RAIRO. RECHERCHE OPÉRATIONNELLE

A. VIGNIER
J.-C. BILLAUT
C. PROUST

Les problèmes d'ordonnancement de type « flow-shop » hybride : état de l'art

RAIRO. Recherche opérationnelle, tome 33, n° 2 (1999), p. 117-183

<http://www.numdam.org/item?id=RO_1999__33_2_117_0>

RAIRO Rech. Opér.
(vol. 33, n° 2, 1999, pp. 117-183)

LES PROBLÈMES D'ORDONNANCEMENT DE TYPE FLOW-SHOP HYBRIDE : ÉTAT DE L'ART (*)

par A. VIGNIER ([1]), J.-C. BILLAUT ([1]) et C. PROUST ([1])

Communiqué par Jacques CARLIER

Résumé. – *Nous nous proposons d'étudier ici une famille de problèmes d'ordonnancement, caractérisés par un ensemble de travaux comportant un enchaînement identique d'opérations sur différents étages d'un outil de production. Chaque étage, ou pool, est constitué d'un ensemble de machines, susceptibles de fabriquer les produits avec plus ou moins d'aptitude. Ce problème se nomme habituellement « flow-shop » hybride. Après une introduction aux problèmes d'ordonnancement en général, nous définissons plus précisément notre préoccupation, nous proposons une notation permettant d'identifier clairement et rapidement un problème donné, puis nous abordons un état de l'art comportant deux parties : les problèmes à deux étages, puis ceux à k étages. Un tableau de synthèse et une typologie mettent en évidence le peu de problèmes déjà traités.*

Mots clés : Ordonnancement, Flow-shop hybride, état de l'art.

Abstract. – *A special class of scheduling problems is studied in this paper, named Hybrid Flowshop. n jobs have to be performed in a shop and each of them has the same routing (so this is a flowshop). A job consists in k different operations. A set of machines are able to perform each operation and this set is called a stage. So when a job consists in two operations, there are two stages in the shop. After introducing the scheduling generalities, we define our preocupations and we propose a notation in order to identify precisely and rapidly a problem. Then a state of the art is proposed and presented in two parts. The first one is dedicated to the 2-stage hybrid flowshops and the second to the general case of the k-stage. Then a summary puts to the fore that many problems remain unsolved.*

Keywords: Scheduling, Hybrid Flowshop, state of the art.

1. INTRODUCTION AUX PROBLÈMES D'ORDONNANCEMENT

On dit qu'on a affaire à un problème d'ordonnancement lorsque : « on doit programmer l'exécution d'une réalisation en attribuant des ressources

(*) Reçu en septembre 1995.

([1]) École d'Ingénieurs en Informatique pour l'Industrie, Université Laboratoire d'Informatique, 64, av. J. Portalis, 37200 Tours, France. e-mail : vignier@univ-tours.fr

Recherche opérationnelle/Operations Research, 0399-0559/99/02/$ 7.00

aux tâches et en fixant leurs dates d'exécution [...]. Les tâches sont le dénominateur commun des problèmes d'ordonnancement, leur définition n'est ni toujours immédiate, ni toujours triviale » [Ca-1988].

Une expertise en ordonnancement nécessite une prise de connaissance d'une problématique particulière pour en faire émerger les contraintes dures, fondamentales. Ceci suppose la connaissance d'une typologie telle que celle présentée dans [go-1993]. Associée à cette typologie, il est nécessaire d'utiliser une notation homogène permettant d'identifier clairement et rapidement un problème donné. Nous utilisons et étendons la notation proposée dans [ri-1976] (*voir* 2.3).

On peut modéliser analytiquement ces problèmes (fonction objectif, ensemble des contraintes) selon une démarche classique de Recherche Opérationnelle afin d'utiliser des outils standards de résolution (*voir* par exemple [st-1990]). Mais d'une part ceci n'autorise pas forcément l'exploitation des propriétés propres au problème étudié pour affronter la complexité de sa résolution (une approche de type Procédure par Séparation et Evaluation spécifique le permet) et d'autre part, le modèle initial ne peut être facilement étendu pour prendre en compte de nouvelles contraintes. On passe très vite d'un besoin d'une résolution de Problème Linéaire Simple (à variables réelles) à un Problème Linéaire Entier ou à un Problème Linéaire Mixte lorsqu'on est heureux d'échapper au Non Linéaire! Cette apparente efficacité est alors synonyme de manque de généricité même si l'on constate actuellement quelques tentatives pour y remédier [fo-1993].

Quoique ces outils ne soient pas à négliger pour la résolution de cas simples – ils permettent aussi très souvent d'obtenir de bonnes bornes inférieures en relaxant certaines contraintes du problème originel – leur emploi, en tant que seul « solveur » lors de la résolution de cas industriels est inenvisageable. Les problèmes d'ordonnancement sont généralement NP-difficiles [la-1989], sauf dans des cas très particuliers (comme par exemple pour la minimisation du temps total d'exécution dans un flow-shop à 2 machines, résolu en $O(n \log n)$ par le célèbre algorithme J de S. M. Johnson [jo-1954] ou ses extensions [pr-1992]). Toutefois ces algorithmes classiques de la Recherche Opérationnelle peuvent aussi être associés – en tant que composants – dans des logiciels plus vastes en vue d'une démarche d'aide à la décision [pr-1995]. En outre, en vue d'une réduction de la taille du problème originel, on peut analyser puis exploiter ses spécificités : présence de machines de type « goulet d'étranglement » [la-1978], [mo-1983], travaux prépondérants [na-1983], décompositions spatio-temporelles [po-1988] ...

D'autres approches de résolution doivent donc être envisagées, par un expert du domaine, telles que le recuit simulé, les méthodes Tabou, les algorithmes génétiques, les réseaux neuromimétiques [al-1997], [co-1994] ou encore l'association de plusieurs de ces techniques comme, par exemple, les Réseaux de Petri et la Programmation Logique avec propagation de Contraintes [ri-1995]. En outre, certains auteurs se contentent de mettre en évidence un ensemble de solutions admissibles en dehors de tout souci d'optimisation ([ba-1987], [ch-1990], [bi-1996], [le-1994], ...), le(s) critère(s) retenu(s) étant sujet(s) à caution.

Le paragraphe 2 présente les flow-shops hybrides et une notation appropriée. Le paragraphe 3 est consacré à l'état de l'art : une première partie est dédiée aux problèmes à deux étages et une seconde aux problèmes à k étages. Le paragraphe 4 présente une typologie des problèmes de type flow-shop hybride ainsi que des tableaux de synthèse.

2. LA PROBLÉMATIQUE DU FLOW-SHOP HYBRIDE

Le rôle essentiel de la gestion de production est de gérer un ensemble de ressources et de charges de travail, selon un ou plusieurs critères ou contraintes, plus ou moins explicites, à satisfaire et/ou à « optimiser » [le-1992]. Deux notions sont sous-jacentes : celle de l'organisation du système de production et celle des problématiques d'ordonnancement associées. Nous rappelons une typologie des systèmes de production qui justifie l'étude des problèmes d'ordonnancement de type flow-shop hybride et plus précisément ceux à deux étages.

