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»

ORDONNANCEMENT DE MACHINES PARALLÈLES IDENTIQUES EN PRÉSENCE D'OPÉRATEURS AVEC DES CHANGEMENTS CALENDAIRES

M. Zouba, P. Baptiste

Dépt. de mathématiques et de génie industriel
École Polytechnique de Montréal
Québec, Canada.
mohammed.zouba@polymtl.ca, pbaptiste@polymtl.ca

D. Rebaine

Dépt. d'informatique et de mathématique
Université du Québec à Chicoutimi
Québec, Canada.
drebaine@uqac.ca

RÉSUMÉ : *Cet article introduit le problème d'ordonnancement de n tâches non préemptives sur m machines parallèles identiques en présence d'opérateurs pour minimiser le critère du makespan. Le nombre d'opérateurs est inférieur à celui des machines, mais un opérateur peut en superviser plusieurs à la fois. Le temps d'exécution des tâches dépend alors de l'affectation des opérateurs aux machines. Les affectations sont réévaluées de façon calendaire, c'est-à- dire en fin de période. Nous présentons dans cet article des heuristiques ainsi qu'une analyse dans le pire cas pour une d'entre elles. Une simulation est également entreprise pour évaluer la performance de ces heuristiques dans le cas moyen.*

MOTS-CLÉS : *Ordonnancement, machines parallèles, makespan, opérateur, changement calendaire.*

1. INTRODUCTION

Nous considérons, dans cet article, le problème d'ordonnancement de n tâches sur m machines parallèles identiques en présence d'opérateurs pour minimiser le critère du makespan. Dans ce modèle, chaque opérateur se voit confier la supervision simultanée de plusieurs machines, induisant par là des temps d'exécution variables pour ces n tâches.

Jusqu'à un passé récent, les travaux de recherche en ordonnancement de production se sont concentrés sur les contraintes liées aux machines et aux tâches : disponibilité des machines, ordre d'exécution des tâches, etc. En effet, la complexité des problèmes d'ordonnancement d'atelier dans la réalité et la difficulté d'offrir un cadre théorique intégrant l'ensemble de la réalité industrielle ont conduit à des hypothèses simplificatrices (Pinedo, 2002). Ces modèles se sont donc limités au séquencement des tâches et relâchent, en général, toutes les contraintes liées à la gestion des opérateurs dans l'atelier, en supposant que leur nombre est infini. Cependant, force est de constater que les ressources humaines sont de plus en plus critiques dans les systèmes de production. Il semble aujourd'hui peu pertinent de se concentrer exclusivement sur les ressources physiques.

L'approche traditionnelle d'ordonnancement d'ateliers, tenant compte des ressources humaines, consiste à résoudre le problème en deux phases : l'ordonnancement de l'atelier est d'abord réalisé et, ensuite, les opérateurs sont affectés aux machines, en modifiant localement, si nécessaire, la solution de la première phase (Hu, 2005 et 2004; Huq *et al.*, 2004), ou vice versa comme cela est fait dans Karlof et Wang (1996).

Dans Pinedo (2002), il est souligné l'intérêt d'étudier les modèles combinant l'ordonnancement des machines à celui des opérateurs. Ces problèmes d'ordonnancement intégrant l'affectation des opérateurs aux machines consistent à déterminer l'état de l'atelier au cours du temps (Cheurfa, 2005). Bourland et Carl (1994), quant à eux, se sont intéressés à l'ordonnancement de machines parallèles avec des ressources humaines partagées, en ne considérant que l'utilisation d'une fraction de la ressource. L'affectation des opérateurs tient compte de la compétence de ces derniers ainsi que des contraintes sociales liées à la législation du travail.

Depuis les travaux de Vickson (1980a, 1980b), de nombreuses études ont porté sur des modèles d'ordonnancement avec des temps d'exécution contrôlables par la quantité de ressources allouée aux tâches (Daniels *et al.*, 1996; Grigoriev *et al.*, 2005; Nowicki et Zdrzalka, 1990). La majorité de ces

travaux porte sur l'analyse de la complexité algorithmique et l'application d'heuristiques, étant donné que ces problèmes, sans la contrainte de ressources, sont déjà $\mathcal{NP}$-Complets. Daniels *et al.* (1996) ont étudié le problème d'ordonnancement de machines parallèles avec pré-affectation de tâches. Ils ont développé un algorithme de branch & bound qui énumère toutes les affectations possibles des ressources et calcule, pour une affectation fixée, des bornes inférieures pour le makespan. Grigoriev *et al.* (2005) considèrent le problème d'ordonnancement sur des machines parallèles non dépendantes où les temps d'exécution sont fonction du nombre d'opérateurs assignés à une machine à un instant donné. Ils utilisent une technique en deux phases, basée sur la programmation linéaire, pour assigner les opérateurs aux tâches et les tâches aux machines, de façon à minimiser le makespan. Ils produisent une borne inférieure pour ce critère ainsi que des méthodes heuristiques. Nowicki et Zdrzalka (1990), quant à eux, présentent un état de l'art sur cette problématique. Par ailleurs, Cheurfa (2005) a étudié le problème d'intégration d'opérateurs en ordonnancement de production cyclique. Il considère l'affectation des opérateurs aux machines et non pas aux tâches. Les durées d'exécution des tâches sont donc variables et sont fonction de l'affectation des opérateurs aux machines retenues.

Dans cet article, nous commençons par proposer un modèle décrivant les modifications des durées d'exécution dues au fait qu'un opérateur peut s'occuper de plusieurs machines en parallèle. Nous supposons que la dégradation de la durée d'exécution est linéaire (indépendante de la tâche réalisée). Ainsi, une affectation donnée d'un opérateur sur un sous-ensemble de machines peut être vue comme un vecteur de coefficients de pondération des durées effectuées sur chacune des machines. D'autre part, nous considérons que si un opérateur peut s'occuper de plusieurs machines en parallèle, il peut le faire de plusieurs manières, en donnant des priorités différentes aux machines. Il peut donc exister plusieurs affectations (vecteurs de coefficients de pondération) pour le même groupe de machines. Cette étude fait suite au travail de Zouba *et al.* (2006). En effet, ces auteurs considèrent la même problématique que celle que nous présentons dans ce présent article. Néanmoins, dans leur article, ils présentent des propriétés caractérisant la solution minimisant le makespan en considérant le changement d'affectation des opérateurs possible à tout instant. Dans ce présent travail, ces mêmes auteurs considèrent un autre mode de changement de supervision, qui peut se faire seulement à des dates fixes, et présentent des heuristiques dont une est suivie d'une analyse dans le pire cas.

