Flowshop hybride avec machines à traitement par batch et compatibilité entre les tâches

Adrien Bellanger, Ammar Oulamara

Flowshop hybride avec machines à traitement par batch et compatibilité entre les tâches

Bellanger, A. et Oulamara, A.

MACSI Projet LORIA-INRIA Lorraine, Ecole des Mines de Nancy, Parc de Saurupt, 54042, Nancy
adrien.bellanger@wanadoo.fr ; ammar.oulamara@loria.fr

Résumé Dans ce papier nous étudions le problème d'ordonnancement dans un flowshop hybride à deux étages. Les m_1 machines du premier étage sont des machines parallèles classiques, et les m_2 machines du second étage sont des machines à traitement par batch. Chaque tâche doit être exécutée sur les deux étages, mais il est nécessaire que les tâches d'un même batch, s'exécutant simultanément sur la même machine du second étage, soient compatibles entre elles. Ces relations de compatibilité sont définies par un graphe d'intervalle. Nous nous intéressons ici à la minimisation de la durée totale de l'ordonnancement. Nous proposons des heuristiques avec performances garanties, ensuite des résultats expérimentaux, basés sur des instances générées aléatoirement, permettent d'exhiber l'efficacité de ces heuristiques,.

Mots-Clefs. Flowshop hybride ; Batch ; Heuristique

1 Introduction

Le problème que nous allons aborder dans ce papier est inspiré de l'industrie de pneumatiques. Nous allons commencer par présenter la problématique industrielle, avant de définir le problème simplifié utilisé dans de ce papier, enfin nous abordons l'état de l'art.

1.1 Problème industriel

La fabrication d'un pneu nécessite plusieurs étapes, chaque étape du processus de fabrication requérant une grande précision, et d'importants contrôles. La première étape est la fabrication de la gomme. Elle consiste à mélanger les matières premières (caoutchoucs naturels ou synthétiques et autres additifs), et ajouter du souffre afin de rendre la gomme vulcanisable en fin de fabrication. Lorsque la gomme refroidie est découpée, la deuxième phase consiste à fabriquer les différentes couches qui composent le pneu ; en fonction des cas, il est possible d'ajouter des fils d'acier et des toiles. Ensuite, une opération d'assemblage des différents éléments du pneu permet d'obtenir un *pneu vert* (avant vulcanisation). Finalement le pneu vert est cuit dans des presses de vulcanisation. Ces presses peuvent contenir deux pneus, qui peuvent être de types différents, mais leur temps de cuisson est identique. En effet, lorsque le moule d'une presse est fermé, et la cuisson commencée, nous ne pouvons pas l'ouvrir avant la fin de la cuisson. A chaque type de pneu est associé un intervalle de durée de cuisson qu'il faut impérativement respecter (l'intervalle est donné par une durée minimale, et une durée maximale qui n'est autre que la durée minimale plus 4% de cette durée). La figure 1 représente les deux dernières étapes du système de production, celles que nous allons représenter dans le problème simplifier.

1.2 Problème simplifié

Dans cette section nous allons présenter le problème simplifié que nous étudions par la suite. C'est un problème d'ordonnancement de type flowshop hybride à deux étages. Le premier étage (assemblage) est composé de m_1 machines classiques parallèles. Le second étage (cuisson) est composé de m_2 machines parallèles, où chaque machine est une machine à traitement par batch qui a la capacité de traiter plusieurs tâches à la fois. Nous devons exécuter n tâches sur les deux étages, chaque tâche est exécutée sur l'une des m_1 machines du premier étage, puis sur l'une des m_2 machines du second étage. Chaque tâche, j, possède une durée d'exécution p_{1j} sur le premier étage, et un intervalle de durée opératoire $[a_j; b_j]$, sur le second étage.

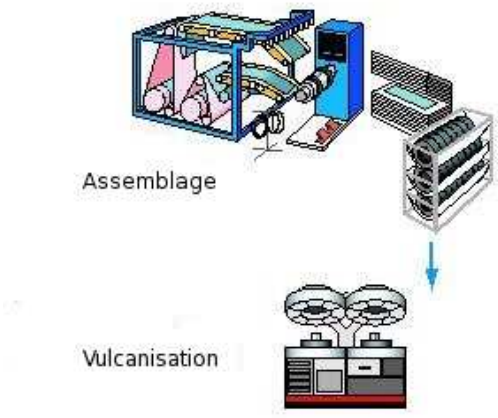

Fig. 1. Etapes étudiées du système de production d'un pneu

Sur le premier étage, une machine n'exécute qu'une tâche à la fois. La tâche j n'est exécutée que sur une seule machine et sans interruption. Lorsque son traitement est terminée au premier étage, elle est exécutée dans un batch sur l'une des machines du deuxième étage. Au second étage, une machine à traitement par batch exécute jusqu'à k tâches simultanément, dans un batch. Pour notre problème, la tâche j peut être exécutée, au second étage, dans un batch dont la durée d'exécution est comprise entre a_j et b_j, point initial et terminal de son intervalle de durée opératoire au second étage. Ainsi l'intersection des intervalles de durées opératoires des tâches d'un même batch B, ne doit pas être vide. La durée d'exécution d'un batch B est notée p_{2B}, avec $p_{2B} = \max_{j \in B} a_j$. Nous cherchons à minimiser la date de fin de l'ordonnancement (makespan).

Pour un ordonnancement donné, la date de fin de la tâche j, $j = 1, ..., n$ est notée C_j. Si la tâche j appartient au batch B du second étage, alors $C_j = C_2(B)$ où $C_2(B)$ désigne la date à laquelle le batch B se termine. Notre objectif est de trouver un ordonnancement tel que sa date de fin (makespan) $C_{max} = \max\{C_j | j = 1, ..., n\}$ soit minimale. Dans ce papier, nous utilisons la notation standard pour un ordonnancement, introduite par Graham et al. [4], c'est à dire $FHB_{m_1,m_2} = F2(m_1, m_2)|p\_batch(II), G_p = INT, k < n|C_{max}$, où $p\_batch(II)$ signifie que le second étage est composé de batching machines en parallèle, $G_p = INT$ spécifie que le graphe de compatibilité entre les tâches est un graphe d'intervalle qui concerne les durées d'exécutions des tâches, et $k < n$ signifie que la capacité des machines à traitement par batch est une donnée du problème.

1.3 Etat de l'art

Dans la littérature, de nombreuses recherches traitent des problèmes d'ordonnancement sur machines à traitement par batch, sans compatibilité entre les tâches et pour diverses fonctions objectifs. Brucker et al. [2], Potts et Van Wassenhove [11], Potts et Kovalyov [10] ont publié l'état de l'art des problèmes d'ordonnancement de machines à traitement par batch. L'article de Finke et al. [3] traite, quant à lui, le problème d'ordonnancement d'une machine à traitement par batch avec compatibilité entre les tâches, dans le cas de graphes généraux et particuliers.

Quant aux flowshops hybrides classiques, la thèse d'Anthony Vignier [13] établit un état de l'art, dans lequel sont présentés différents algorithmes. Les articles de Lee et Vairaktarakis [12], Guinet et al. [6], Allaoui et Artiba[1], Haouari et M'Hallah [7], présentent d'autres algorithmes utilisés pour la résolution du problème de flowshop hybride. Le flowshop hybride avec traitement par batches est peu étudié dans la littérature, toutefois quelques articles traitent de cas particuliers de ce problème, par exemple celui de Shanling Li [14] fournit des heuristiques pour la minimisation du makespan dans un flowshop hybride à deux étages. Dans cet article le premier étage est composé d'une machine, alors que le second étage est composé de machines parallèles à traitement par batch, avec d'importants temps de setup lorsque les tâches ne sont pas compatibles, et un faible temps de setup lorsque les tâches forment un batch. L'article de Narashiram et Mangiameli [15] fournit quant à lui une heuristique pour résoudre le problème de flowshop hybride avec machines parallèles

au premier étage, et machines à traitement par batches identiques au second étage, c'est à dire qu'une machine à traitement par batches du second étage n'effectue qu'un type de batches.