2.1. Une réalité industrielle

Une typologie des systèmes de production a été proposée par Woodward et reprise en particulier par [le-1979], étendue et raffinée depuis. Cette classification simple, voire simpliste, offre l'avantage de fournir instantanément une image claire des entreprises lorsqu'on focalise l'analyse sur l'organisation de l'outil de production [le-1992].

En réalité, il est très rare que l'organisation d'une entreprise puisse nous permettre de la classer uniquement dans une des cases de cette typologie. On découvre beaucoup plus souvent des organisations mixtes, soit en parallèle, soit en série de type MASSE-ATELIER (*cf.* Fig. 1).

Dans cet exemple, l'entreprise choisit d'anticiper la production de produits semi-finis et d'en constituer un stock afin de raccourcir le délai entre

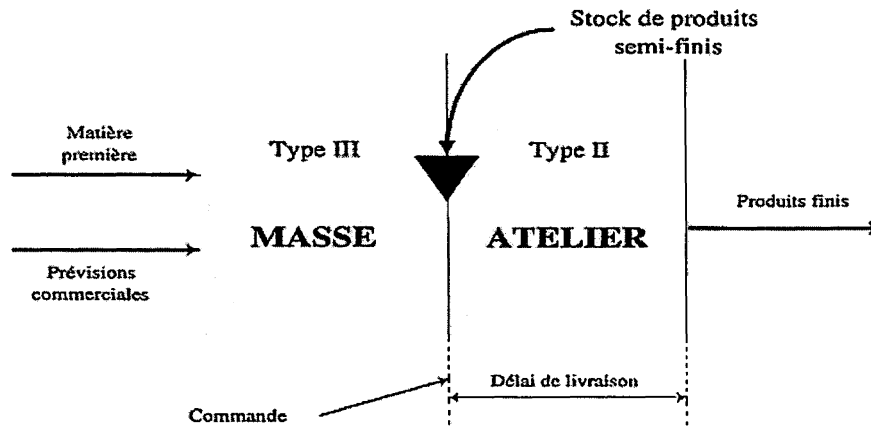

Figure 1. – Système de production organisé en série [le-1979].

la commande et la livraison. C'est très souvent le cas dans l'industrie électronique, du verre, du bois, du papier, des moquettes, de la mécanique de sous-traitance... Cette approche n'autorise pas pour autant à ignorer le cycle de vie du produit fabriqué [la-1982] et donc l'interaction des cycles « produit et processus » [ha-1979].

2.2. Définition du flow-shop hybride

Nous concentrons désormais nos réflexions sur une organisation du processus de production en série « MASSE-ATELIER » telle qu'elle est présentée figure 1. Les produits fabriqués passent dans un premier temps dans la cellule « MASSE » puis dans la cellule « ATELIER ». Dans chacune des deux cellules, on considère que les ressources sont regroupées en pools. Un pool contient un ensemble de ressources susceptibles d'exécuter une même opération, équivalentes dans leur fonctionnement mais pas dans leurs performances. C'est pourquoi la durée d'une opération peut dépendre des ressources choisies dans le pool et du nombre de ressources affectées à l'opération (si la ressource est une machine, on peut parler alors de machines parallèles identiques, proportionnelles ou différentes). Dans un premier temps, on supposera qu'il n'existe qu'un pool de ressources par étage (donc chaque produit ne subit que deux opérations : une par étage ou cellule). C'est le cas par exemple de la fabrication des bouteilles en verre [pr-1995], [pa-1979]. On se place ainsi dans la catégorie des problèmes d'ordonnancement connue sous le nom de flow-shop hybride à deux étages.

On espère que la disposition d'algorithmes efficaces pour la résolution de ce type de problème permettra ultérieurement une mise au point relativement aisée de bonnes heuristiques [va-1994] pour la résolution des problèmes à k étages comme ce fut le cas dans le cadre des flow-shops de base [pr-1992].

Dans un flow-shop hybride, les gammes sont toutes identiques – comme dans tout flow-shop – et les machines sont regroupées par étages. Il apparaît donc que le flow-shop hybride est un cas particulier du job-shop avec machines dupliquées. De plus, si le nombre d'étages est égal à 1, alors on est en présence soit d'un problème à machines parallèles, soit d'un problème à une machine. En revanche s'il n'y a qu'une seule machine à chaque étage, alors on se ramène à un problème traditionnel de flow-shop. On peut donc faire apparaître la hiérarchie de la figure 2.

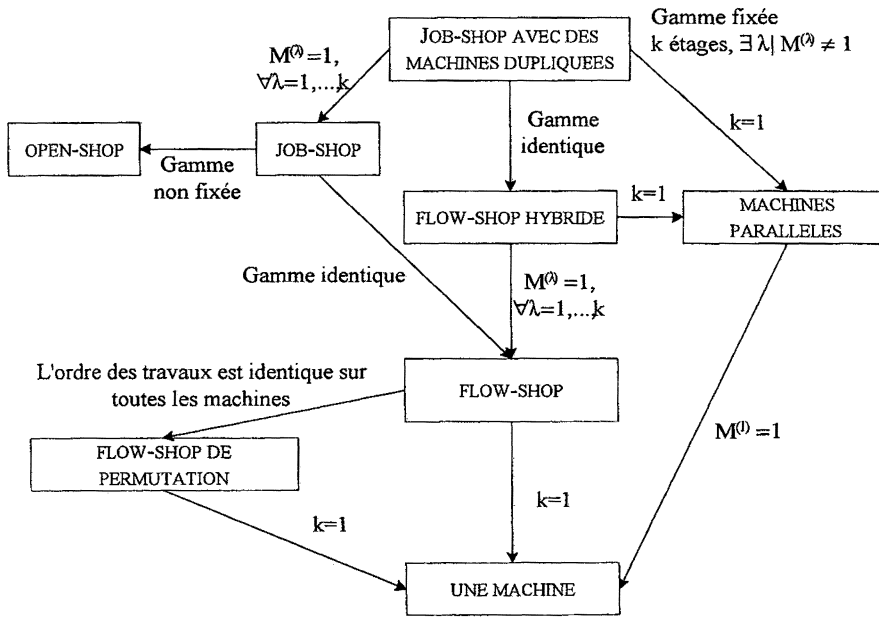

Figure 2. – Les problèmes d'ordonnancement d'atelier (d'après [ma-1993]).