Cet article est organisé comme suit. La Section 2 décrit le problème étudié. La Sections 3 est consacrée à la présentation de trois heuristiques dans le cas où l'affectation des tâches aux machines est connue. La Section 4 présente une autre heuristique, suivie de son analyse dans le pire cas, dans le cas où la supervision des opérateurs est connue pour chaque période de travail. La Section 5 présente une autre heuristique dans le cas où l'affectation des tâches aux machines et celle des opérateurs ne sont pas connues a priori. La Section 6 est consacrée aux résultats obtenus par la simulation effectuée sur la performance de ces heuristiques. La Section 7 est notre conclusion.

2. DESCRIPTION DU PROBLÈME

Le problème que nous considérons dans cet article consiste à réaliser un ensemble de n tâches $J = (J_i : i = 1, \ldots, n)$ sur un ensemble $M = (M_j : j = 1, \ldots, m)$ de m machines parallèles identiques. Il est supposé que les n tâches et les m machines sont disponibles, d'une manière continue, dès l'instant $t = 0$. Pour qu'une tâche, J_i, puisse être exécutée sur une machine, il est nécessaire que cette dernière soit supervisée par un opérateur. Le temps d'exécution de la tâche J_i est p_i, connu à l'avance, si la machine sur laquelle elle s'exécute est supervisée exclusivement par un opérateur. Ce temps est appelé temps d'exécution de base de la tâche J_i. Nous supposons que le nombre d'opérateurs en présence dans l'atelier est inférieur au nombre de machines. Ceci fait que chaque opérateur doit s'occuper de plusieurs machines à la fois. La productivité de chacune de ces machines peut se trouver ainsi affectée. Autrement dit, les temps d'exécution varient selon le taux de productivité induit par le mode de fonctionnement (affectation) choisi. Notons que, pour une même disposition des opérateurs, plusieurs vecteurs de productivité peuvent être définis.

En effet, soit x_{kj} le taux de productivité de la machine m_j pour une affectation AT_k des opérateurs. Une affectation AT_k, $k = 1, \ldots, \ell$ (ℓ étant le nombre d'affectations), peut être définie comme suit :

$$AT_k = (x_{kj} : j = 1, \ldots, m).$$

Pour une affectation AT_k, le temps d'exécution de base, p_i, de la tâche J_i sur la machine m_j devient donc $p_{ikj} = p_i / x_{kj}$.

L'ordonnancement en présence d'opérateurs est entièrement caractérisé par la connaissance du séquencement des tâches sur chacune des machines et les différentes affectations utilisées par les opérateurs.

Définition 1 *Le taux global de service de l'affecta-*

tion AT_k est

$$\tau_k = \sum_{j=1}^{m} x_{kj}.$$

Remarquons que la valeur de τ_k peut être supérieure au nombre d'opérateurs.

L'affectation d'un opérateur peut changer au cours du temps. On distingue trois modes de changement, à savoir le mode libre, calendaire et en fin de tâche. Dans le mode libre, l'affectation des opérateurs peut changer à tout instant. Dans le mode calendaire, la durée des affectations est un multiple d'une période : journée, demi-journée, etc. En fin de tâche, l'opérateur peut changer d'affectation dès qu'il termine une des tâches dont il s'occupe.

La problématique consiste donc, selon le mode de changement choisi, à déterminer des intervalles de temps d'utilisation des différentes affectations et ordonnancer l'ensemble des tâches sur les machines de façon à minimiser le makespan. Dans ce présent travail, nous nous intéressons au mode de changement calendaire.

3. CAS OÙ L'AFFECTATION DES TÂCHES AUX MACHINES EST CONNUE

Le problème dans cette section est de déterminer les affectations des opérateurs à utiliser de façon à minimiser le makespan, en supposant que l'affectation des tâches aux machines est connue. Notons par P_j, $j = 1, \ldots, m$, la somme des temps d'exécution des tâches affectées à la machine m_j. De par son fonctionnement, le mode de changement calendaire induit un certain nombre de périodes de travail pour les opérateurs qui a une influence directe sur la valeur du makespan. Remarquons qu'il est facile de montrer que ce problème est équivalent à celui du sac à dos multidimensionnel à variables entières, d'où sa $\mathcal{NP}$-complétude. Avant d'aller plus loin, nous présentons ci-dessous la formulation mathématique de ce problème.

Soient donc ℓ le nombre d'affecations, Y_k le nombre de périodes où l'affectation AT_k est utilisée, et T la durée d'une période. Le problème de minimisation du nombre de périodes, quand l'affectation des tâches aux machines est donnée, peut être formulé comme suit :

$$\begin{aligned} & \min \sum_{k=1}^{\ell} Y_k \\ \text{sujet à :} \quad & \\ & \sum_{k=1}^{\ell} T x_{kj} Y_k \geq P_j; j = 1, \ldots, m \\ & \\ & Y_k \geq 0, \text{ entier.} \end{aligned} \qquad (P1)$$

Proposition 1 *Si PN représente le nombre minimum de périodes à utiliser, alors*

$$PN \geq \frac{\sum_{j=1}^{m} P_j}{T \max\limits_{k=1,\ldots\ell} \tau_k}.$$

Preuve : Si $C_{\max}(S)$ est la valeur du makespan d'un ordonnancement quelconque S, alors il est clair :

$$\begin{aligned} PN &\geq \frac{C_{\max}(S)}{T} \\ &\geq \frac{\sum_{j=1}^{m} P_j}{T \max\limits_{k=1,\ldots\ell} \tau_k}. \end{aligned}$$

D'où le résultat.