Même si les problèmes de flowshops hybrides et de machines parallèles à traitement par batch ont été l'objet de recherches séparées, la combinaison de ces deux types de problèmes a, à notre connaissance, peu été étudiée précédemment.

Ce papier est organisé comme suit. La section 2 introduit d'autres notations utilisées dans ce papier. La section 3 aborde le problème général, pour lequel nous fournissons quelques bornes inférieures, des heuristiques avec garanties de performance. La section 4 traite quant à elle le cas particulier avec une seule machine au second étage, alors que la section 5 aborde celui où il n'y a qu'une seule machine au premier étage. La section 6 illustre les différentes heuristiques en présentant des résultats expérimentaux. La section 7 conclu quant à elle ce papier.

2 Notations

Pour compléter les notations incluses dans la section précédente nous allons également utiliser :

- $m = \max(m_1; m_2)$.
- $P_h^{(l)}$: la somme des h plus petites durées d'exécution de l'étage l ($l = 1, 2$) ; pour le second étage nous prendrons comme durée d'exécution d'une tâche j, le point initial de son intervalle de durée opératoire, c'est à dire a_j.
- FAM : La règle FAM (First-Available-Machine) est utilisée pour placer une tâche lorsqu'il y a plusieurs machines, elle consiste à placer chaque tâche d'une liste sur la première machine libre.
- FBCLPT : La règle FBCLPT (Full-Batch-Compatible-Longest-Processing-Time) consiste à trier les tâches par durées d'exécution (point initial de l'intervalle de durée opératoire) décroissantes. Ensuite créer un batch vide et ajouter à ce batch la première tâche j de la liste, puis ajouter les $k-1$ premières tâches compatibles avec la tâche j. Retirer de la liste les tâches placées. Réitérer cette création de batch tant qu'il reste des tâches dans la liste. Cette règle donne un ordonnancement optimal pour le problème de minimisation du makespan notée ici $C_{max}(B)$ pour le problème avec une machine à traitement par batches, et compatibilité entre les tâches $(B|G_p = INT, k < n|C_{max})$ (voir A. Kamgaing Kuiten[8]).

3 Plusieurs machines à chaque étage

Dans cette section, nous nous intéressons au cas général où il y a plusieurs machines au premier et au second étage. C'est-à-dire le problème $FHB_{m_1,m_2} = F2(m_1, m_2)|\ p\_batch(II), G_p = INT, k < n|C_{max}$.

3.1 Bornes inférieures

Nous avons développé trois bornes pour le problème FHB_{m_1,m_2}, la première est basée sur une borne inférieure du problème à un seul étage représentant le premier étage, la seconde sur celui représentant le second. La dernière borne inférieure est basée sur le fait que la tâche la plus longue doit être séquencée.

Borne LB_1. La date de fin de l'exécution de toutes les tâches sur le premier étage est supérieure à $\frac{\sum_i p_{1i}}{m_1}$; après l'exécution de la dernière tâche il faut (au minimum) séquencer le batch dans lequel elle est, batch dont la durée est minorée par $\min_j a_j$. D'où la borne :

$$LB_1 = \frac{\sum_i p_{1i}}{m_1} + \min_j a_j$$

Borne LB_2. Un batch est composé au moins d'une tâche, qu'il faut avoir séquencée préalablement sur le premier étage. C'est-à-dire que le premier batch du second étage ne peut pas commencer avant la fin d'exécution d'au moins une tâche, dont la durée est supérieure à $\min_i p_{1i}$. Ensuite tous les batches doivent être exécutés, cette durée d'exécution est, dans le meilleur des cas, la date de fin de la résolution du problème représentant le second étage ($Bm_2|G_p = INT, k < n|C_{max}$). Or le C_{max} fourni par l'exécution de l'algorithme FBCLPT, noté $C_{max}(H_B)$, permet d'obtenir une borne inférieure pour ce problème. D'où la borne :

$$LB_2 = \min_i p_{1i} + \frac{C_{max}(H_B)}{m_2}$$

Borne LB_3. Toute tâche doit être réalisée, ainsi,

$$LB_3 = \max_i \{p_{1i} + a_i\}$$

3.2 Heuristiques avec performances

Heuristique H_{LPT} Dans cette heuristique nous séquençons les tâches sur le premier étage indépendamment du second étage. Les tâches sont séquencées au premier étage suivant les règles LPT et FAM. Pour le second étage nous utilisons les batches fournis par l'algorithme FBCLPT. Lorsque toutes les tâches d'un batch sont réalisées au premier étage, nous plaçons le batch en question sur la première machine libre du second étage.

AlgorithmeH_{LPT}

1. Placer les tâches sur le premier étage, suivant les règles FAM et LPT.
2. Appliquer l'algorithme FBCLPT, soit L la liste de batches obtenue, soit B_i le premier batch de la liste L, $i = 1$.
3. A l'itération i, placer le batch B_i suivant la règle FAM sur le second étage. C'est-à-dire lorsque toutes les tâches de B_i au premier étage sont terminées. Supprimer le batch B_i de la liste L. Si $L = \emptyset$, alors, retourner l'ordonnancement obtenu, sinon $i = i + 1$, aller à 3.

Theorem 1. *L'heuristique H_{LPT} fournit un ordonnancement $S_{H_{LPT}}$ du problème FHB_{m_1,m_2} en $O(nlogn)$, avec un ratio de garantie de performance de $\frac{8}{3} - \frac{2}{3m}$, et cette borne est atteinte.*

Preuve : Soit $S_{H_{LPT}}$ l'ordonnancement obtenu par l'heuristique H_{LPT}. L'algorithme FBCLPT, accompagné de LPBT, fournit un ordonnancement du problème $Bm|G_p = INT, k < n|C_{max}$, noté (Bm), avec une garantie de performance de $\frac{4}{3} - \frac{1}{3m}$ [9]. De plus LPT appliquée au premier étage fournit un ordonnancement de $Pm||C_{max}$, noté (Pm), avec une garantie de performance de $\frac{4}{3} - \frac{1}{3m}$ [5]. D'où :

$$\begin{aligned} C_{max}(S_{H_{LPT}}) &\leq C_{max}(S_{H_{Pm}}) + C_{max}(S_{H_{Bm}}) \\ &\leq \left(\tfrac{4}{3} - \tfrac{1}{3m}\right) C_{max}(S^*_{Pm}) + \left(\tfrac{4}{3} - \tfrac{1}{3m}\right) C_{max}(S^*_{Bm}) \end{aligned}$$

Soit S^* l'ordonnancement optimal du problème de flowshop hybride, et $C_{max}(S^*)$ sa date de fin, alors

$$C_{max}(S^*) \geq C_{max}(S^*_{Pm}) \qquad et \qquad C_{max}(S^*) \geq C_{max}(S^*_{Bm})$$

D'où :

$$\begin{aligned} C_{max}(S_{H_{LPT}}) &\leq \left(\tfrac{4}{3} - \tfrac{1}{3m}\right) C_{max}(S^*_{Pm}) + \left(\tfrac{4}{3} - \tfrac{1}{3m}\right) C_{max}(S^*_{Bm}) \\ &\leq \left(\tfrac{8}{3} - \tfrac{2}{3m}\right) C_{max}(S^*) \end{aligned}$$

Ainsi l'heuristique H_{LPT} fournit une solution avec un facteur de performance de $\frac{8}{3} - \frac{2}{3m}$.

Nous allons présenter un cas pour lequel la borne est atteinte. Nous supposons que le nombre de machine à chaque étage est pair.