Dans un problème de flow-shop on ne considère que le problème de séquencement des tâches. Le problème d'affectation n'existe pas puisqu'il n'y a qu'une seule ressource par étage. Ce n'est pas le cas dans le flow-shop hybride où on peut trouver à chaque niveau des machines parallèles. De plus il n'y a aucune raison pour que les tailles des lots, donc les temps opératoires, soient systématiquement des données du problème (surtout

lorsqu'on considère que les temps de réglage sont négligeables et qu'on n'en tient pas compte!). On distingue donc les problèmes suivants :

1. le problème de dimensionnement de lots,
2. le problème d'affectation des sous-lots sur les machines de chaque étage,
3. le problème d'ordonnancement des sous-lots sur les différentes machines.

2.3. Proposition d'une notation pour la prise en compte du flow-shop hybride à k étages

La notation la plus couramment utilisée est celle proposée par [ri-1976] (*voir* également [gr-1979] et [bl-1994]). Cependant, pour un certain nombre de problèmes d'ordonnancement – et pour le flow-shop hybride en particulier – elle n'est pas adaptée. De plus, dans la littérature spécialisée traitant de ces problèmes, il n'existe pas non plus de notation rigoureuse. Chaque auteur en utilise une, parfois bien particulière. Pour des raisons de clarté, de compréhension et d'efficacité d'intervention, il est important d'adopter une notation qui soit basée sur l'existant, homogénéisant les notations actuelles, et qui permette d'établir rapidement une typologie des problèmes déjà étudiés. Nous nous proposons donc d'enrichir la notation habituelle, basée sur la concaténation de trois champs $\alpha \mid \beta \mid \gamma$. Une étude similaire devrait être menée pour proposer une notation des problèmes de job-shop généralisé.

• Le champ α se compose de quatre paramètres que l'on peut représenter de la manière suivante $\alpha = \alpha_1\alpha_2, ((\alpha_3\alpha_4^{(\ell)})_{\ell=1}^{\alpha_2})$. Le premier, α_1, permet de savoir s'il s'agit d'un flow-shop hybride (noté FH) ou autre ($\emptyset$, O, J, F). Le deuxième, α_2, indique le nombre d'étages lorsqu'il est supérieur à 1. α_3 et α_4 sont répétés par couple, autant de fois qu'il y a d'étages. α_3 est égal à $\emptyset$ s'il n'y a qu'une machine, à P s'il s'agit d'un problème à machines parallèles identiques, à Q si c'est un problème à machines parallèles proportionnelles, à R s'il s'agit d'un problème à machines parallèles quelconques. Le nombre de machines de l'étage ℓ, $M^{(\ell)}$ en général, est indiqué dans α_4 et $M^{(\ell)}(t)$ indique que ce nombre est variable dans le temps ; il peut être représenté aussi par une fonction du temps ; c'est la notion de profil variable qui permet de tenir compte d'indisponibilités éventuelles des ressources [sa-1992]. α_4 peut être égal à $\emptyset$...

Ainsi pour un flow-shop hybride, $\alpha = \alpha_1\alpha_2, ((\alpha_3\alpha_4^{(\ell)})_{\ell=1}^{\alpha_2})$ où $\alpha_1 = FH$, $\alpha_2 = k$, $\alpha_3 \in (\emptyset, P, Q, R)$ et $\alpha_4 \in (M^{(\ell)}, M^{(\ell)}(t))$. On pourra utiliser la notation $(\alpha_3\alpha_4)^x$ pour indiquer sur les x étages consécutifs une configuration identique de α_4 machines parallèles de type α_3.

Exemple : Considérons un flow-shop hybride à 5 étages composé de 3 machines identiques aux deux premiers étages et de 2 machines identiques pour les autres. La notation du champ α sera donc $FH5$, $(P3^{(1)}, P3^{(2)}, P2^{(3)}, P2^{(4)}, P2^{(5)})$ ou $FH5$, $((P3^{(\ell)})_{\ell=1}^{2}, (P2^{(\ell)})_{\ell=3}^{5})$ ou bien encore $FH5$, $((P3)^2, (P2)^3)$. Si le détail n'est pas significatif, on pourra se contenter de $FH5$, $((PM^{(\ell)})_{\ell=1}^{k})$.

• Le champ β permet de définir l'ensemble des contraintes prises en compte (aussi bien celles concernant les travaux que celles concernant le procédé de fabrication). Une forme générique de ce champ peut être notée $\beta = \beta_1\beta_2...\beta_n$. Les β_i peuvent prendre de nombreuses valeurs en fonction de la particularité du problème étudié par rapport au problème de base (ce dernier a de nombreuses spécificités implicites qu'il convient de connaître). Les caractéristiques d'un travail sont notées habituellement en ordonnancement : r_i, d_i, $\tilde{d}_i$, q_i, w_i, C_i, F_i, L_i, T_i, U_i, $pmtn$, $split$, $a_i^{(k)(k+1)}$, $b^{(k)(k+1)}$, $prec$...(ces différents termes sont définis au fur et à mesure de leur utilisation dans cet article).

Nous allons maintenant préciser les notations qui nous semblent nécessaires pour formaliser l'ensemble des problèmes de type flow-shop hybride. On utilise un indiçage supérieur pour noter le numéro de l'étage. Les indices inférieurs i et j correspondent respectivement au numéro du travail et au numéro de la machine. On note $M^{(\ell)}$ le nombre de machines à l'étage ℓ. La machine j de l'étage ℓ est notée $M_j^{(\ell)}$. On considère qu'il y a n travaux à ordonnancer.

La nature même du problème et l'existence de machines parallèles, nous obligent à considérer la notion de « splitting » c'est-à-dire autoriser qu'un travail donné puisse être découpé en plusieurs parties. Ces parties peuvent être réalisées soit par plusieurs machines simultanément, soit par une ou plusieurs machines de sorte qu'à tout instant, deux machines n'exécutent pas deux parties du même travail. Le travail i consiste à réaliser Q_i produits. Cette quantité est réalisée à chaque étage ℓ et est découpée en un certain nombre de sous-lots $q_{ij}^{(\ell)}$ (quantité du travail i affectée à la machine j de l'étage ℓ). Il peut y avoir plusieurs $q_{ij}^{(\ell)}$ nuls et si on souhaite réaliser un travail sur une machine unique (pour un étage donné) alors un seul $q_{ij}^{(\ell)}$ sera non nul. On définit la quantité $v_{ij}^{(\ell)}$ qui représente le temps nécessaire pour réaliser une unité du travail i sur la machine j de l'étage ℓ. Soient $p_i^{(\ell)}$ le temps total nécessaire pour réaliser les Q_i produits du travail i à l'étage ℓ, $p_{ij}^{(\ell)}$ le temps nécessaire pour réaliser $q_{ij}^{(\ell)}$ produits du travail i sur la machine

j de l'étage ℓ, alors on peut établir les relations suivantes :

$$\sum_{j=1}^{M^{(\ell)}} q_{ij}^{(\ell)} = Q_i \ \forall i = 1, ..., n, \text{ et } \forall \ell = 1, ..., k$$

$$p_{ij}^{(\ell)} = q_{ij}^{(\ell)} \times v_{ij}^{(\ell)} \ \forall i = 1, ..., n, \ \forall \ell = 1, ..., k \text{ et } \forall j = 1, ..., M^{(\ell)}$$

$$p_i^{(\ell)} = \sum_{j=1}^{M^{(\ell)}} q_{ij}^{(\ell)} \times v_{ij}^{(\ell)} = \sum_{j=1}^{M^{(\ell)}} p_{ij}^{(\ell)} \ \forall i = 1, ..., n, \text{ et } \forall \ell = 1, ..., k.$$

Les caractéristiques β_i seront indicées supérieurement par le numéro des étages concernés. Si la propriété est valable pour tous les étages, on ne précise aucun indice. Par exemple, on peut imaginer qu'à un étage ℓ, et seulement sur celui-ci, des contraintes de précédence existent : on notera cette contrainte $prec^{(\ell)}$.

