1.

4. EXPÉRIMENTATION NUMÉRIQUE

4.1. Création des jeux de données

La méthode de génération de colonnes a été testée sur des jeux de données de tailles différentes générées aléatoirement. Dans ces jeux de données, le nombre de tâches est 10, 20, 50 ou 100. Les temps de processus sont déterminés en utilisant une distribution uniforme ($p_j = U[1, 99]$). Cette distribution est basée sur des applications liées au problème de l'industrie électronique. Pour un jeu de données déterminé, après avoir calculé les durées opératoires p_j de chaque tâche j, les dates échues sont générées en utilisant la formule : $d_j = \widetilde{C}_{\max} U[0, \alpha] + U[1, \beta] p_j$ (inspirée des travaux de Uzsoy, 1995; Wang et Uzsoy, 2002; Malve et Uzsoy, 2007) où $\widetilde{C}_{\max} = \sum s_j \sum p_j / bn$ est une approximation du temps passé par toutes les tâches sur la machine, $\alpha = 0.1$ et $\beta = 3$ (ce qui correspond aux problèmes les plus difficiles). La taille des tâches est générée à partir d'une distribution uniforme discrète entre 1 et 10. La capacité de la machine est de 10 (inspiré des travaux de Dupont et Dhaenens-Flipo, 2002 et, Azizoglu et Webster, 2000). Pour chacune de ces combinaisons de paramètres, 100 jeux de données sont générés.

4.2. Résultats

Les tableaux 1 et 2 comparent les différentes méthodes utilisées lors de la génération de colonnes : $GLO - ENUM$ résout le sous-problème avec l'algorithme glouton en générant une fournée, puis appelle l'algorithme d'énumération pour résoudre le sous-problème de façon exacte, $GLO_N - ENUM$ utilise l'algorithme glouton pour générer N fournées à chaque itération puis appelle l'algorithme d'énumération une fois que la première méthode ne trouve plus de fournées, $ENUM$ utilise l'algorithme d'énumération à chaque étape de résolution du sous-problème, $PLNE$ utilise la Programmation Linéaire en Nombres Entiers à chaque étape de résolution du sous-problème, enfin $ENUM_N$ utilise l'algorithme d'énumération génèrant N fournées à chaque itération du sous-problème.

Les résultats pour N=10, 20 et 50 montrent que

la méthode GLO_N-$ENUM$ (qui associe l'algorithme glouton générant plusieurs colonnes à l'énumération) est la plus performante. C'est pourquoi pour $N = 100$ nous n'avons testé que cette méthode. Elle nécessite moins d'itérations et moins d'appels à une résolution exacte que les autes méthodes, en particulier moins que la méthode GLO-$ENUM$ qui ne génère qu'une seule fournée à chaque itération (voir tableau 1). Concernant les temps d'exécution, c'est aussi la méthode la plus rapide : elle nécessite environ deux fois moins de temps que la méthode GLO-$ENUM$ (voir tableau 2).

N=10	# Colonnes Générées	# Itérations	# Appels Résolution Exacte
GLO-ENUM	17.26	9.25	2.37
GLO_N-*ENUM*	24.82	4.73	2.02
ENUM	18.14	9.14	9.14
$ENUM_N$	18.32	9.32	9.32
PLNE	18.32	9.32	9.32
N=20	# Colonnes Générées	# Itérations	# Appels Résolution Exacte
GLO1-ENUM	49.01	21.88	2.53
GLO_N-*ENUM*	77.23	6.39	2.35
ENUM	42.38	23.38	23.38
$ENUM_N$	42.36	23.36	23.36
PLNE	42.36	23.36	23.36
N=50	# Colonnes Générées	# Itérations	# Appels Résolution Exacte
GLO1-ENUM	181.83	67.33	3.43
GLO_N-*ENUM*	342.55	9.78	2.91
ENUM	127.844	78.8438	78.8438
$ENUM_N$	423.125	9.71875	9.71875
PLNE	127.094	78.0938	78.0938
N=100	# Colonnes Générées	# Itérations	# Appels Résolution Exacte
GLO_N-*ENUM*	1430	11.5	2.92

Table 1: Résultats des différentes méthodes de résolution du sous-problème

Afin d'évaluer la qualité de la borne inférieure obtenue par génération de colonnes, nous l'avons comparée aux solutions exactes obtenues par Xpress-MP pour le modèle MM (ou aux meilleures solutions trouvées par Xpress-MP si la solution optimale n'est pas trouvée au bout d'une heure). Nous avons également effectué cette comparaison pour la borne inférieure obtenue par relaxation linéaire du modèle MM (notée MM RL).