Soit I une instance du problème FHB_{m_1,m_2}, la capacité des batches est $2m_1+4$. Cette instance est composée des tâches :

Type	Nombre de tâches	p_1	p_2
A_1	2	$2m_1-1$	$[\epsilon; 2m_2-1]$
A_2	2	$2m_1-2$	$[\epsilon; 2m_2-1]$
...			
A_{m_1-1}	2	m_1+1	$[\epsilon; 2m_2-1]$
A_{m_1}	3	m_1	$[\epsilon; 2m_2-1]$
B_1	2	ϵ	$[2m_2-1; 2m_2-1]$ et $[2m_2-1-\epsilon; 2m_2-1-\epsilon]$
B_2	2	ϵ	$[2m_2-2; 2m_2-2]$ et $[2m_2-2-\epsilon; 2m_2-2-\epsilon]$
...			
B_{m_2-1}	2	ϵ	$[m_2+1; m_2+1]$ et $[m_2+1-\epsilon; m_2+1-\epsilon]$
B_{m_2}	3	ϵ	$[m_2; m_2]$, $[m_2-\epsilon; m_2-\epsilon]$ et $[m_2-2\epsilon; m_2-2\epsilon]$
C	nb_C (entier quelconque)	ϵ	$[\epsilon; \epsilon]$

Dans ce cas l'algorithme FBCLPT fournit un batch contenant la première tâche de type B_1, et les $k-1$ tâches avec lesquelles elle est compatible. C'est-à-dire, toutes les tâches de type A (de A_1 à A_{m_1}). Ensuite les autres tâches de type B n'étant compatibles qu'avec des tâches déjà placées dans un batch, nous n'obtenons que des batches unitaires. Quant aux tâches de type C elles forment $\lceil \frac{nb_C}{k} \rceil$ batches. Afin de simplifier les diagrammes de Gantt nous supposons nb_C nul.

Dans ce cas l'algorithme fournit l'ordonnancement de la figure 2, alors qu'un ordonnancement optimal est celui de la figure 3. En effet la date de fin $C_{max}(S_{H_{LPT}})$ pour l'ordonnancement $S_{H_{LPT}}$ est égale à $C_{max}(S_{H_{LPT}}) = 2m_1 - 1 + m_1 + m_1 + 2m_2 - 1 + m_2 + m_2 - 2\epsilon = 4m_1 + 4m_2 - 2 - 2\epsilon$ (figure 2).

La valeur optimale pour cette instance I est $C_{max}(S^*) = \lceil \frac{m_2}{m_1} + 1 \rceil \epsilon + 3m$ (figure 3).

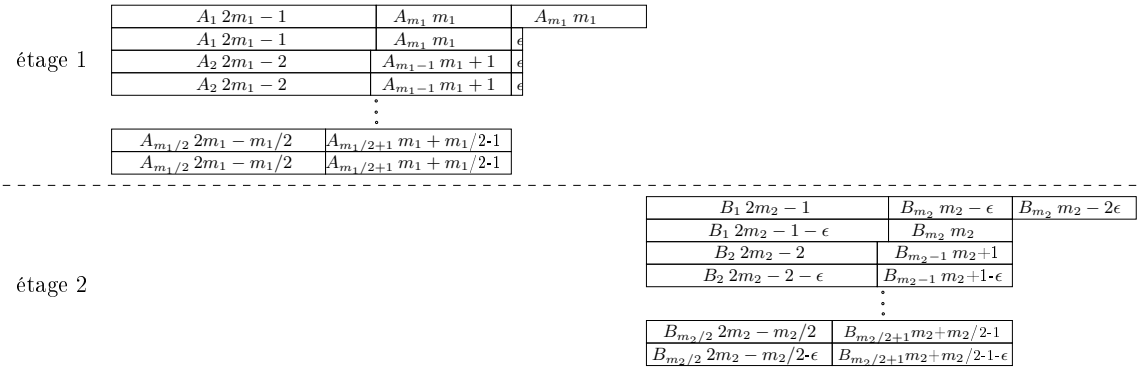

Fig. 2. Ordonnancement $S_{H_{LPT}}$

Si $m_1 \leq m_2$ alors $C_{max}(S^*)$ est optimal car la borne inférieure $LB_1 = \frac{\sum p_{1,i}}{m_1} + \min_j a_j$ est atteinte par $C_{max}(S^*)$. Sinon, à ϵ près, la borne inférieure : $LB_2 = \min_i p_{1i} + \frac{C_{max}(S^*_{Bm})}{m_2}$ est atteinte par $C_{max}(S^*)$.

D'où :

$$\frac{C_{max}(S_{H_{LPT}})}{C_{max}(S^*)} = \frac{4m_1 + 4m_2 - 2 - 2\epsilon}{\lceil \frac{m_2}{m_1} + 1 \rceil \epsilon + 3m}$$

Et

$$\lim_{\epsilon \to 0} \frac{4m_1 + 4m_2 - 2 - 2\epsilon}{\lceil \frac{m_2}{m_1} + 1 \rceil \epsilon + 3m} = \frac{4m_1 + 4m_2 - 2}{3m}$$

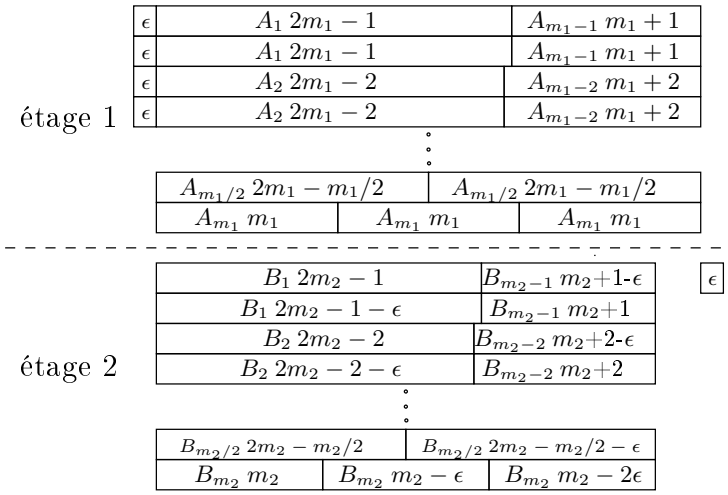

FIG. 3. Ordonnancement optimal

Et donc si $m_1 = m_2$ alors $m = \max(m_1; m_2) = m_1 = m_2$. D'où :

$$\lim_{\epsilon \to 0} \frac{C_{max}(S_{H_{LPT}})}{C_{max}(S^*)} = \frac{4m_1 + 4m_2 - 2}{3m} = \frac{8}{3} - \frac{2}{3m}$$

Si le nombre de machines de l'étage l n'est pas pair, nous obtenons le même résultat. En effet, les tâches centrales ($p_l = 2m_l - (m_l + 1)/2$) de l'étage en question, sont alors séquencées sur la dernière machine de l'étage dans l'ordonnancement optimal, ceci avec la même durée que pour les autres machines. Pour l'ordonnancement fourni par l'heuristique ce sont les 4 tâches centrales ($p_l = 2m_l - (m_l - 1)/2$ et $p_l = 2m_l - (m_l + 1)/2$) qui sont placées sur les deux dernière machines, toujours sans modification de la date de fin par rapport au cas pair. Si $nb_C > 0$, alors les batches correspondants aux tâches de type C sont placés en fin d'ordonnancement. Ainsi lorsque ϵ tend vers 0, la durée de ces tâches ne compte plus.

Heuristique H_{LPBT} Cette heuristique est une amélioration de l'heuristique H_{LPT}, dans la mesure où elle donne en moyenne de meilleurs résultats (voir la section expérimentations). Ceci ne l'empêche pas d'être moins bonne dans le pire cas. En effet dans cette heuristique le traitement du premier étage ne se fait plus suivant LPT sur l'ensemble des tâches, mais pour chaque batch du second étage. C'est à dire que les tâches qui le composent sont séquencées suivant FAM-LPT sur le premier étage.

Algorithme H_{LPBT}

1. Appliquer l'algorithme FBCLPT.
2. Soit L la liste des batches B_i obtenue grâce à l'algorithme FBCLPT, $i = 1$.
3. A l'itération i, soit L_{B_i} la liste des tâches du batch B_i. Séquencer les tâches de L_{B_i} sur les machines du premier étage en suivant les règles FAM et LPT. Placer le batch B_i suivant la règle FAM, c'est-à-dire sur la première machine libre du second étage, lorsque toutes ses tâches sont exécutées au premier étage. Supprimer le batch B_i de la liste L. Si $L = \emptyset$, alors, retourner l'ordonnancement obtenu, sinon $i = i + 1$, aller à 3.