Des exemples réels de production nous obligent à introduire la notion de dépendance technologique inter-étages. Elle permet de préciser quel sera l'ensemble des machines que le travail pourra utiliser à l'étage $(\ell + 1)$, en fonction de ses affectations possibles à l'étage ℓ. Par exemple dans un problème à trois étages et avec 2 machines par étages, on peut considérer que si le travail i passe sur la machine 2 de l'étage 1 alors il ne pourra passer que sur la machine 1 de l'étage 2 et que sur la machine 2 de l'étage 3. Nous proposons de noter ces dépendances technologiques par des matrices $(DT^{(\ell,\ell+1)})$ qui permettent de croiser les machines du niveau ℓ à celles du niveau $(\ell + 1)$. Les éléments de ces matrices sont 0 ou 1. Ce sera le cas pour des machines couplées à des fours, des machines réparties dans des sites différents et éloignés... Évidemment, dans le même ordre d'idées, un travail, de par ses spécificités, peut n'être réalisé que sur certaines machines d'un étage donné.

• Le champ γ correspond classiquement au critère à optimiser [bl-1994] : C_{max}, F_{max}, L_{max}, T_{max}...

2.4. Propriétés des ordonnancements : rappels

Les ordonnancements peuvent être classés au moins en trois catégories : les ordonnancements « semi-actifs », les ordonnancements « actifs » et enfin les « sans délai » [ba-1974]. Ces notions fondamentales permettent de caractériser les solutions obtenues, indépendamment de la méthode de résolution utilisée [be-1982].

DÉFINITION 2.4.1 [Ordonnancement semi-actif] : *Un ordonnancement est dit semi-actif si aucun travail i ne peut être avancé sur la ressource où il se trouve compte tenu des contraintes de gamme et de précédence. Cela correspond au chemin le plus long dans un graphe potentiels-tâches.*

Une illustration d'un ordonnancement semi-actif mais ni actif, ni sans délai est donnée figure 3.

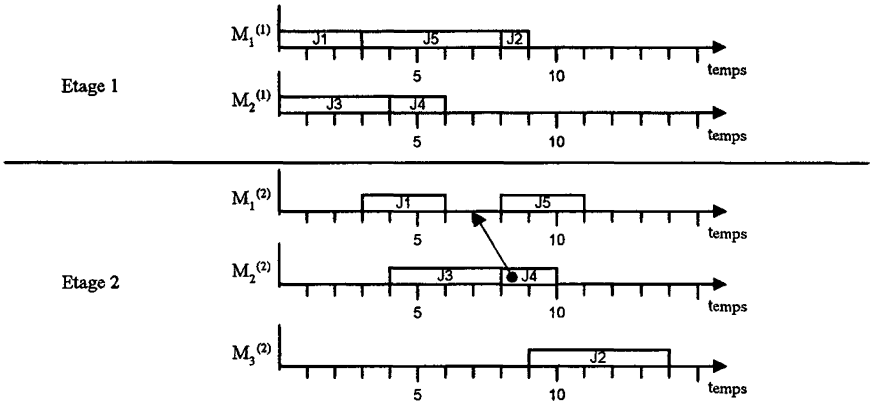

Figure 3. – Exemple d'un ordonnancement semi-actif.

DÉFINITION 2.4.2 [Ordonnancement actif] : *Un ordonnancement est dit actif si aucune opération d'un travail i ne peut débuter son exécution plus tôt sans déplacer au minimum une autre opération.*

Une illustration d'un ordonnancement actif mais pas sans délai est donnée figure 4.

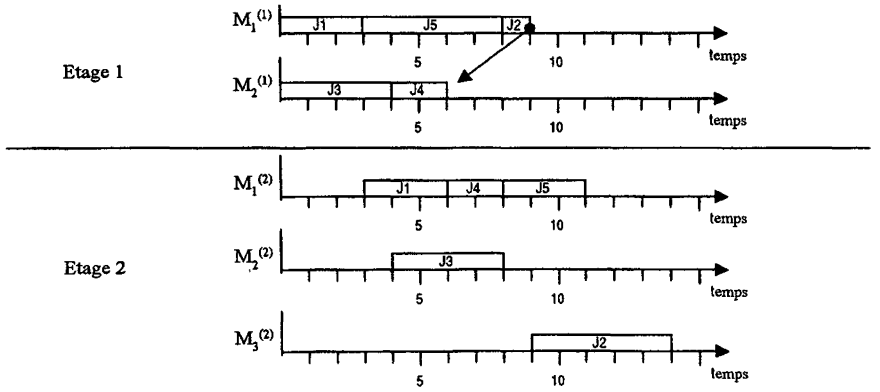

Figure 4. – Exemple d'un ordonnancement actif.

DÉFINITION 2.4.3 [Ordonnancement sans délai] : *Un ordonnancement est sans délai quand on ne laisse pas une machine inoccupée alors qu'une file d'attente contient au moins un travail susceptible de débuter son exécution sur cette machine. Une illustration d'un ordonnancement sans délai est donnée figure 5.*

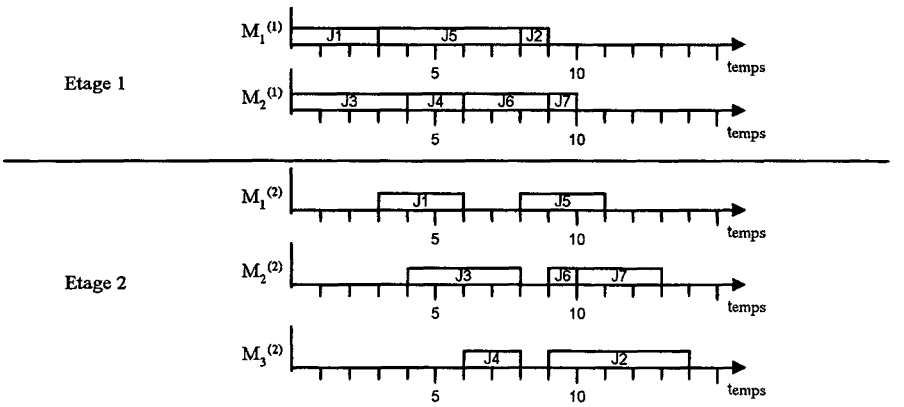

Figure 5. – Exemple d'ordonnancement sans délai.