Il est utile de remarquer que si on connaît la valeur du makespan, alors on connaît également le nombre de périodes utilisées. Par conséquent, le problème de minimisation du makespan est au moins aussi difficile que le problème de recherche du nombre de périodes. Ce problème est donc $\mathcal{NP}$-difficile.

Propriété 1 *Si la durée des affectations est un nombre entier de périodes de durée constante, alors pour un ensemble donné d'affectations et de nombre de périodes correspondantes, le makespan de la solution correspondante ne dépend que de la dernière affectation.*

Il s'ensuit alors que les séquences ne sont pas indépendantes. Toutefois, un simple index permet de trouver l'affectation avec laquelle il faut finir. Il sera donc suffisant de se contenter de chercher les affectations, et l'ordre sera donné par la suite. L'affectation AT_k à mettre en dernier est celle qui minimise l'expression suivante :

$$\max_j \left(\frac{P_j - T\left(\sum_{h \neq k} Y_h x_{hj} - (Y_k - 1) x_{kj}\right)}{x_{kj}} \right).$$

Autrement dit, le temps maximal restant sur une des machines. En effet, la différence porte simplement sur le fait que la dernière période n'est pas entièrement utilisée.

Théorème 1 *Soient $AT_1 = (x_{11}, x_{12}, \ldots, x_{1m})$ et $AT_2 = (x_{21}, x_{22}, \ldots, x_{2m})$ deux affectations et T la longueur d'une période. Soit une partition telle que*

$$\max\left(\frac{P_1}{x_{11}}, \ldots, \frac{P_m}{x_{1m}}\right) \leq \max\left(\frac{P_1}{x_{21}}, \ldots, \frac{P_m}{x_{2m}}\right). \quad (1)$$

Si $C_{\max}(S1)$ et $C_{\max}(S2)$ désignent, respectivement, le makespan des solutions $S1$ et $S2$ utilisant seulement AT_1 et AT_2, alors $C_{\max}(S1) \leq C_{\max}(S2)$.

Preuve : Par définition, nous avons

$$C_{\max}(S1) = \max\left(\frac{P_1}{x_{11}}, \ldots, \frac{P_m}{x_{1m}}\right),$$

et

$$C_{\max}(S2) = \max\left(\frac{P_1}{x_{21}}, \ldots, \frac{P_m}{x_{2m}}\right).$$

D'où le résultat.

Théorème 2 *Soient $AT_1 = (x_{11}, x_{12}, \ldots, x_{1m})$ et $AT_2 = (x_{21}, x_{22}, \ldots, x_{2m})$ deux affectations et T la longueur d'une période. Soit une partition telle que*

$$\max\left(\frac{P_1}{x_{11}}, \ldots, \frac{P_m}{x_{1m}}\right) \leq \max\left(\frac{P_1}{x_{21}}, \ldots, \frac{P_m}{x_{2m}}\right). \quad (2)$$

Si $C_{\max}(S1)$ désigne le makespan de la solution $S1$ utilisant AT_1 en première période et AT_2 pour le reste des périodes, et $C_{\max}(S2)$, le makespan de la solution $S2$ utilisant seulement AT_2, alors $C_{\max}(S1) \leq C_{\max}(S2)$.

Preuve : Sans perte de généralité, nous supposons

$$\max\left(\frac{P_1}{x_{21}}, \ldots, \frac{P_m}{x_{2m}}\right) = \frac{P_k}{x_{2k}}. \quad (3)$$

Soient $CM_j(S1)$ et $CM_j(S2)$ les temps de fin d'exécution de la machine m_j pour les solutions $S1$ et $S2$, respectivement . Si $\frac{P_j}{x_{1j}} \leq \frac{P_j}{x_{2j}}$, c'est-à-dire $x_{1j} \geq x_{2j}$, alors $CM_j(S1) \leq CM_j(S2)$. Sinon $CM_j(S1) > CM_j(S2)$. Dans ce dernier cas, nous allons comparer $CM_j(S1)$ à $C_{\max}(S2)$.

Des équations (2)-(3) et $\frac{P_j}{x_{1j}} > \frac{P_j}{x_{2j}}$, il s'ensuit les inégalités suivantes :

$$\frac{P_j}{x_{2j}} < \frac{P_j}{x_{1j}} < \frac{P_k}{x_{2k}}.$$

Évaluons maintenant $CM_j(S1) - C_{\max}(S2)$.

Nous avons

$$C_{\max}(S2) = \frac{P_k}{x_{2k}},$$

et

$$CM_j(S1) = \frac{P_j - Tx_{1j}}{x_{2j}} + T.$$

Par conséquent,

$$\begin{aligned} CM_j(S1) - C_{\max}(S2) &= \frac{P_j - Tx_{1j}}{x_{2j}} + T - \frac{P_k}{x_{2k}} \\ &= \frac{P_j x_{2k} - Tx_{1j}x_{2k}}{x_{2j}x_{2k}} + \\ &\quad \frac{Tx_{2j}x_{2k} - P_k x_{2j}}{x_{2j}x_{2k}} \\ &= \frac{P_j x_{2k} - P_k x_{2j}}{x_{2j}x_{2k}} + \\ &\quad \frac{Tx_{2k}(x_{2j} - x_{1j})}{x_{2j}x_{2k}}. \end{aligned}$$

Or,

$$x_{2j} - x_{1j} \geq 0 \text{ et } T \leq \frac{P_k}{x_{2k}}.$$

Donc,

$$\begin{aligned} CM_j(S1) - C_{\max}(S2) &\leq \frac{P_j x_{2k} - P_k x_{2j}}{x_{2j}x_{2k}} + \\ &\quad \frac{P_k(x_{2j} - x_{1j})}{x_{2j}x_{2k}} \\ &\leq \frac{P_j x_{2k} - P_k x_{1j}}{x_{2j}x_{2k}} \\ &< 0 \text{ car } \frac{P_j}{x_{1j}} < \frac{P_k}{x_{2k}}. \end{aligned}$$

Par conséquent,

$$CM_j(S1) < C_{\max}(S2).$$

Ce qui donne

$$C_{\max}(S1) \leq C_{\max}(S2).$$

D'où le résultat.