Theorem 2. *L'heuristique H_{LPBT} fournit un ordonnancement $S_{H_{LPTB}}$ du problème FHB_{m_1,m_2} en $O(nlogn)$, avec un ratio de garantie de performance de $\frac{10}{3} - \frac{4}{3m}$.*

Preuve : Soit $S_{H_{LPBT}}$ l'ordonnancement obtenu par l'heuristique H_{LPBT}. L'algorithme FBCLPT, accompagné de LPT, fournit un ordonnancement du problème $Bm|G_p = INT, k < n|C_{max}$, noté (Bm), avec une garantie de performance de $\frac{4}{3} - \frac{1}{3m}$ [9]. En revanche le séquencement sur le premier

étage ne respecte plus LPT. L'heuristique H_{LPBT} fournit seulement un ordonnancement de liste sur le premier étage, la garantie de performance est donc de $2 - \frac{1}{m}$. D'où :

$$C_{max}(S_{H_{LPBT}}) \leq C_{max}(S_{H_{Pm}}) + C_{max}(S_{H_{Bm}})$$

$$\leq \left(2 - \tfrac{1}{m}\right) C_{max}(S^*_{H_{Pm}}) + \left(\tfrac{4}{3} - \tfrac{1}{3m}\right) C_{max}(S^*_{Bm})$$

Soit S^* l'ordonnancement optimal du problème de flowshop hybride, et $C_{max}(S^*)$ sa date de fin, alors

$$C_{max}(S^*) \geq C_{max}(S^*_{H_{Pm}}) \qquad et \qquad C_{max}(S^*) \geq C_{max}(S^*_{Bm})$$

d'où :

$$C_{max}(S_{H_{LPBT}}) \leq \left(2 - \tfrac{1}{m}\right) C_{max}(S^*_{H_{Pm}}) + \left(\tfrac{4}{3} - \tfrac{1}{3m}\right) C_{max}(S^*_{Bm})$$

$$\leq \left(\tfrac{10}{3} - \tfrac{4}{3m}\right) C_{max}(S^*)$$

Ainsi l'heuristique H_{LPBT} fournit une solution avec un facteur de performance de $\frac{10}{3} - \frac{4}{3m}$. Nous n'avons pas trouvé d'exemple atteignant cette borne.

L'heuristique H_J L'algorithme FBCLPT fournit une liste de batches que nous trions selon la règle de Johnson. Les tâches de chaque batch sont placées sur le premier étage suivant les règles FAM et LPT, en respectant l'ordre des batches fourni par la règle de Johnson. Lorsque toutes les tâches d'un batch sont réalisées au premier étage, nous séquençons le batch en question sur la première machine libre du second étage.

Algorithme H_J

1. Appliquer l'algorithme FBCLPT.
2. Soit L la liste des batches B_i obtenue grâce à l'algorithme FBCLPT, triée selon la règle de Johnson, avec pour durée d'exécution du batch B au premier étage $p_{1B} = \frac{\sum_{j \in B} p_{1j}}{m_1}$ et au second étage $\frac{p_{2B}}{m_2}$, $i = 1$.
3. A l'itération i, soit L_{B_i} la liste des tâches du batch B_i. Séquencer les tâches de la liste L_{B_i} sur les machines du premier étage (pour ce séquencement il est possible de prendre les tâches dans l'ordre de leur énumération, ou suivant LPT).
4. Placer le batch B_i suivant la règle FAM. C'est-à-dire sur la première machine libre, lorsque toutes ses tâches sont exécutées au premier étage. Supprimer le batch B_i de la liste L. Si $L = \emptyset$, alors, retourner l'ordonnancement obtenu, sinon $i = i + 1$, aller à 3.

Theorem 3. *L'heuristique H_J fournit un ordonnancement S_{H_J} du problème FHB_{m_1,m_2} en $O(nlogn)$, avec un ratio de garantie de performance de $4 - \frac{2}{m}$.*

Preuve : Soit S_{H_J} l'ordonnancement obtenu par l'heuristique H_J. Considérons le problème de machines parallèles à traitement par batch, noté (Bm). L'algorithme FBCLPT et l'algorithme de Johnson (qui est un algorithme de liste) fournissent un ordonnancement de H_{Bm} : $S_{H_{Bm}}$ en $O(nlogn)$, avec une garantie de performance de $2 - \frac{1}{m}$. L'ordonnancement optimal du problème (Bm) sera noté S^*_{Bm}. De plus, un algorithme de liste fournit un ordonnancement de (Pm) en $O(nlogn)$, avec une garantie de performance de $2 - \frac{1}{m}$. D'où :

$$C_{max}(S_{H_J}) \leq C_{max}(S_{H_{Pm}}) + C_{max}(S_{H_{Bm}})$$

$$\leq \left(2 - \tfrac{1}{m}\right) C_{max}(S^*_{Pm}) + \left(2 - \tfrac{1}{m}\right) C_{max}(S^*_{Bm})$$

Soit S^* l'ordonnancement optimal du problème de flowshop hybride, et $C_{max}(S^*)$ sa date de fin, alors

$$C_{max}(S^*) \geq C_{max}(S^*_{Bm}) \qquad et \qquad C_{max}(S^*) \geq C_{max}(S^*_{Bm})$$

d'où :

$$C_{max}(S_{H_J}) \leq \left(2 - \tfrac{1}{m}\right) C_{max}(S^*_{Pm}) + \left(2 - \tfrac{1}{m}\right) C_{max}(S^*_{Bm})$$

$$\leq \left(4 - \tfrac{2}{m}\right) C_{max}(S^*)$$

Ainsi l'heuristique H_J fournit une solution avec un facteur de performance de $4 - \frac{2}{m}$. Nous n'avons pas trouvé d'exemple atteignant cette borne.

4 Une machine au second étage

Dans cette section nous nous intéressons au problème où il n'y a qu'une machine à traitement par batch au second étage, le premier étage est constitué de m machines parallèles. Ce problème est noté $FHB_{m,1}$, nous pouvons remarquer que les bornes inférieures du problème FHB_{m_1,m_2} sont également valables pour ce problème $FHB_{m,1}$. C'est pourquoi cette section est uniquement consacrée aux différentes heuristiques avec garantie de performance.

Heuristique H_{LPT} Dans cette heuristique, nous séquençons les tâches sur le premier étage indépendamment du second étage, suivant les règles LPT et FAM. Pour le second étage nous utilisons les batches fournis par l'algorithme FBCLPT. Lorsque toutes les tâches d'un batch sont réalisées au premier étage, nous séquençons le batch en question sur la machine du second étage.

Algorithme H_{LPT}

1. Placer les tâches sur le premier étage, suivant les règles FAM-LPT. C'est-à-dire que l'on place la plus grande tâche sur la première machine libre.
2. Appliquer l'algorithme FBCLPT, soit L la liste des batches obtenus.
3. Soit r_j la date de fin d'exécution au premier étage des tâches qui composent le batch j. Séquencer les batches de la liste L dans l'ordre des r_j croissants.

Theorem 4. *L'heuristique H_{LPT} fournit un ordonnancement $S_{H_{LPT}}$ du problème $FHB_{m,1}$ en $O(nlogn)$, avec un ratio de garantie de performance de $\frac{7}{3} - \frac{1}{3m}$, et cette borne est atteinte.*

Preuve : La preuve est similaire à celle du théorème 1, seulement ici nous obtenons la solution optimale pour le problème $B|G_p = INT, k < n|C_{max}$ au second étage (voir la section notation). En effet le ratio $\frac{8}{3} - \frac{2}{3m}$ du théorème 1, est obtenu par l'ajout de $\frac{4}{3} - \frac{1}{3m}$ au second étage, alors que le ratio $\frac{7}{3} - \frac{1}{3m}$ de cette preuve est obtenu par l'ajout de 1 au second étage. Nous exposons ici un exemple permettant d'atteindre la borne $\frac{7}{3} - \frac{1}{3m}$.