3. ÉTAT DE L'ART [vi-1995]

Notons que les termes « Flow Shop with Multiple Processors » (FSMP) désignent également le flow-shop hybride ([br-1991], [va-1994], ...). Si des produits peuvent sauter des étages le problème se nomme aussi « flexible flow line » [wi-1985], [wi-1988]. Toutes les études évoquées ci-après, sauf spécification contraire, considèrent les tailles des lots comme données du problème (le temps d'exécution du travail est connu) et le découpage en sous-lots n'est pas autorisé. Nous présentons ci-dessous les principales idées et on renvoie à [vi-1995] pour les algorithmes détaillés et des exemples. C^*_{max} représente le temps d'achèvement total optimal et $C^{(M)}_{max}$ le temps d'achèvement total obtenu par la méthode M.

3.1. Les problèmes à deux étages

• Un des premiers articles traitant des flow-shops hybrides à deux étages date des années 1970. En effet Shen et Chen [sh-1972] proposent une stratégie d'ordonnancement pour le $FH2, ((PM^{(\ell)})^2_{\ell=1})\ ||\ C_{max}$. Cette stratégie appelée « ME strategy » (More Earlier time), est basée sur la détermination d'un ordre partiel sur l'ensemble des travaux. Ainsi, un travail

i précédera un travail j si les relations (3.1.1) et (3.1.2) sont vérifiées.

$$p_i^{(1)} + p_i^{(2)} \geq p_j^{(1)} + p_j^{(2)} \tag{3.1.1}$$

$$p_i^{(1)} \leq p_j^{(1)}. \tag{3.1.2}$$

Dans le cas particulier où cet ordre partiel est un ordre total sur l'ensemble des travaux, cette stratégie permet de libérer les processeurs du deuxième étage au plus tôt. Bien entendu, cette stratégie n'est pas optimale, mais les auteurs montrent que dans le cas particulier considéré ci-dessus, alors on a la relation $\frac{C_{max}^{(ME)}}{C_{max}^*} \leq \frac{3}{2} - \frac{1}{2 \times \max\{M^{(1)}, M^{(2)}\}}$. Cette stratégie permet de déterminer un ordre de passage des travaux mais pas de règle d'affectation. Aucune stratégie d'affectation n'est fournie. On peut supposer que la liste de priorité définie par cet ordre est employée dans un algorithme de liste classique, plaçant les travaux au plus tôt. Les solutions obtenues semblent être sans délai mais cela n'est pas précisé dans l'article.

• Buten et Shen [bu-1973] étudient le $FH2, ((PM^{(\ell)})_{\ell=1}^2) \mid\mid C_{max}$. L'algorithme utilise une relation de précédence déduite de celle de Johnson, appelée MJO (Modified Johnson Order). Le travail i précède le travail j si la relation (3.1.3) est vérifiée.

$$\min\left(\frac{p_i^{(1)}}{M^{(1)}}; \frac{p_j^{(2)}}{M^{(2)}}\right) < \min\left(\frac{p_j^{(1)}}{M^{(1)}}, \frac{p_i^{(2)}}{M^{(2)}}\right). \tag{3.1.3}$$

Ils montrent également que $\frac{C_{max}^{(MJO)}}{C_{max}^*} \leq 2 - \frac{1}{\max(M^{(1)}, M^{(2)})}$ lorsque la taille des travaux n'excède pas une certaine limite. Concernant la construction de la solution, on peut faire la même remarque que précédemment.

• Langston [la-1987], traite le $FH2, (PM, PM) \mid transport \mid C_{max}$. Le moyen de transport évolue entre les deux étages. Quand la première opération du travail i est terminée, un chariot vient au pied de la machine pour le transporter au pied d'une machine du deuxième étage. Le temps de transport est donné en fonction de la distance séparant deux machines. L'algorithme consiste à aggréger les deux étages en un, en ajoutant la durée d'exécution du premier et du deuxième étage pour chaque travail ; on définit ainsi M machines fictives. Les travaux sont affectés dans un ordre arbitraire à la machine « fictive » disponible le plus tôt. Ainsi, M sous-ensembles de travaux sont construits. Enfin l'affectation de ces M sous-ensembles est

réalisée. Le sous-ensemble ayant la plus petite somme des durées d'exécution au premier étage est affecté au couple de machines de plus petit indice. Cette dernière étape est répétée jusqu'à ce que les M sous-ensembles soient affectés à un couple de machines.

• Gupta [gu-1988], traite le $FH2, (PM^{(1)}, 1) \mid\mid C_{max}$. Il montre que le problème de flow-shop hybride à deux étages est un problème NP complet au sens fort quels que soient $M^{(1)}$ et $M^{(2)}$ tels que $\max\{M^{(1)}, M^{(2)}\} > 1$. Ce résultat est fort intéressant car il montre non seulement la difficulté de ce problème mais également l'intérêt d'étudier de plus près les problèmes de flow-shop hybride à deux étages avant d'étudier ceux à k étages.

L'auteur fait remarquer que la résolution de ce problème revient à minimiser le temps total d'inactivité au deuxième étage. On retrouve ici la même idée que celle énoncée par S. M. Johnson pour le flow-shop traditionnel à deux machines. Gupta propose une heuristique (*cf.* Fig. 6), qui dans un premier temps, établit une liste de priorité pour déterminer une séquence selon l'indice $f(i)$ croissant [pa-1961]. C'est une mise en œuvre de l'algorithme J. Cet indice est défini par la relation (3.1.4). Puis, pour placer les travaux sur les différentes machines du premier étage, il choisit la machine libre le plus tard possible parmi les machines qui engendrent le minimum d'inactivité sur la machine du deuxième étage devant le travail que l'on place. Si cela n'est pas possible c'est la machine libre le plus tôt

Algorithme G : Gupta's Latest Start List Scheduling

Entrées : les travaux, leurs durées d'exécution au premier et deuxième étage et une séquence σ obtenue en classant les travaux par $f(i)$ croissant

Etape 1 :
- i=1

Etape 2 :
- Prendre le travail placé en $i^{ème}$ position dans σ
- Parmi les machines du premier étage, qui minimisent le temps d'inactivité devant ce travail au deuxième étage, choisir celle libre le plus tard
- *Le travail en $i^{ème}$ position dans σ est placé au plus tôt sur la machine j du premier étage et sur la machine du $2^{ème}$ étage*

Etape 3 :
- *Si $i < n$ Alors*
  - i = i+1
  - Aller à l'étape 2
- *Sinon*
  - Fin
- *FinSi*

Figure 6. – Algorithme 1 [gu-1988].

qui est choisie.