Dans ce qui suit, nous proposons trois heuristiques pour résoudre le problème que nous considérons dans cette section. Notons que les heuristiques H1 et H2 peuvent être implémentées en $O(PN \times \ell \times m)$, tandis que la complexité temporelle de l'heuristique H3 est dominée par la résolution du programme linéaire relâché $P1$.

• **Heuristique H1** : Cette heuristique s'inspire du résultat du Théorème 2 en utilisant, à chaque période, l'affectation pour laquelle le temps maximal restant sur une quelconque des machines est minimum. Si l'on dispose d'une seule affectation, alors H1 génère la solution optimale. Cette heuristique peut être décrite comme suit :

À la période $i; i = 1, 2, \ldots$, choisir une affectation AT_k telle que l'expression suivante soit minimisée :

$$\max_j \left(\frac{\max\left(0, P_j - T\sum_{p=1}^{i-1} x_{k_p j}\right)}{x_{kj}}\right).$$

$AT_{k_p} = (x_{k_p j} : j = 1, \ldots, m)$ est l'affectation choisie à la période p.

• **Heuristique H2 :** Cette heuristique minimise, à chaque période, la somme quadratique des temps restants sur chacune des machines. Elle permet ainsi d'équilibrer la charge des machines. Elle peut être décrite comme suit :

À la période i, $i = 1, 2, \ldots$, choisir une affectation AT_k telle que l'expression suivante soit minimisée :

$$\sum_{i=1}^{m} \left(\frac{\max(0, P_j - T \sum_{p=1}^{i-1} x_{k_p j})}{x_{kj}} \right)^2 .$$

$AT_{k_p} = (x_{k_p j} : j = 1, \ldots, m)$ est l'affectation choisie à la période p.

Remarquons que pour les heuristiques H1 et H2, les affectations avec $x_{kj} = 0$ ne sont considérées que lorsque $\max(0, P_j - T \sum_{p=1}^{i-1} x_{k_p j}) = 0$.

• **Heuristique H3 :** Cette heuristique utilise la relaxation du programme linéaire en nombres entiers ($P1$) pour construire une solution partielle complétée ensuite par l'heuristique H1 ou H2. L'heuristique H3 peut être décrite comme suit :

1. Résoudre la relaxation du programme $P1$ avec $T = 1$.
2. Prendre les parties entières des valeurs trouvées.
3. Calculer les temps résiduels sur chacune des machines.
4. Appliquer l'heuristique H1 ou H2 pour trouver les autres affectations en utilisant les temps résiduels.

Exemple 1 *Considérons l'instance suivante avec $m = 3$ machines, $n = 10$ tâches ayant les temps d'exécution : $p_1 = 21$, $p_2 = 10$, $p_3 = 14$, $p_4 = 20$, $p_5 = 21$, $p_6 = 6$, $p_7 = 15$, $p_8 = 15$, $p_9 = 6$, $p_{10} = 12$. La longueur d'une période est fixée à 10 unités de temps, et le nombre d'affectations est 6 et sont comme suit : $AT_1 = (0.7, 0.7, 1)$, $AT_2 = (0.7, 1, 0.6)$, $AT_3 = (1, 0.6, 0.6)$, $AT_4 = (1, 1, 0)$, $AT_5 = (1, 0, 1)$ et $AT_6 = (0, 1, 1)$. Supposons que les trois premières tâches soient affectées à la première machine, les trois suivantes à la deuxième machine et le reste à la troisième machine. On obtient $P_1 = 45$, $P_2 = 47$ et $P_3 = 48$.*

L'heuristique H1 génère un makespan de 60 (solution optimale). L'affectation AT1 est utilisée aux périodes 1, 2 et 3, l'affectation AT2 est utilisée aux périodes 4 et 5, et l'affectation AT3 est utilisée à la dernière période.

L'heuristique H2 génère un makespan de 63. L'affectation AT1 est utilisée aux périodes 1, 2, 4, 5 et 6, l'affectation AT2 est utilisée à la période 3, et l'affectation AT4 est utilisée à la dernière période.

L'heuristique H3 génère un makespan de 60 (solution optimale). La relaxation du programme linéaire génère $Y1 = 30$, $Y2 = 20$ et $Y3 = 10$. Par conséquent, l'affectation AT_1 est utilisée pendant trois périodes, AT_2 pendant deux périodes et AT_3 pendant une période.

4. CAS OÙ LES AFFECTATIONS DES OPÉRATEURS SONT CONNUES

Dans le cas où les affectations des opérateurs aux machines sont connues, le problème revient à trouver seulement l'affectation optimale des tâches aux machines. Il est clair que ce problème est $\mathcal{NP}$-difficile. En effet, pour une seule affectation, le problème est équivalent au problème classique des machines parallèles uniformes. Nous allons donc évaluer l'heuristique, ECT-LPT, qui consiste à ordonner les tâches suivant l'ordre décroissant de leur temps d'exécution et d'ordonnancer chacune d'elles sur la machine qui les termine le plus tôt possible. Considérons les affectations de taux de service τ.