N=10	Moyen	Minimal	Maximal
GLO1-ENUM	0.11	0.02	0.942
GLO_N-*ENUM*	0.08	0.02	1.20
ENUM	0.19521	0.03	1.041
$ENUM_N$	5.37724	1.172	15.563
PLNE	5.77975	1.191	15.873
N=20	Moyen	Minimal	Maximal
GLO1-ENUM	0.94	0.22	2.053
GLO_N-*ENUM*	0.55	0.11	1.482
ENUM	5.23922	0.461	29.863
$ENUM_N$	41.0848	9.654	77.812
PLNE	43.7229	9.754	111.3
N=50	Moyen	Minimal	Maximal
GLO1-ENUM	32.98	9.70	135.80
GLO_N-*ENUM*	17.37	4.33	48.92
ENUM	149.111	18.391	1164.48
$ENUM_N$	36.1674	6.875	171.859
PLNE	577.041	292.761	1219.97
N=100	Moyen	Minimal	Maximal
GLO_N-*ENUM*	905.782	122.245	5753.58

Table 2: Temps d'exécution en secondes

Le tableau 3 donne pour chaque taille de problème et pour chacune des deux bornes inférieures, la déviation entre la borne inférieure $BInf$ et la solution exacte (ou approchée) SOL obtenue par Xpress-MP en utilisant la formule suivante, inspirée des travaux de Malve et Uzsoy, 2007 : $DEV = \frac{SOL+d_{\max}}{BInf+d_{\max}}$ où $d_{\max} = \max_{j \in J}\{d_j\}$. Dans cette formule, une valeur $d_{\max}$ est ajoutée aux bornes inférieures et solutions exactes afin d'éviter les valeurs négatives. Plus cette déviation est proche de 1 et plus la borne inférieure est de bonne qualité.

N=10	Moyen	Min	Max
GEN COL	1.13439	1	1.58824
MOD1 RL	3.95544	2.59551	6.40845
N=20	Moyen	Min	Max
GEN COL	1.05209	1	1.16429
MOD1 RL	6.87773	4.72165	10.4444
N=50	Moyen	Min	Max
GEN COL	1.08982	1.02149	1.1895
MOD1 RL	16.8483	13.1263	21.0714
N=100	Moyen	Min	Max
GEN COL	1.18218	1.11958	1.31387
MOD1 RL	35.6144	31.3646	38.9184

Table 3: Déviation par rapport aux meilleures solutions connues

Il est clair d'après ce tableau que la borne inférieure

obtenue par génération de colonnes est bien meilleure que la relaxation linéaire de MM. Elle est de plus proche de l'optimum (déviation proche de 1).

	SOL	GEN COL	MM RL
N=10	135.06	98.56	-134.87
N=20	386.96	357.73	-172.05
N=50	1305.88	1176.72	-238.5
N=100	3015.36	2480.28	-389.5

Table 4: Comparaison des bornes inférieures

Ces résultats sont confirmés par le tableau 4 qui compare la moyenne des bornes inférieures obtenues avec la génération de colonnes avec celle obtenue par la relaxation linéaire de MM, ainsi qu'avec celle des meilleures solutions obtenues par Xpress-MP en moins d'une heure.

5. CONCLUSION

Dans cet article, nous proposons une méthode de génération de colonnes afin d'obtenir une borne inférieure pour le problème $1|p - batch, non - identical, b < n|L_{\max}$, qui est NP-Difficile au sens fort. Les résultats obtenus sont très satisfaisants. La borne inférieure est obtenue rapidement : moins d'une seconde pour des problèmes de taille $N = 20$, alors que la résolution exacte de ces problèmes nécessite plus d'une heure de calcul, et en moyenne 15 minutes pour les problèmes de taille $N = 100$. La qualité de cette borne est également remarquable, améliorant nettement la borne inférieure obtenue par le solveur.

Nous envisageons maintenant d'utiliser cette borne dans un algorithme de *Branch and Price* afin d'obtenir une solution exacte au problème. Cette borne pourra de plus permettre d'évaluer la qualité d'heuristiques.

6. REMERCIEMENTS

Nous tenons à remercier la région Pays de la Loire qui a soutenu financièrement ces travaux.