$$f(i) = \frac{sign(p_i^{(1)} - p_i^{(2)})}{\min(p_i^{(1)}, p_i^{(2)})}$$

$$\text{avec } sign(p_i^{(1)} - p_i^{(2)}) = \begin{cases} 1 & \text{si } p_i^{(1)} > p_i^{(2)} \\ -1 & \text{si } p_i^{(1)} < p_i^{(2)} \\ \text{« comme on veut »} & \text{sinon.} \end{cases} \quad (3.1.4)$$

Cet algorithme permet de réduire le nombre de machines au premier étage. En effet, la règle d'affectation permet de charger au maximum les machines pourvu que le temps d'inactivité sur la machine du deuxième étage reste minimum. Il n'est pas dit dans l'article que les travaux peuvent se glisser ultérieurement dans les temps d'inactivité introduits sur la machine du deuxième étage. A priori les ordonnancements sont ni actifs ni sans délai sauf si une démonstration montrait que l'ordre fourni par $f(i)$ ne permet jamais de réutiliser les temps d'inactivité de la machine du deuxième étage. L'auteur propose aussi deux bornes inférieures en vue d'une application dans une PSE. Pour cela, considérons les travaux i_1 et i_2 comme étant respectivement le travail ayant la plus petite durée d'exécution et la deuxième plus petite durée d'exécution au premier étage. La première borne inférieure est alors définie par $LB_1 = p_{i_1}^{(1)} + \sum_{i=1}^{n} p_i^{(2)}$. La deuxième, LB_2, prend en considération les deux travaux i_1 et i_2 et plus particulièrement l'incompressible inactivité engendrée au deuxième étage par ces deux travaux, ou bien le « splitting ». Sa définition est donnée par la relation (3.1.5).

$$LB_2 = \max\{LB', LB''\}$$

$$\text{avec } LB' = \max\{p_{i_1}^{(1)}, p_{i_2}^{(1)} - p_{i_1}^{(2)}\} + \sum_{i=1}^{n} p_i^{(2)}$$

$$\text{et} \quad LB'' = \left\lceil \frac{\sum_{i=1}^{n} p_i^{(1)}}{M^{(1)}} \right\rceil + \min_{1 \le i \le n} \{p_i^{(2)}\}. \quad (3.1.5)$$

• Les résolutions du $FH2, (1, PM^{(2)}) \mid\mid C_{max}$ et du $FH2, ((PM)^2) \mid\mid C_{max}$ sont traitées dans [sr-1989]. Les auteurs proposent une heuristique pour la résolution du premier problème et trois heuristiques pour la résolution du

second. Pour le premier problème, l'heuristique H_j correspond à l'algorithme J appliqué au $FH2,(1,PM^{(2)})\ ||\ C_{max}$ dont les durées d'exécution du premier étage et du deuxième étage sont utilisées dans J comme étant respectivement les durées d'exécution sur la première et sur la deuxième machine. Ainsi, on peut déterminer l'ordonnancement par un algorithme de liste pour l'affectation des n travaux au plus tôt, classés par l'algorithme J appliqué au pseudo-problème. On obtient donc un ordonnancement sans délai. Les auteurs déterminent ensuite que selon le nombre de machines et sous certaines conditions, la borne supérieure du quotient $\frac{C_{max}^{(H_j)}}{C_{max}^*}$ n'est pas la même. Ainsi on peut fournir le tableau récapitulatif (*cf.* Tab. 1).

TABLEAU 1
Étude des performances de l'heuristique H_j dans le pire des cas.

Contraintes	Borne supérieure
$M^{(2)}=2$	$\frac{C_{max}^{(H_j)}}{C_{max}^*} \leq 2$
$M^{(2)}=3$ et $C_{max}^{(H_j)} \leq \sum_{i=1}^{n} p_i^{(1)} + \max_{1\leq i\leq n}\{p_i^{(2)}\}$	$\frac{C_{max}^{(H_j)}}{C_{max}^*} \leq 2$
$M^{(2)}=3$ et $C_{max} > \sum_{i=1}^{n} p_i^{(1)} + \max_{1\leq i\leq n}\{p_i^{(2)}\}$	$\frac{C_{max}^{(H_j)}}{C_{max}^*} \leq 1+\left(2-\frac{1}{M^{(2)}}\right)\times\left(1-\frac{1}{M^{(2)}}\right)$

Il est également montré que pour le $FH2,(1,PM^{(2)})\ ||\ C_{max}$, où $M^{(2)} \geq 2$, si $C_{max}^{(Liste)}$ est la valeur de la date d'achèvement total obtenue par un algorithme de liste quelconque, alors $\frac{C_{max}^{(Liste)}}{C_{max}^*} \leq 3 - \frac{1}{M^{(2)}}$.

Nous présentons maintenant les heuristiques obtenues pour le $FH2,((PM)^2)\ ||\ C_{max}$. Elles sont toutes les trois basées sur une décomposition spatiale parfaite du problème considéré : le flow-shop hybride est décomposé en M flow-shops classiques complètement indépendants. On définit $T_i = p_i^{(1)} + p_i^{(2)}$ la durée totale d'exécution du travail i et $T_{ij} = T_i$ si le travail i est affecté au j^{ème} flow-shop et 0 sinon. La technique d'affectation est décrite à l'étape 2 de l'algorithme H_a (*cf.* Fig. 7).

L'étape 3 de l'algorithme H_a (*cf.* Fig. 7) applique la règle de S. M. Johnson dans le cas de stockage inter-étages illimité. Les auteurs signalent que dans le cas sans attente on pourra utiliser un algorithme de résolution du Problème de Voyageur de Commerce. De même s'il y a des temps de

Algorithme H_a

Etape 1 :
Pour j=1 à M *Faire*
Définir le flow-shop j à deux machines, constitué de la machine $M_j^{(1)}$ - j^{ème} machine du premier étage - et de la machine $M_j^{(2)}$ - j^{ème} machine du deuxième étage -.
FinPour
Etape 2 : *On doit résoudre un problème d'affectation* $PM \mid\mid C_{max}$, *c'est-à-dire minimiser* $\max_{1 \leq j \leq M} \sum_{i=1}^{n} T_{ij}$ *où* T_{ij} *est la durée d'exécution du travail* i *affecté au flow-shop* j
Pour H_a, considérer une liste de priorité arbitraire des travaux.
Pour H_b, considérer une liste de priorité des travaux par la règle LPT
Appliquer le principe des algorithmes de liste sur cette liste, en plaçant les travaux au plus tôt.
L'affectation des n travaux est maintenant réalisée
Pour j=1 à M *Faire*
Construire l'ensemble σ_j des travaux qui ont été affectés sur le j^{ème} flow-shop à deux machines.
FinPour
Etape 3 : *On doit résoudre maintenant M flow-shops par l'algorithme J.*
Pour j=1 à M *Faire*
Considérer l'ensemble σ_j.
Appliquer l'algorithme J sur l'ensemble σ_j.
FinPour

Figure 7. – Algorithme 2 [sr-1989].

montage et de démontage ne dépendant pas de la séquence on pourra utiliser les travaux de Sule et Huang [su-1983], s'il y a des décalages temporels on se reportera au travaux de Mitten [mi-1959]... Ils notent que si l'étape 3 était remplacée par un algorithme de liste, la borne supérieure dans le pire des cas de $\frac{C_{max}^{(H_a)}}{C_{max}^*}$ serait inchangée. Cela montre que la phase importante est la deuxième étape.