Théorème 3 *Si $C_{\max}(OPT)$ et $C_{\max}(ECT-LPT)$ désignent le makespan de la solution optimale et celui de l'heuristique ECT-LPT, respectivement, alors pour deux machines et un nombre n de tâches plus grand qu'un entier k, la relation suivante est vérifiée :*

$$\frac{C_{\max}(ECT-LPT)}{C_{\max}(OPT)} < 1 + \frac{1}{k}.$$

Preuve : Nous allons procéder par contradiction. Supposons que le résultat ne soit pas vérifié et considérons un contre exemple ayant le plus petit nombre de tâches. Il est connu que, pour ECT-LPT et ce contre exemple, la tâche ayant le plus petit temps d'exécution, p_n, est la dernière à commencer et terminer son exécution sur l'ensemble des machines. Avant d'exécuter cette tâche, soit a (respectivement b) la date de fin d'exécution de la dernière tâche sur la machine m_2 (respectivement m_1). Sans perte de généralité, nous pouvons supposer $a \leq b$. Soit x la productivité moyenne de la machine m_1 entre les temps a et b. La relation suivante est alors vérifiée :

$$\begin{aligned} C_{\max}(ECT-LPT) &\leq \min\Bigg(b + \frac{2p_n}{\tau}, a + \\ &\quad (b-a)(\tau - x) + \\ &\quad 2 \times \frac{p_n - (b-a)(\tau - x)}{\tau}\Bigg). \\ &\leq \frac{2p_n}{\tau} + \min\left(b, a + \right. \\ &\quad \left. (b-a)(\tau - x) - \right. \end{aligned}$$

$$
\begin{aligned}
& \frac{2(b-a)(\tau-x)}{\tau}\Big) \\
\leq\ & \frac{2p_n}{\tau} + \frac{(\tau-2)(b-a)(\tau-x)}{\tau} \\
& +a \\
\leq\ & \frac{p_n}{\tau} + a + \frac{x(b-a)}{\tau} + \frac{p_n}{\tau} + \\
& (b-a)(\tau-x) - 2(b-a) + \\
& \frac{x(b-a)}{\tau} \\
\leq\ & \frac{p_n + \tau a + x(b-a)}{\tau} + \frac{p_n}{\tau} + \\
& (b-a)\left(\tau - x - 2 + \frac{x}{\tau}\right) \\
\leq\ & \frac{p_n + \tau a + x(b-a)}{\tau} + \frac{p_n}{\tau}.
\end{aligned}
$$

Or,

$$\tau a + x(b-a) = \sum_{i=1}^{n-1} p_i.$$

Donc,

$$
\begin{aligned}
C_{\max}(ECT-LPT) &\leq \frac{p_n + \sum_{i=1}^{n-1} p_i}{\tau} + \frac{p_n}{\tau} \\
&= \frac{\sum_{i=1}^{n} p_i}{\tau} + \frac{p_n}{\tau}.
\end{aligned}
$$

Comme

$$C_{\max}(OPT) \geq \frac{\sum_{i=1}^{n} p_i}{\tau},$$

on aura

$$C_{\max}(ECT-LPT) \leq C_{\max}(OPT) + \frac{p_n}{\tau},$$

et

$$\frac{C_{\max}(ECT-LPT)}{C_{\max}(OPT)} \leq 1 + \frac{p_n}{\tau \times C_{\max}(OPT)}.$$

Comme on a supposé

$$1 + \frac{1}{k} \leq \frac{C_{\max}(ECT-LPT)}{C_{\max}(OPT)},$$

on obtient

$$1 + \frac{1}{k} \leq 1 + \frac{p_n}{\tau \times C_{\max}(OPT)}.$$

D'où,

$$
\begin{aligned}
C_{\max}(OPT) &\leq \frac{kp_n}{\tau} \\
&\leq \frac{np_n}{\tau}.
\end{aligned}
$$

Ce résultat implique que le nombre de tâches doit être égal à k et ces tâches doivent toutes avoir le même temps d'exécution. Or, dans ce cas, l'heuristique ECT-LPT est optimale; ce qui est une contradiction.

Corollaire 1

$$\frac{C_{\max}(ECT-LPT)}{C_{\max}(OPT)} < 1 + \frac{1}{n}.$$

Exemple 2 *Considérons l'instance suivante à deux affectations* $AT_1 = (0.8, 0.2)$ *et* $AT_2 = (0.5, 0.5)$*, trois périodes de 10 unités de temps.* AT_1 *est utilisée durant les périodes 1 et 3,* AT_2 *est utilisée durant la période 2, et* $n = 5$ *tâches ayant les temps d'exécution :* $p_1 = 7$*,* $p_2 = 4$*,* $p_3 = 4$*,* $p_4 = 4$ *et* $p_5 = 1$*. L'heuristique ECT-LPT génère un makespan de 22.5, tandis que le makespan optimal est égal à 20 (en ordonnançant la tâche 1 sur la machine* m_2 *et le reste des tâches sur la machine* m_1*). On obtient le ratio suivant :*

$$\frac{C_{\max}(ECT-LPT)}{C_{\max}(OPT)} = \frac{22.5}{20} = 1.125,$$

tandis que

$$1 + \frac{1}{n} = 1 + \frac{1}{5} = 1.333.$$

Théorème 4 *Si* $C_{\max}(OPT)$ *et* $C_{\max}(ECT-LPT)$ *désignent le makespan de la solution optimale et celui de l'heuristique ECT-LPT, respectivement, alors pour* m *machines et un nombre* n *de tâches plus grand qu'un entier* k*, la relation suivante est vérifiée :*

$$\frac{C_{\max}(ECT-LPT)}{C_{\max}(OPT)} < 1 + \frac{m-1}{k}.$$

Preuve : Nous procédons par contradiction, comme au Théorème 3. Supposons que le résultat ne soit pas vérifié et considérons un contre exemple ayant le plus petit nombre de tâches. Il est connu que, pour ECT-LPT et ce contre exemple, la tâche ayant le plus petit temps d'exécution, p_n, est la dernière à commencer et terminer son exécution sur l'ensemble des machines. Soit m_s la machine sur laquelle est ordonnancée la plus petite tâche. Si on ajoute une tâche dont le temps d'exécution est p_n sur la machine m_j, $j \neq s$, alors chacune des machines aura un temps de fin d'exécution supérieur ou égal à $C_{\max}(ECT-LPT)$. Par conséquent,

$$C_{\max}(ECT-LPT) \leq \frac{\sum_{i=1}^{n} p_i}{\tau} + \frac{(m-1)p_n}{\tau}.$$