Supposons que le nombre de machines du premier étage, m est pair. Soit I une instance du problème $FHB_{m,1}$, pour laquelle la capacité d'un batch est $2m+2$, I est composée des tâches :

Type	Nombre de tâches	p_1	p_2
A_1	2	$2m-1$	$[\epsilon; 3m]$
A_2	2	$2m-2$	$[\epsilon; 3m]$
...			
A_{m-1}	2	$m+1$	$[\epsilon; 3m]$
A_m	3	m	$[\epsilon; 3m]$
B	1	ϵ	$[3m; 3m]$
C	nb_C (entier quelconque)	ϵ	$[\epsilon; \epsilon]$

L'algorithme FBCLPT fournit un batch contenant les tâches de types A et B, il y a ensuite $\lceil \frac{nb_C}{k} \rceil$ batches contenant les tâches de type C. Afin de simplifier les diagrammes de Gantt nous travaillons avec $nb_C = 0$, la généralisation étant immédiate (voir en fin de preuve).

Dans ce cas l'algorithme fournit l'ordonnancement de la figure 4, alors qu'un ordonnancement optimal serait celui de la figure 5. En effet, la date de fin $C_{max}(S_{H_{LPT}})$ pour l'ordonnancement $S_{H_{LPT}}$ est égale à $C_{max}(S_{H_{LPT}}) = 2m - 1 + m + m + 3m = 7m - 1$ (figure 4).

La valeur optimale pour cette instance I est $C_{max}(S^*) = \epsilon + 3m + \epsilon = 3m + 2\epsilon$ (figure 5). $C_{max}(S^*)$ est optimal car la borne inférieure $LB_1 = \frac{\sum p_{1,i}}{m} + \min_j a_j$ est atteinte par $C_{max}(S^*)$.

d'où :

$$\frac{C_{max}(S_{H_{LPT}})}{C_{max}(S^*)} = \frac{7m-1}{3m+2\epsilon}$$

et

$$\lim_{\epsilon \to 0} \frac{7m-1}{3m+2\epsilon} = \frac{7}{3} - \frac{1}{3m}$$

Si le nombre de machines du premier étage n'est pas pair, nous obtenons le même résultat. En effet, les tâches centrales ($p_1 = 2m - (m+1)/2$) du premier étage, sont alors séquencées sur

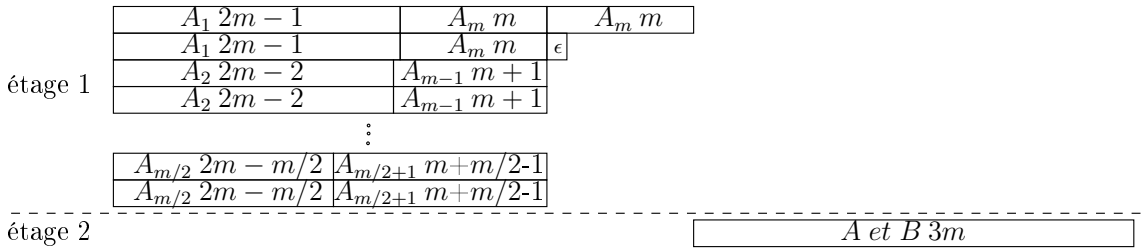

Fig. 4. Ordonnancement $S_{H_{LPT}}$

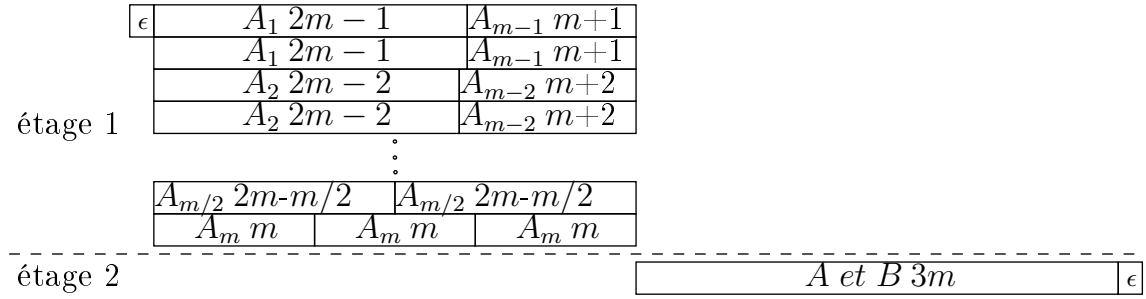

Fig. 5. Ordonnancement optimal

la dernière machine du premier étage dans l'ordonnancement optimal, ceci avec la même durée que pour les autres machines. Pour l'ordonnancement fourni par l'heuristique ce sont les 4 tâches centrales ($p_1 = 2m - (m-1)/2$ et $p_1 = 2m - (m+1)/2$) qui sont placées sur les deux dernière machines, toujours sans modification de la date de fin par rapport au cas pair. Si $nb_C > 0$, alors les batches correspondants aux tâches de type C sont placés en fin d'ordonnancement. Ainsi lorsque ϵ tend vers 0, la durée de ces tâches ne compte plus.

Heuristique H_J L'algorithme FBCLPT fournit une liste de batches, que nous trions selon la règle de Johnson. Les tâches de chaque batch sont séquencées sur les machines du premier étage, en respectant l'ordre des batches fournit par la règle de Johnson. Lorsque les tâches d'un batch sont réalisées au premier étage, nous séquençons le batch sur la machine du second étage.

Algorithme H_J

1. Appliquer l'algorithme FBCLPT.
2. Soit L la liste des batches B_i obtenus grâce à l'algorithme FBCLPT, triée selon la règle de Johnson. Les durées d'exécution du batch B sont $p_{1B} = \frac{\sum_{j \in B} p_{1j}}{m}$ au premier étage, et $p_{2B} = \max_{j \in B} a_j$ au second étage, $i = 1$.
3. A l'itération i, soit L_{B_i} la liste des tâches du batch B_i. Séquencer les tâches de L_{B_i} sur les machines du premier étage (pour ce séquencement il est possible de prendre les tâches dans l'ordre de leur énumération, ou suivant LPT).
4. Placer le batch B_i, dès que la machine du second étage est libre, lorsque toutes ses tâches sont exécutées au premier étage. Supprimer le batch B_i de la liste L. Si $L = \emptyset$ alors retourner l'ordonnancement obtenu, sinon $i = i + 1$, aller à 3.

Theorem 5. *L'heuristique H_J donne un ordonnancement S_{H_J} du problème $FHB_{m,1}$ en $O(nlogn)$, avec un ratio de garantie de performance de $3 - \frac{1}{m}$, et cette borne est atteinte.*

Preuve : La preuve est similaire à celle du théorème 3, seulement ici nous obtenons la solution optimale pour le problème $B|G_p = INT, k < n|C_{max}$ au second étage (voir la section notation). En effet le ratio $4 - \frac{2}{m}$ du théorème 3, est obtenu par l'ajout de $2 - \frac{1}{m}$ au second étage, alors que

le ratio $3 - \frac{1}{m}$ de cette preuve est obtenu par l'ajout de 1 au second étage. Nous exposons ici un exemple permettant d'atteindre la borne $3 - \frac{1}{m}$.

Soit L une constante quelconque, I une instance du problème $FHB_{m,1}$, la capacité d'un batch est $2m$, et l'instance est composée des tâches :

Type	Nombre de tâches	p_1	p_2
A	$m-1$	$(m-1)L$	$[\epsilon; mL]$
B	$m-1$	L	$[\epsilon; mL]$
C	1	mL	$[\epsilon; mL]$
D	1	ϵ	$[mL; mL]$
E	nb_E (entier quelconque)	ϵ	$[\epsilon/2; \epsilon/2]$

L'algorithme FBCLPT fournit un batch regroupant les tâches de type A, B, C et D, ensuite il place les batches de type E dans $\lceil \frac{nb_E}{k} \rceil$ batches. Afin de simplifier les diagrammes de Gantt nous supposons que $nb_E = 0$.

Dans ce cas l'algorithme fournit l'ordonnancement de la figure 6, alors qu'un ordonnancement optimal serait celui de la figure 7. En effet la date de fin $C_{max}(S_{H_J})$ pour l'ordonnancement S_{H_J} est égale à $C_{max}(S_{H_J}) = (m-1)L + mL + mL = (3m-1)L$ (figure 6).