L'heuristique H_b, proposée par les mêmes auteurs, est similaire à H_a, sauf à l'étape 2 où on construit la liste de priorité par LPT. Graham [gr-1966] a montré, pour la résolution du problème $PM \mid\mid C_{max}$, que pour un algorithme de liste arbitraire LS : $\frac{C_{max}^{(LS)}}{C_{max}^*} \leq 2 - \frac{1}{M}$ et pour l'algorithme LPT : $\frac{C_{max}^{(LPT)}}{C_{max}^*} \leq \frac{4}{3} - \frac{1}{3 \times M}$.

Dans l'algorithme approché H_c l'affectation des n travaux sur les M machines identiques est résolue par un algorithme optimal.

Les auteurs donnent la borne supérieure du quotient $\frac{C_{max}^{(Heuristique)}}{C_{max}^*}$ pour les heuristiques H_a, H_b et H_c (*cf.* Tab. 2). Cette table montre que le rapport $\frac{C_{max}}{C_{max}^*}$, dans le pire des cas, n'est jamais inférieur à 2.

En outre ils montrent que dans le $FHk, ((PM)^k) \mid\mid C_{max}$, où $M \geq 2$, on a $\frac{C_{max}^{(H_a)}}{C_{max}^*} \leq k + 1 - \frac{1}{M}$ en utilisant un algorithme de liste à la place de l'algorithme J, à l'étape 3, pour résoudre le problème de séquencement. Les

TABLEAU 2
Résultat des performances des heuristiques H_a, H_b et H_c.

Heuristiques	Borne supérieure
H_a	$\frac{C_{max}^{(H_a)}}{C_{max}^*} \leq 3 - \frac{1}{M}$
H_b	$\frac{C_{max}^{(H_b)}}{C_{max}^*} \leq r$ où $\frac{7}{3} - \frac{2}{3 \times M} \leq r \leq 3 - \frac{1}{M}$
H_c	$\frac{C_{max}^{(H_c)}}{C_{max}^*} \leq r$ où $2 \leq r \leq 3 - \frac{1}{M}$

auteurs ne donnent aucune précision sur cet algorithme de séquencement. On peut penser que cela se fait au plus tôt sans délai étage par étage.

Voici une application de l'algorithme H_a :

Considérons un $FH2, ((PM)^2) \mid\mid C_{max}$ avec huit travaux à ordonnancer et $M = 2$. Les durées d'exécution des travaux sont données dans le tableau 3.

TABLEAU 3
Données de l'exemple d'exécution de l'algorithme H_a.

N° travail	1	2	3	4	5	6	7	8
$p_i^{(1)}$	7	5	6	7	1	2	2	3
$p_i^{(2)}$	3	6	1	2	2	7	1	1
T_{ij}	10	11	7	9	3	9	3	4

Étape 2 : On bâtit une liste L arbitraire. Prenons par exemple $L = (1, 2, 3, 4, 5, 6, 7, 8)$.

On a $\sigma_1 = \{1, 3, 5, 6\}$ et $\sigma_2 = \{2, 4, 7, 8\}$.

Étape 3 : On a deux flow-shops à 2 machines. Le tableau 4 montre le résultat de l'affectation.

Pour le travail numéro 6, on a le choix : soit l'affecter sur le flow-shop numéro 1 soit sur le flow-shop numéro 2. Les deux solutions sont les mêmes. À l'étape 3, comme on est dans un cas où le stock inter-étages est illimité, on applique l'algorithme J.

TABLEAU 4
Données de l'exemple d'exécution de l'algorithme H_a.

N° travail	1	2	3	4	5	6	7	8
flow-shop N° 1	• (10)		• (17)		• (20)	• (29)		
flow-shop N° 2		• (11)		• (20)			• (23)	• (27)

• Pour le flow-shop numéro 1, l'ordre de passage des travaux affectés sur ce flow-shop est $O(1) = (5, 6, 1, 3)$ et le C_{max} est égal à 17.

• Pour le flow-shop numéro 2, l'ordre de passage des travaux affectés sur ce flow-shop est $O(2) = (2, 4, 7, 8)$ et le C_{max} est égal à 18. Donc la valeur du C_{max} donné par l'algorithme H_a est 18.

Les trois heuristiques ne fournissent pas toujours des ordonnancements sans délai. En effet une machine peut être laissée inoccupée alors qu'un travail attend une autre machine.

• Gupta et Tunc [gu-1991] proposent une étude du problème $FH2, (1, PM^{(2)}) \mid\mid C_{max}$. Ce problème est le problème inverse de celui abordé dans [gu-1988] et reprend dans l'esprit, l'algorithme qui a été donné en 1988. Les auteurs montrent que le problème est NP-difficile et fournissent deux heuristiques (*cf.* Fig. 8) qui seront réutilisées dans une PSE. La première heuristique détermine dans un premier temps une liste de

Algorithme de l'Heuristique 1 et 2

Etape 1 : *Détermination d'une séquence*
  Pour l'heuristique 1 :
    Soit $S = (i_1, i_2, ..., i_n)$ la liste des travaux classés par $f(i)$ croissant
    S'il y a égalité, favoriser le travail ayant la plus petite durée d'exécution au premier étage ou la plus grande durée d'exécution au deuxième étage.
  Pour l'heuristique 2 :
    Soit $S = (i_1, i_2, ..., i_n)$ la liste des travaux classés selon LPT. S'il y a égalité, favoriser le travail ayant la plus petite durée d'exécution au premier étage
Etape 2 : *Affectation de la séquence obtenue*
  Pour $p = 1$ à n *Faire*
    Soit Ω l'ensemble des machines du deuxième étage dont la date de disponibilité est inférieure à la date de fin d'exécution du travail i_p au premier étage.
    Si Ω est vide *Alors*
      Affecter le travail i_p à la machine libre la première
    sinon
      Affecter le travail i_p sur la machine de Ω libre la dernière.
    FinSi
  FinPour
Etape 3 :
  La solution finale est la solution approchée pour le problème.

Figure 8. – Algorithme 3 [gu-1991].

travaux ordonnés selon les $f(i)$ croissants [pa-1961]. Ensuite, l'affectation des travaux sur les machines du deuxième étage se fait sur la machine libre le plus tard, permettant au travail de ne pas attendre et, si ce n'est pas possible, sur la machine qui minimise le temps d'attente du travail. Les ordonnancements construits sont donc sans délai.