Comme

$$C_{\max}(OPT) \geq \frac{\sum_{i=1}^{n} p_i}{\tau},$$

on obtient

$$C_{\max}(ECT-LPT) \leq C_{\max}(OPT) + \frac{(m-1)p_n}{\tau}.$$

Autrement dit,

$$\frac{C_{\max}(ECT-LPT)}{C_{\max}(OPT)} \leq 1+\frac{(m-1)p_n}{\tau \times C_{\max}(OPT)}.$$

Par ailleurs, nous avons supposé

$$1+\frac{m-1}{k} \leq \frac{C_{\max}(ECT-LPT)}{C_{\max}(OPT)}.$$

Par conséquent,

$$1+\frac{m-1}{k} \leq 1+\frac{(m-1)p_n}{\tau \times C_{\max}(OPT)}.$$

D'où,

$$\begin{aligned} C_{\max}(OPT) &\leq \frac{kp_n}{\tau} \\ &\leq \frac{np_n}{\tau}. \end{aligned}$$

Ce résultat implique que le nombre de tâches devrait être égal à k et ces tâches doivent toutes avoir le même temps d'exécution. Or, dans ce cas, l'heuristique ECT-LPT est optimale; ce qui est une contradiction.

Corollaire 2

$$\frac{C_{\max}(ECT-LPT)}{C_{\max}(OPT)} < 1+\frac{m-1}{n}.$$

Exemple 3 *Considérons l'instance suivante avec $m = 3$ machines, $n = 5$ tâches ayant les temps d'exécution suivants : $p_1 = 7$, $p_2 = 7$, $p_3 = 6$, $p_4 = 5$ et $p_5 = 5$, et trois affectations $AT_1 = (0.8, 0.2, 0.5)$, $AT_2 = (0.5, 0.5, 0.5)$ et $AT3 = (0.6, 0.6, 0.3)$ utilisées respectivement pendant les périodes 1, 2 et 3. La longueur de chaque période est de 10 unités de temps. Le makespan généré par ECT-LPT est alors $C_{\max}(ECT - LPT) = 26.66$, tandis que celui de la solution optimal est $C_{\max}(OPT) = 20$. Cela donne le ratio suivant :*

$$\frac{C_{\max}(ECT-LPT)}{C_{\max}(OPT)} = \frac{26.66}{20} = 1.33,$$

qui n'est pas très loin de la borne supérieure :

$$1+\frac{m-1}{n} = 1+\frac{2}{5} = 1.4.$$

5. CAS GÉNÉRAL

Lorsque la partition des tâches sur les machines n'est pas connue a priori, nous proposons de résoudre le problème en deux phases. Dans un premier temps, nous cherchons une partition équilibrée en utilisant l'heuristique ECT-LPT. Dans une deuxième phase, nous appliquons l'une des heuristiques H1, H2 ou H3 pour trouver les affectations à utiliser de façon à minimiser le makespan. Une fois les affectations trouvées, nous cherchons une partition en appliquant l'heuristique ECT-LPT. Cette procédure est répétée jusqu'à ce que le makespan ne puisse plus être amélioré.

Exemple 4 *Soit l'instance avec $m = 3$ machines, $n = 7$ tâches ayant les temps d'exécution suivants : $p_1 = 10$, $p_2 = 8$, $p_3 = 6$, $p_4 = 5$, $p_5 = 4$, $p_6 = 3$ et $p_7 = 1$, et 5 affectations $AT_1 = (1, 0.8, 0.5)$, $AT_2 = (0.8, 0.5, 1)$, $AT_3 = (1, 1, 0)$, $AT_4 = (1, 0, 1)$ et $AT_5 = (0, 1, 1)$. La durée de la période de référence est de 10 unités de temps.*

L'heuristique ECT-LPT genère la partition suivante : les tâches J_1 et J_6 sur la machine m_1, J_2 et J_5 sur m_2 et J_3, J_4 et J_7 sur m_3. Nous aurons donc $P1 = 13$, $P2 = 12$ et $P3 = 12$. En appliquant l'heuristique H1, on obtient un makespan de 18 et l'affectation AT_1 en première période et AT_2 en seconde période. Si l'heuristique ECT-LPT est appliquée avec ces affectations, on obtient un makespan de 16.25 et la partition suivante : J_1, J_5 et J_7 sur m_1, J_2 et J_6 sur m_2 et J_3 et J_4 sur m_3. Si l'heuristique H1 est appliquée une nouvelle fois avec cette dernière partition, la valeur du makespan et les affectations ne changent pas. Notons qu'on obtient le même résultat si c'est l'heuristique H2 au lieu de H1 qui est appliquée.

6. RÉSULTATS NUMÉRIQUES

Les différentes heuristiques ont été codées en Turbo Pascal et exécutées sur un PC muni d'un processeur INTEL T5200 cadencé à 1.6 GHz et d'une mémoire vive de 1Go. La relaxation du programme linéaire en nombres entiers $P1$ est résolue avec CPLEX 8.0. Les différents tests expérimentaux ont été réalisés sur des instances générées aléatoirement. Les principaux paramètres caractérisant une instance sont le nombre m de machines, le nombre n de tâches et le nombre *nop* d'opérateurs. Comme chaque opérateur peut s'occuper de plusieurs machines en parallèle, nous aurons comme paramètres supplémentaires les différentes façons d'affecter ce dernier et, pour chacune de ces affectations, les productivités sur chaque machine. Pour cette simulation, nous limitons le nombre de machines supervisées par un opérateur à 3. Lorsqu'un opérateur s'occupe de deux machines, nous considérons deux affectations sur les deux machines avec des productivités respectives de (0.8, 0.4) et (0.6, 0.6). Lorsqu'il supervise 3 machines, nous considérons seulement une affectation avec des productivités de 0.3, 0.4 et 0.5 sur les trois machines. Notons que

le nombre possible d'affectations pour les m machines explose lorsque m augmente; il est de l'ordre de $m!$. Nous limitons donc le nombre de machines à 6. La performance de nos heuristiques est évaluée en comparant les solutions obtenues à une borne inférieure. Dans le cas où l'affectation des tâches aux machines est connue, la borne inférieure est donnée par la résolution de la relaxation du programme linéaire $P1$ avec $T = 1$. Dans le cas où l'affectation des opérateurs aux machines et l'affectation des tâches aux machines ne sont pas connues, la borne inférieure est donnée par $\left(\sum_{i=1}^{n} p_i\right) / \tau_{\max}$.