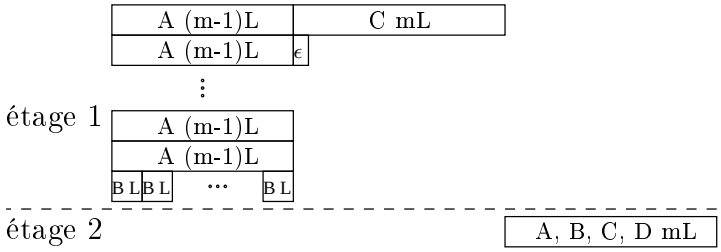

Fig. 6. Ordonnancement S_{H_J}

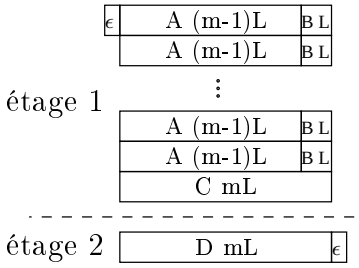

Fig. 7. Ordonnancement optimal

La valeur optimale pour cette instance I est $C_{max}(S^*) = \epsilon + mL + \epsilon = mL + 2\epsilon$ (figure 7). $C_{max}(S^*)$ est optimal car la borne inférieure $LB_1 = \frac{\sum p_{1,i}}{m} + \min_j a_j$ est atteinte par $C_{max}(S^*)$.

d'où :

$$\frac{C_{max}(S_{H_J})}{C_{max}(S^*)} = \frac{3m-1}{m+2\epsilon}$$

et

$$\lim_{\epsilon \to 0} \frac{(3m-1)L}{mL+2\epsilon} = 3 - \frac{1}{m}$$

Si $nb_E > 0$, alors les batches correspondants aux tâches de type E sont placés en fin d'ordonnancement. Ainsi lorsque ϵ tend vers 0, la durée de ces tâches ne compte plus.

5 Une machine au premier étage

Dans cette section, nous nous intéressons au cas où il n'y a qu'une machine au premier étage, C'est-à-dire le problème $FHB_{1,m}$, m désignant le nombre de machines du second étage. Nous pouvons remarquer que les bornes inférieures du problème FHB_{m_1,m_2} sont également valables pour ce problème $FHB_{m,1}$. C'est pourquoi cette section est uniquement consacrée aux différentes heuristiques avec garantie de performance.

5.1 Heuristiques avec performances

Heuristique H_{LPT} Dans cette heuristique, pour chaque batch fourni par l'algorithme FBCLPT (dans l'ordre fourni), nous séquençons les tâches qui le composent sur la machine du premier étage. Lorsqu'elles sont exécutées sur le premier étage, placer le batch sur la première machine libre du second étage.

Algorithme H_{LPT}

1. Appliquer l'algorithme FBCLPT. Nous obtenons la liste L, i=1, soit B_i le premier batch de la liste la liste L.
2. A l'itération i, placer toutes les tâches du batch B_i sur le premier étage, dès que la machine est libre. Placer le batch B_i avec la règle FAM au second étage. C'est-à-dire lorsque toutes les tâches du batch sont terminées au premier étage. Supprimer le batch B_i de la liste L. Si $L = \emptyset$ alors retourner l'ordonnancement obtenu, sinon i=i+1, aller à 2.

Theorem 6. *L'heuristique H_{LPT} fournit un ordonnancement $S_{H_{LPT}}$ du problème $FHB_{1,m}$ en $O(nlogn)$, avec un ratio de garantie de performance de $\frac{7}{3} - \frac{1}{3m}$, cette borne est atteinte.*

Preuve : La preuve est similaire à celle du théorème 1, seulement ici nous obtenons la solution optimale pour le problème $1||C_{max}$ au premier étage. En effet le ratio $\frac{8}{3} - \frac{2}{3m}$ du théorème 1, est obtenu par l'ajout de $\frac{4}{3} - \frac{1}{3m}$ au premier étage, alors que le ratio de $\frac{7}{3} - \frac{1}{3m}$ de cette preuve est obtenu par l'ajout de 1 au premier étage. Nous exposons ici un exemple permettant d'atteindre la borne $\frac{7}{3} - \frac{1}{3m}$

On suppose que le nombre de machine m est pair, et que la capacité des batches est 2. Soit I une instance du problème $FHB_{1,m}$, composée des tâches :

Type	nombre de tâches	p_1	p_2
A_1	2	ϵ	$[2m-1; 2m-1]$ et $[2m-1-\epsilon; 2m-1-\epsilon]$
A_2	2	ϵ	$[2m-2, 2m-2]$ et $[2m-2-\epsilon; 2m-2-\epsilon]$
		...	
A_{m-1}	2	ϵ	$[m+1; m+1]$ et $[m+1-\epsilon; m+1-\epsilon]$
A_m	3	ϵ	$[m; m]$, $[m-\epsilon; m-\epsilon]$ et $[m-2\epsilon; m-2\epsilon]$
B	1	$3m$	$[\epsilon; 2m]$
C	nb_C (entier quelconque)	ϵ	$[\epsilon; \epsilon]$

L'algorithme FBCLPT fournit un batch contenant deux tâches, la première des tâches de type A_1 et la tâche de type B, alors que toutes les autres tâches de type A sont dans des batches unitaires. Quant aux tâches de type C elles forment $\lceil \frac{nb_C}{k} \rceil$ batches. Afin de simplifier les diagrammes de Gantt nous supposons que $nb_C = 0$.

Dans ce cas l'algorithme fournit l'ordonnancement de la figure 8, alors qu'un ordonnancement optimal serait celui de la figure 9. En effet la date de fin $C_{max}(S_{H_{LPT}})$ pour l'ordonnancement $S_{H_{LPT}}$ est égale à $C_{max}(S_{H_{LPT}}) = 3m + \epsilon + 2m - 1 + m - \epsilon + m - 2\epsilon = 7m - 1 - 2\epsilon$ (figure 8). La valeur optimale pour cette instance I est $C_{max}(S^*) = (2m+2)\epsilon + 3m$ (figure 9). $C_{max}(S^*)$ est optimal car la borne inférieure $LB_1 = \sum p_{1,i} + \min_j a_j$ est atteinte par $C_{max}(S^*)$.

d'où :

$$\frac{C_{max}(S_{H_{LPT}})}{C_{max}(S^*)} = \frac{7m - 1 - 2\epsilon}{3m + (2m+2)\epsilon}$$

et

$$\lim_{\epsilon \to 0} \frac{7m - 1 - 2\epsilon}{3m + (2m+2)\epsilon} = \frac{7}{3} - \frac{1}{3m}$$