References

Azizoglu M. and S. Webster, 2000. Scheduling a batch processing machine with non-identical job sizes. *International Journal of Production Research*, 38(10):2173–2184.

Azizoglu M. and S. Webster, 2001. Scheduling a batch processing machine with incompatible job families. *Computers and Industrial Engineering*, 39((3-4)):325–335, April 2001.

Baptiste P., 2000. Batching identical jobs. *Mathematical Methods of Operations Research*, 52:355–367.

Barnhart C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance, 1998. Branch and price : Column generation for solving huge integer programs. *Operations Research*, 46:316–329.

Brucker P., A. Gladky, J.A. Hoogeveen, M.Y. Kovalyov, C.N. Potts, T.Tautenhahn, and S.L.van de Velde, 1998. Scheduling a batching machine. *Journal of Scheduling*, 1:31–54.

Chang P.C. and H.M. Wang, 2004. A heuristic for a batch processing machine scheduled to minimise total completion time with non-identical job sizes. *International Journal Adv Manufacturing Technologies*, 24:615–620.

Cheng T.C.E., Z. Liu, and W. Yu, 2001. Scheduling jobs with release dates and deadlines on a batch processing machine. *IIE Transactions*, 33:685–690.

Damodaran P., P.K. Manjeshwar, and K. Srihari, 2006. Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms. *International Journal of Production Economics*, 103(2):2064–2079, October 2006.

Dantzig G.B. and P. Wolf, 1960. Decomposition principle for linear programs. *Operations Research*, 8:101–111.

Dobson G. and R.S. Nambimadom, 2001. The batch loading and scheduling problem. *Operations Research*, 49(1):52–65, January-February 2001.

Dupont L. and C. Dhaenens-Flipo, 2002. Minimizing the makespan on a batch machine with non-identical job sizes: An exact procedure. *Computers and Operations Research*, 29:807–819.

Ghazvini F.J. and L. Dupont, 1998. Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes. *International Journal of Production Economics*, 55:273–280.

Graham R.L., E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnoy Kan, 1979. Optimization and approximation in deterministic sequencing and scheduling : a survey. *Annals of Discrete Mathematics*, 5:287–326.

Ikura Y. and M. Gimple, 1986. Scheduling algorithms for a single batch processing machine. *Operations Research Letters*, 5:61–65.

Lee C.Y., R. Uzsoy, and L.A. Martin-Vega, 1992. Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research*, 40(4):764–775, July-August 1992.

Li C.L. and C.Y. Lee, 1997. Scheduling with agreable release times and due dates on a batch processing machine. *European Journal of Operational Research*, 96:564–569.

Li S., G. Li, and S. Zhang, 2005. Minimizing makespan with release times on identical parallel batching machines. *Discrete Applied Mathematics*, 148:127–134.

Liu Z., J. Yuan, and T.C.E. Cheng, 2003. On scheduling an unbounded batch machine. *Operations Research Letters*, 31:42–48.

Malve S. and R. Uzsoy, 2007. A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. *Computers and Operations Research*, 34(10):3016–3028.

Melouk S., P. Damodaran, and P.Y. Chang, 2004. Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. *International Journal of Production Economics*, 87:141–147.

Mehta S.V. and R. Uzsoy, 1998. Minimizing total tardiness on a batch processing machine with incompatible job families. *IIE Transactions*, 30:165–178.

Perez I.C., J.W. Fowler, and W.M. Carlyle, 2000. *Minimizing total weighted tardiness on a single batch process machine with incompatible job families*. PhD thesis, Arizona State University, Tempe, AZ.

Potts C.N. and M.Y. Kovalyov, 2000. Scheduling with batching : A review. *European Journal of Operational Research*, 120:228–249.

Uzsoy R., 1995. Scheduling batch processing machines with incompatible job families. *International Journal of Production Research*, 33(10):2685–2708.

Wang C.S. and R. Uzsoy, 2002. A genetic algorithm to minimize maximum lateness on a batch processing machine. *Computers and Operations Research*, 29(12):1621–1640.

Webster S. and K.R. Baker, 1995. Scheduling groups of jobs on a single machine. *Operations Research*, 43:692–703.

Zhang G., X. Cai, C.Y. Lee, and C.K. Wong, 2001. Minimizing makespan on a single batch processing machine with nonidentical job sizes. *Naval Research Logistics*, 48:226–240.