Dans ce qui suit H31 (respectivement H32) désigne l'heuristique H3 lorsque l'heuristique H1 (respectivement H2) est utilisée pour trouver les affectations à utiliser avec les temps résiduels. Dans la première simulation (le cas où la partition des tâches est connue), nous avons généré des valeurs désignant la somme des temps d'exécution des tâches affectées à chaque machine. Nous avons considéré deux intervalles pour ces valeurs : [60, 100] et [80, 100]. Pour chaque intervalle, nous générons 5 instances que nous résolvons chacune en prenant comme valeur de la période de référence 5, 10, 15 et 20, et nous calculons la moyenne des valeurs du makespan trouvé.

Les résultats globaux, Figure 1, montrent qu'en moyenne, les heuristiques produisent des solutions à moins de 5% de la borne inférieure, ce qui est déjà un résultat positif. Nous constatons aussi une similitude de comportement entre les heuristiques H1 et H2 et entre les heuristiques H31 et H32 (Figure 1). Le second groupe se détache nettement du premier. De même, cette figure montre également que plus la taille de la période de référence est petite, meilleurs sont les résultats pour toutes les heuristiques et dans tous les cas. Ceci peut assez bien s'expliquer comme suit : H1 et H2 sont des optimisations successives sur toutes les périodes. Il semble que plus le découpage est fin, meilleur est le résultat. Pour H31 et H32, comme il s'agit de relaxation linéaire, plus les valeurs réelles sont grandes, meilleure est l'approximation.

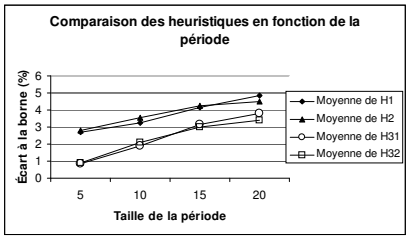

Figure 1:

On peut reprendre les résultats en fonction de l'équilibre de la partition initiale. Pour toutes les tailles de période et pour l'ensemble des heuristiques, les performances avec une partition plus équilibrée (valeurs entre 80 et 100, Figure 2) sont meilleures qu'avec une partition moins équilibrée (valeurs entre 60 et 100, Figure 3).

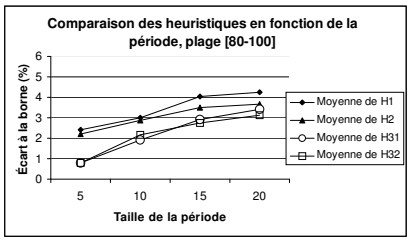

Figure 2:

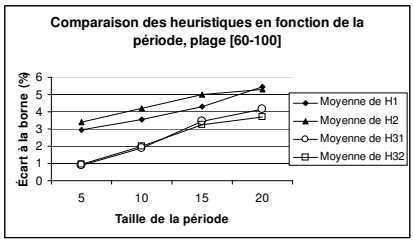

Figure 3:

Pour la deuxième simulation, représentant le cas général, nous avons généré d'une manière aléatoire les temps d'éxécution des tâches dans deux intervalles : [1, 20] et [1, 50]. Pour le nombre de tâches, nous avons pris les valeurs 20, 50 et 100. Comme pour la première simulation, nous générons 5 instances pour chaque intervalle des temps d'exécution et chaque nombre de tâches. Nous avons comparé les écarts à la borne inférieure en fonction de la période pour toutes les heuristiques, Figure 4. Nous constatons le même phénomène d'appariement entre les heuristiques H1, H2 et H31, H32. Nous constatons surtout que toutes les heuristiques produisent en moyenne des résultats à moins de 1.6% de la borne inférieure. Les heuristiques sont donc efficaces. De manière plus surprenante, les heuristiques H31, H32 semblent dominer les heuristiques H1 et H2 pour une petite taille de période, alors que le phénomène semble s'inverser pour les grandes périodes.

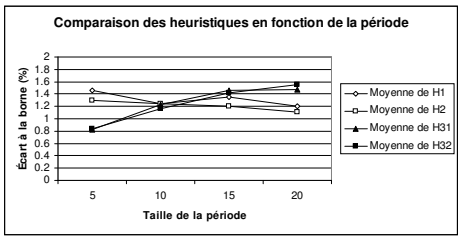

Figure 4:

Pour analyser ce phénomène en détails, nous pouvons observer les résultats en fonction de la variabilité des temps d'exécution. Nous avons séparé les résultats obtenus avec des temps d'exécution distribués entre 1 et 20 (Figure 5) et les résultats obtenus avec des temps d'exécution entre 1 et 50 (Figure 6). Dans les deux cas, le même phénomène d'appariement est constaté pour les heuristiques. Le comportement des heuristiques est également similaire pour les deux cas. Dans le second cas (tâches entre 1 et 50), les heuristiques ont la même performance pour la période de 15 unités de temps alors que l'heuristique H2 semble dominer légèrement les autres pour de grandes périodes. Pour toutes les tailles de période et pour toutes les heuristiques, les performances avec des tâches plus diversifiées (entre 1 et 50 unités de temps, Figure 6) sont meilleures qu'avec des tâches moins diversifiées (entre 1 et 20 unités de temps, Figure 5).