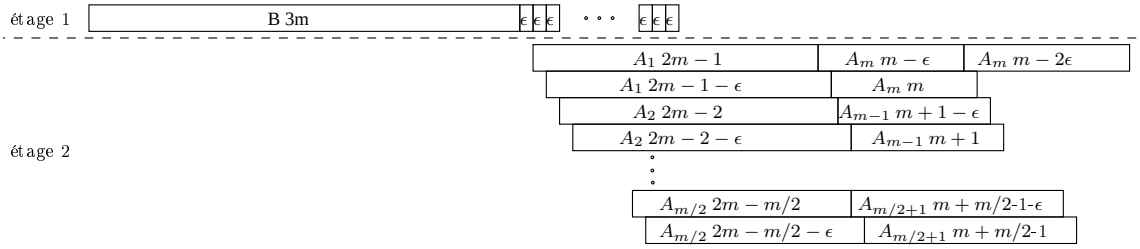

Fig. 8. Ordonnancement $S_{H_{LPT}}$

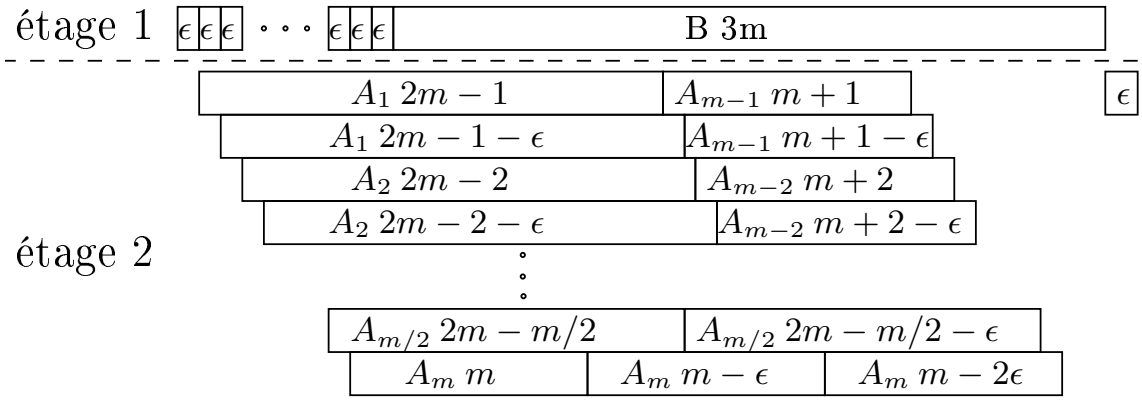

Fig. 9. Ordonnancement optimal

Si le nombre de machines du second étage n'est pas pair, nous obtenons le même résultat. En effet, les tâches centrales $p_2 = 2m - (m+1)/2$ du second étage, sont alors séquencées sur la dernière machine du second étage dans l'ordonnancement optimal, et ceci avec la même durée que pour les autres machines. Pour l'ordonnancement fourni par l'heuristique ce sont les 4 tâches centrales ($p_2 = 2m - (m-1)/2$ et $p_2 = 2m - (m+1)/2$) qui sont placées sur les deux dernière machines, toujours sans modification de la date de fin par rapport au cas pair. Si $nb_C > 0$, alors les batches correspondants aux tâches de type C sont placés en fin d'ordonnancement. Ainsi lorsque ϵ tend vers 0, la durée de ces tâches ne compte plus.

Heuristique H_J L'algorithme FBCLPT fournit une liste de batches que nous trions selon la règle de Johnson. Les tâches de chaque batch sont placées sur la machine du premier étage, en respectant l'ordre des batches fourni par la règle de Johnson. Lorsque les tâches d'un batch sont réalisées au premier étage, nous séquençons le batch sur la première machine libre du second étage.

Algorithme H_J

1. Appliquer l'algorithme FBCLPT.
2. Soit L la liste des batches B_i obtenus grâce à l'algorithme FBCLPT, triée selon la règle de Johnson. Les durées d'exécution du batch B sont $p_{1B} = \sum_{j \in B} p_{1j}$ au premier étage, et $p_{2B} = \frac{\max_{j \in B} a_j}{m}$ au second étage, $i = 1$.
3. A l'itération i, soit L_{B_i} la liste des tâches du batch B_i. Séquencer les tâches de la liste L_{B_i} sur la machine du premier étage, la règle important peu étant donné qu'il n'y a qu'une machine.
4. Placer le batch B_i, sur la première machine libre du second étage, lorsque toutes ses tâches sont exécutées au premier étage. Supprimer le batch B_i de la liste L. Si $L = \emptyset$ alors retourner l'ordonnancement obtenu, sinon $i = i + 1$, aller à 3.

Theorem 7. *L'heuristique H_J fournit un ordonnancement S_{H_J} du problème $FHB_{1,m}$ en $O(nlogn)$, avec un ratio de garantie de performance de $3 - \frac{1}{m}$.*

Preuve : La preuve est similaire à celle du théorème 3, seulement ici nous obtenons la solution optimale pour le problème $1||C_{max}$ au premier étage (voir la section notation). En effet le ratio $4 - \frac{2}{m}$ du théorème 3 est obtenu par l'ajout de $2 - \frac{1}{m}$ au premier étage, alors que le ratio $3 - \frac{1}{m}$ de cette preuve est obtenu par l'ajout de 1 au premier étage.

Nous n'avons pas trouvé d'exemple atteignant cette borne.

6 Expériences Numériques

Pour les différentes heuristiques, nous avons établi la déviation (théorique) par rapport à l'optimum dans le pire cas, mais il est intéressant de connaître en pratique les déviations obtenues en appliquant les algorithmes sur des instances construites aléatoirement. Les expériences concernent le problème général (FHB_{m_1,m_2}). Les différentes instances se différencient par leurs nombres de machines à chaque étage m_1, m_2 et par la structure des intervalles de durées opératoires du second étage. En revanche, pour la capacité des batches, nous préférons nous contenter de celle du problème industriel, c'est à dire $k = 2$. Nous n'avons hélas pas eu la possibilité d'avoir des instances réelles, ainsi nous avons généré pour chaque cas 20 instances aléatoires différentes.

Pour chaque instance générée aléatoirement, les durées opératoires de chaque tâche au premier étage, sont générées dans l'intervalle $[5, 100]$. Les durées opératoires du second étage sont données sous forme d'intervalle $[a_i, b_i]$, avec $b_i = a_i + a_i \times \alpha$, $\alpha \in \{5\%, 15\%, 25\%\}$. Cette structure d'intervalle traduit la tolérance sur le temps de cuisson d'un pneumatique, tolérance dépendant d'une durée minimale de cuisson. La valeur de cette durée minimale (a_i), est générée dans l'intervalle $[5, 100]$, puis $[50, 500]$. Le deuxième cas (l'intervalle $[50, 500]$) traduit le fait que généralement l'assemblage d'un pneu est moins long que sa cuisson.

Quant aux nombres de machines, nous avons examiné trois cas : soit une machine au premier étage et 10 aux second, soit 10 au premier et une seule au second, et le cas où il y a 10 machines à chaque étage. Les expériences sont effectuées avec 100 tâches dans le tableau 1, puis avec 250 tâches dans le tableau 2.

Ne pouvant obtenir la solution optimale, nous utilisons pour la comparaison la meilleure des bornes inférieures présentées précédemment, notée ici LB. Pour comparer la performance de ces heuristiques nous évaluons les distances moyennes D^i_{moy} qui séparent la solution de l'heuristique H_{LPT}, H_{LPBT}, H_J et H_{LPT_a} (une seule machine sur l'un des étages) de la borne inférieure. La distance relative entre la solution π_i fournie par l'heuristique H_i et la meilleure borne inférieure LB est donnée par :

$$D^i_{moy} = \frac{C_{max}(\pi_i) - LB}{LB} \times 100$$

Afin de faciliter la lecture des tables, la distance relative est fournie sous forme de pourcentage. Deux valeurs sont fournies par heuristique, la moyenne des distances relatives, et la pire des distances relatives.

La lecture des résultats nous permet de remarquer que l'heuristique H_{LPT} est la moins intéressante de toutes, avec une erreur moyenne proche de 50 % lorsqu'il y a 10 machines à chaque étage. Même lorsqu'il n'y a qu'une machine sur l'un des étages, et que l'on utilise H_{LPT_a} (c'est à dire H_{LPT} adapté aux deux cas particuliers où il n'y a qu'une machine sur l'un des étages), elle reste moins efficace que les autres. En revanche départager les heuristiques H_{LPBT} et H_J est plus compliqué : en effet lorsque les durées opératoires du second étage sont supérieures à celles du premier ($p_{2i} \in [50, 500]$), et qu'il n'y a pas trop de tâches (n=100) H_{LPBT} semble meilleur que H_J alors que c'est l'inverse dans tous les autres cas. Ce qui permet à l'heuristique H_{LPBT} d'avoir une pire déviation 12,3% moins importante que celles de l'heuristique H_J 20,0% pour les instances étudiées.

Nous observons également, que hormis pour l'heuristique H_{LPT} l'augmentation du nombre de tâches entraîne une diminution de la déviation. Pour toutes les heuristiques présentées, la déviation est plus importante lorsqu'il y a 10 machines sur chaque étage. En revanche, la longueur des intervalles de durées opératoires sur le second étage n'influe pas les résultats de manière significative.