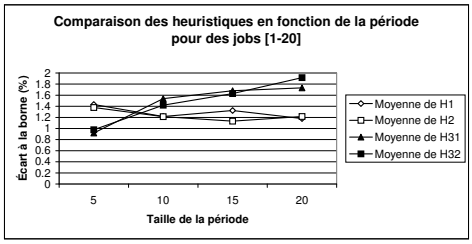

Figure 5:

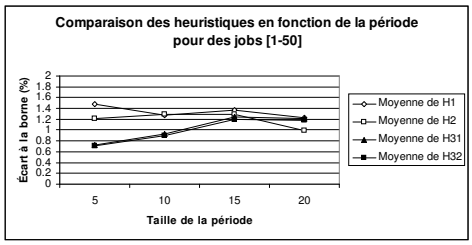

Figure 6:

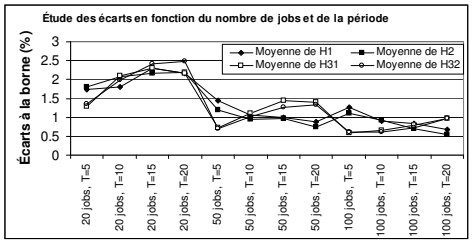

Figure 7:

Dans le cas général, nous avons également comparé les écarts à la borne inférieure en fonction du nombre de tâches et de la période pour toutes les heuristiques (Figure 7). Les résultats montrent le même phénomène d'appariement pour les heuristiques. Les heuristiques H31 et H32 sont plus performantes que H1 et H2 pour des petites périodes et le phénomène s'inverse pour les grandes périodes pour tous les nombres considérés de tâches. De même, plus le nombre de tâches augmente, meilleure est la performance de ces heuristiques. À partir de 50 tâches, les résultats donnés par les heuristiques sont en moyenne à moins de 1.5% de la borne inférieure.

7. CONCLUSION

Cet article étudie une modélisation particulière de l'influence, sur l'ordonnancement, de la détérioration des durées d'exécution due au fait qu'il y a moins d'opérateurs que de postes de travail. Ce modèle lui-même peut être questionné. Nous avons fait l'hypothèse que la détérioration est indépendante des tâches effectuées. Ce n'est sans doute pas toujours le cas. Nous avons également fait l'hypothèse que l'affectation reste constante sur la période, même si une des machines a terminé son travail. Cela peut ne pas encore être le cas. Si un opérateur peut s'occuper de plusieurs manières différentes du même sous-ensemble de machines (il existe donc plusieurs vecteurs de productivité sur le même sous-ensemble de machines), nous avons supposé que l'opérateur ne pouvait pas en changer à l'intérieur de la période. Ceci est sans doute restrictif. Cependant, devant un problème complexe aussi ouvert, il fallait poser quelques hypothèses pour commencer à avoir des résultats.

Sous ces hypothèses, nous avons considéré trois situations : le cas où l'affectation des tâches aux machines est connue, le cas où les affectations des opérateurs sont connues et le cas où l'affectation des tâches aux machines et celle des opérateurs ne sont pas connues a priori. Dans le premier cas, nous avons présenté trois heuristiques et une étude expérimentale pour évaluer leur performance dans le cas moyen. Des résultats obtenus, nous concluons principalement que les heuristiques présentées sont efficaces et que les résultats générés sont meilleurs en considérant des partitions équilibrées. Dans le deuxième cas, nous avons présenté une heuristique pour laquelle une analyse dans le pire cas est entreprise. Ainsi, nous avons obtenu un ratio de performance égal à $(1+\frac{1}{n})$. Enfin, dans le troisième cas, nous avons présenté une autre heuristique utilisant l'une des heuristiques présentées dans le premier cas. Une évaluation de sa performance dans le cas moyen a été également entreprise. Nous avons relevé sa très bonne performance quand le nombre de tâches devient de plus en plus grand.

Les perspectives de cette recherche porteront sur la remise en question des hypothèses restrictives, l'extension aux autres problèmes d'ordonnancement, et finalement au problème de changement d'affectations en fin de tâche.

Remerciements

Cette recherche a été financée en partie par le Conseil de Recherche en Sciences Naturelles et en Génie du Canada.

RÉFÉRENCES

Bourland, K.E., and L.K. Carl, 1994. Parallel-machine scheduling with fractional operator requirements. *IIE Transactions (Institute of Industrial Engineers)*, 26(5), p. 56-65.

Cheurfa, M. 2005. *Gestion des ressources humaines en production cyclique*. Thèse de doctorat, École des Mines de Saint-Étienne, France.

Daniels, R.L., B.J. Hoopes, and J.B. Mazzola, 1996. Scheduling parallel manufacturing cells with resource flexibility. *Management Science*, 42(9), p. 1260-1276.

Grigoriev, A., M., Sviridenko, and M. Uetz, 2005. Unrelated parallel machine scheduling with resource dependent processing times. *Lecture Notes in Computer Science*, 3509, p. 182-195.

Hu, P.C. 2005. Minimizing total flow time for the worker assignment scheduling problem in the identical parallel-machine models. *International Journal of Advanced Manufacturing Technology*, 25(9-10), p. 1046-1052.

Hu, P.C. 2004. Minimising total tardiness for the worker assignment scheduling problem in identical parallel-machine models. *International Journal of Advanced Manufacturing Technology*, 23(5-6), p. 383-388.

Huq, F., K. Cutright, and C. Martin, 2004. Employee scheduling and makespan minimization in a flow shop with multi-processor work stations: a case study. *Omega*, 32(2), p. 121-129.

Karlof, J.K., and W. Wang, 1996. Bilevel programming applied to the flow shop scheduling problem. *Computers and Operations Research*, 23(5), p. 443-451.

Nowicki, E., and S. Zdrzalka, 1990. A survey of results for sequencing problems with controllable processing times. *Discrete Applied Mathematics*, 26(2-3), p. 271-287.

Pinedo, M. 2002. *Scheduling: Theory, Algorithms, and Systems*, 2nd edition, Prentice Hall.

Vickson, R.G. 1980a. Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine. *Operations Research*, 28(5), p. 1155-1167.

Vickson, R.G. 1980b. Two single machine sequencing problems involving controllable job processing times. *AIIE transactions*, 12(3), p. 258-262.

Zouba, M, P. Baptiste, and D. Rebaine, 2006. Ordonnancement de jobs sur deux machines parallèles en présence d'un seul opérateur. 6^e *Conférence Francophone de Modélisation et simulation (MOSIM'06)*, Rabat, Maroc, 07 pages.