Ainsi, lorsqu'il y a 100 tâches, les pires déviations des algorithmes H_{LPBT} et H_J sont proches de 12% lorsque $p_{2i} \in [5; 100]$, et respectivement de 10% et 20% lorsque $p_{2i} \in [50; 500]$. Pour les mêmes instances les déviations de l'algorithme H_{LPT} sont respectivement de 53% et 45%. Avec

$p_{2i} \in$	$m_1 - m_2$	α	H_{LPT}		H_{LPT_a}		H_{LPBT}		H_J	
			moy	pire	moy	pire	moy	pire	moy	pire
[5, 100]	1-10	0.05	2,962	4,007	0,244	0,273	0,244	0,273	0,244	0,273
		0,15	2,747	3,786	0,249	0,284	0,249	0,284	0,249	0,284
		0,25	2,434	3,481	0,253	0,283	0,253	0,283	0,253	0,283
	10-1	0,05	15,693	19,783	4,683	7,795	1,739	2,765	0,233	0,701
		0,15	14,552	18,804	6,103	9,187	1,848	3,316	0,255	0,613
		0,25	15,152	19,154	5,918	8,399	1,965	2,990	0,287	0,755
	10-10	0,05	40,713	53,332	-	-	8,326	12,101	8,338	11,933
		0,15	39,363	48,178	-	-	8,790	12,255	8,690	11,928
		0,25	38,446	50,842	-	-	7,559	9,760	7,480	9,955
[50, 500]	1-10	0,05	14,337	17,475	0,264	0,323	0,264	0,323	0,264	0,323
		0,15	14,105	18,844	0,282	0,435	0,282	0,435	0,282	0,435
		0,25	13,115	17,538	0,298	0,466	0,298	0,466	0,298	0,466
	10-1	0,05	2,879	4,075	0,791	1,405	0,408	0,703	0,060	0,176
		0,15	2,382	3,272	1,181	2,207	0,421	0,699	0,077	0,300
		0,25	1,860	3,244	1,393	1,950	0,494	0,779	0,067	0,177
	10-10	0,05	35,627	43,105	-	-	7,516	9,135	10,779	16,966
		0,15	36,412	44,111	-	-	8,037	9,570	12,528	18,478
		0,25	37,769	44,737	-	-	8,143	10,127	10,637	19,971

TAB. 1. D_{moy} pour chaque heuristique, pour n=100

$p_{2i} \in$	$m_1 - m_2$	α	H_{LPT}		H_{LPT_a}		H_{LPBT}		H_J	
			moy	pire	moy	pire	moy	pire	moy	pire
[5, 100]	1-10	0.05	3,862	4,242	0,096	0,102	0,096	0,102	0,096	0,102
		0,15	3,615	4,431	0,097	0,101	0,097	0,101	0,097	0,101
		0,25	3,632	4,402	0,097	0,102	0,097	0,102	0,097	0,102
	10-1	0,05	17,346	19,327	4,507	5,922	0,940	1,244	0,064	0,277
		0,15	17,462	20,333	4,540	6,407	0,791	1,269	0,066	0,190
		0,25	17,396	19,242	4,802	6,170	0,862	1,234	0,083	0,185
	10-10	0,05	43,802	48,198	-	-	3,230	5,270	3,250	5,391
		0,15	42,032	46,273	-	-	3,381	4,370	3,395	4,242
		0,25	42,600	46,323	-	-	3,162	4,496	3,166	4,560
[50, 500]	1-10	0,05	16,885	19,006	0,103	0,122	0,103	0,122	0,103	0,122
		0,15	16,676	18,983	0,106	0,174	0,106	0,174	0,106	0,174
		0,25	16,799	18,556	0,109	0,170	0,109	0,170	0,109	0,170
	10-1	0,05	3,199	4,249	0,626	1,004	0,220	0,304	0,015	0,044
		0,15	3,415	4,035	0,743	1,221	0,219	0,311	0,019	0,045
		0,25	3,127	4,226	0,808	1,578	0,189	0,298	0,015	0,045
	10-10	0,05	38,351	44,161	-	-	3,123	3,608	1,342	3,138
		0,15	40,447	43,943	-	-	2,935	3,540	1,409	2,361
		0,25	39,356	44,929	-	-	3,101	3,593	1,353	2,175

TAB. 2. D_{moy} pour chaque heuristique, pour n=250

250 tâches, les pires cas des deux heuristiques H_{LPBT} et H_J, ne sont plus que de 3% (et 4% pour H_{LPBT} avec $p_{2i} \in [50;500]$) Le passage à 250 tâches ne permet pas réellement à H_{LPT} de s'améliorer avec 48% et 45% pour $p_{2i} \in [5;100]$ et $p_{2i} \in [50;500]$.

7 Conclusion et perspectives

Dans ce papier nous avons étudié un problème d'ordonnancement de type flowshop hybride issu d'une application industrielle. Le problème est NP-difficile pour la minimisation du makespan. Afin de contourner cette difficulté, nous avons proposé des approches heuristiques avec une garantie de performance. Bien que en théorie, le pire cas obtenu par les heuristiques est plus de trois fois l'optimum, les expérimentations que nous avons menées montrent que ces heuristiques donnent des résultats assez intéressants. Nous obtenons, en moyenne pour le cas où il y a 10 machines à chaque étage, des solutions à moins de 12% de la meilleure borne inférieure, donc dans le pire des cas à 12% de l'optimum.

Les perspectives de ce travail sont donc nombreuses, par exemple améliorer les heuristiques, construire une méthode exacte. De plus, même si le problème que nous avons étudié est proche du problème industriel, ce dernier prend en compte des temps de setup (changement de moule dans les presses de vulcanisation), que nous n'avons pas traité ici, et qu'il faudrait inclure dans la perspective d'une solution industrielle.

Références

1. Hamid Allaoui, and Abdelhakim Artiba. Scheduling two-stage hybrid flow shop with availability constraints. *Computers & Operations Research*, 33 : 1399-1419 (2006).
2. P. Brucker, A. Gladky, J.A. Hoogeveen, M.Y. Kovalyov, C.N. Potts, T. Tautenhahn,and S.L. Van de Velde. Scheduling a batch processing machine *Journal of Scheduling*,1 : 31-54 (1998).
3. G. Finke, V. Jost, M. Queyranne, and A. Sebo. Batch processing with interval graph compatibilities between tasks. In *Second International Workshop on Discrete Optimization Methods and Logistics*, , pages 20-27, Omsk-Russia, July 2004.
4. R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rimooy Kan. Optimization and approximation in deterministic sequencing and scheduling : a survey. *Annals of Discrete Mathematics*, 5 : 287-326 (1979).
5. R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17 : 263-269 (1969).
6. A. Guinet, M.M. Solomon, P.K. Kedia, and A. Dussauchoy. A computational study of heuristics for two-stage flexible flowshop *Int. J. Prod. Res.*, 34(5) : 1399-1415 (1996).
7. Mohamed Haouari, and Rym M'Hallah. Heuristic algorithms for the two-stage hybrid flowshop problem. *Operations Research Letters*, 21 : 43-53 (1997).
8. Aimé Kamgaing Kuiten, Ammar Oulamara, and Gerd Finke. Flowshop Scheduling problem with Batching Machine and Task Compatibilities. INCOM, 2006.
9. Chung-Yee Lee, Reha Uzsoy, and Louis A. Martin-Vega. Efficient algorithms for scheduling semiconductor burn-in operations, *Operations Research* 40(4) :764-775, July-August 1992.
10. C.N. Potts, and M.Y. Kovalyov. Scheduling with batching : a review. *European journal of Operational Research*, 120 : 228-249 (2000).
11. C.N. Potts, and L.N. Van Wassenhove. Integrating scheduling with batching and lotsizing : a review of algorithm and complexity *Journal of Operational Research Society*, 43 : 395-406 (1992).
12. Chung-Yee Lee, and L. Vairaktarakis. Minimizing makespan in hybrid flowshop. *Operations Research Letters*, 16 : 149-158 (1994).
13. Anthony Vignier. *Contribution à la résolution des problèmes d'ordonnancement de type monogamme, multimachine ("flow-shop hybride")* Université de Tours (1997).
14. Shanling Li. A hybrid two-stage flowshop with part family, batch production, major and minor set-ups. *European Journal of Operational Research*, 102 : 142-156 (1997).
15. S.L. Narasimhan and P.M. Mangiameli. A comparison of sequencing rules for two stage hybrid flow shop. *Decision Sciences*, 18 : 250-265 (1987)