La deuxième heuristique est basée sur [la-1987] : il affirme qu'il est préférable d'ordonnancer les travaux, à l'étape 1, selon la règle LPT, appliquée sur les durées d'exécution du deuxième étage, quand le deuxième étage domine le premier (c'est-à-dire que la moyenne des durées d'exécution au deuxième étage est supérieure à la moyenne des durées d'exécution au premier étage). Elle permet également de minimiser le nombre de machines au deuxième étage (au plus égal, évidemment, au nombre de travaux à ordonnancer). Dans le cas où le nombre de machines au deuxième étage est supérieur ou égal au nombre de travaux, il suffit de classer les travaux par ordre décroissant de leur durée d'exécution au deuxième étage pour minimiser le temps d'achèvement total.

Gupta et Tunc fournissent ensuite trois bornes inférieures, LB_1, LB_2 et LB_3 qu'ils utiliseront dans une amélioration de la PSE de [br-1991]. Ils proposent de prendre comme borne supérieure, UB en entrée de la PSE, la meilleure des deux solutions données par les deux heuristiques.

$$LB_1 = \sum_{i=1}^{n} p_i^{(1)} + \min_{1 \leq i \leq n} \{p_i^{(2)}\}$$

$$LB_2 = M^*$$

$$LB_3 = \min_{1 \leq i \leq n} \{p_i^{(1)}\} + \left\lceil \frac{\sum_{i=1}^{n} p_i^{(2)}}{M^{(2)}} \right\rceil .$$

M^* désigne la valeur du temps total d'achèvement lorsqu'on considère le nombre de machines au deuxième étage supérieur ou égal au nombre de travaux. Cet ordonnancement est obtenu en classant les travaux par ordre décroissant des durées d'exécution au deuxième étage.

Plus le nombre de travaux est important, meilleurs semblent être les résultats des deux heuristiques. Les auteurs soulignent que l'heuristique $H2$ est meilleure.

• Deal et Hunsucker [de-1991] traitent le $FH2,((PM)^2) \mid\mid C_{max}$. Pour ce problème, ils donnent de nouvelles bornes inférieures :

$$LB_1 = \max_{1 \leq i \leq n} \{p_i^{(1)} + p_i^{(2)}\}$$

$$LB_2 = \max\left\{ \frac{1}{M} \times \left(\sum_{i=1}^{n} p_i^{(1)} \right) + \min_{1 \leq i \leq n} (p_i^{(2)}); \right.$$

$$\left. \min_{1 \leq i \leq n} (p_i^{(1)}) + \frac{1}{M} \times \left(\sum_{i=1}^{n} p_i^{(2)} \right) \right\}$$

$$LB = \max\{LB', LB''\}$$

$$\text{avec } LB' = \frac{1}{M} \times \left\{ \sum_{i=1}^{n} p_i^{(2)} + \sum_{h=1}^{M} p^{(1)}(h) \right\}$$

$$\text{et } LB'' = \frac{1}{M} \times \left\{ \sum_{i=1}^{n} p_i^{(1)} + \sum_{h=1}^{M} p^{(2)}(h) \right\}$$

où $p^{(\ell)}(h)$ est la h^{ième} plus petite durée d'exécution à l'étage ℓ.

Les auteurs comparent ensuite ces bornes et les résultats obtenus par :

1. un ordonnancement au hasard.
2. un ordonnancement produit par l'algorithme J en considérant comme durée d'exécution d'un travail sur la première machine (resp. sur la deuxième machine) la durée d'exécution du travail au premier étage (resp. au deuxième étage). L'affectation des travaux au premier étage revient à considérer la liste de priorité établie par J, et à affecter à chaque machine un travail dès que cela est possible. Pour l'affectation au deuxième étage, il faut considérer une liste FIFO, c'est-à-dire que le travail le premier arrivé dans la file est le premier à s'exécuter sur une machine disponible au deuxième étage (le travail le premier arrivé dans la file est celui qui a fini de s'exécuter le premier au premier étage). Les ordonnancements obtenus sont sans délai (pas d'inoccupation des machines au premier étage et utilisation de la règle FIFO au deuxième étage).
3. l'algorithme SPT appliqué aux durées d'exécution des travaux au premier étage. L'affectation des travaux se fait comme dans le cas précédent. Les ordonnancements obtenus sont à nouveau sans délai.

Les résultats obtenus par l'heuristique qui utilise l'algorithme J sont les meilleurs.

• Lee, Cheng et Lin [lee-1993] s'intéressent à un problème d'ordonnancement de type flow-shop hybride à deux étages pour lequel le « splitting » est imposé uniquement au premier étage. La taille des sous-lots est une donnée du problème et chaque travail se décompose, au premier étage, en deux sous-lots qui y passent en parallèle. Pour que le travail puisse commencer son exécution sur le deuxième étage, il faut que les deux sous-lots aient fini leur exécution sur le premier étage. Il s'agit du $FH2,(P2,1) \mid split^{(1)}$, tailles des sous-lots imposées sur chaque machine $\mid C_{max}$. Cela engendre donc naturellement des temps d'inactivité sur la machine du deuxième étage. C'est un problème NP-complet au sens fort. D'abord les auteurs proposent des conditions à satisfaire pour résoudre le problème de façon optimale pour des cas triviaux.

• (C1) Si $p_{i1}^{(1)} \geq p_{i2}^{(1)}$ (ou $p_{i1}^{(1)} \leq p_{i2}^{(1)}$) pour tout i, alors appliquer l'algorithme J de Johnson sur l'ensemble des travaux $J' = \{J'_1, ..., J'_n\}$ où les durées d'exécution du travail J'_i sont données par $(p_{i1}^{(1)}, p_i^{(2)})$ (ou $p_{i2}^{(1)}, p_i^{(2)}$).

THÉORÈME 1 : *Si J_i et J_j sont deux travaux consécutifs dans un ordonnancement et si $p_i^{(2)} \leq \min\{p_{i1}^{(1)}, p_{i2}^{(1)}, p_j^{(2)}\}$ alors la date de fin d'exécution d'un ordonnancement dans lequel J_j est avant J_i n'est pas pire que celle d'un ordonnancement où J_i est avant J_j.*

COROLLAIRE 1 : *Si $p_k^{(2)} = \min\left(p_{k1}^{(1)}, p_{k2}^{(1)}, \min_{1 \leq i \leq n}(p_i^{(2)})\right)$ alors il y a un ordonnancement optimal dans lequel J_k est placé le dernier.*

• (C2) Si $p_i^{(2)} \leq \min\{p_{i1}^{(1)}, p_{i2}^{(1)}\}$ pour tout i, alors il existe un ordonnancement optimal où les travaux sont classés par ordre non croissant des $p_i^{(2)}$, *i.e.* LPT (c'est-à-dire qu'il faut minimiser la date de fin d'exécution sur la machine du deuxième étage d'un ensemble de travaux dont chacun a une date de début au plus tôt r_i imposée ($r_i = \max\{p_{i1}^{(1)}, p_{i2}^{(1)}\}$)).

• (C3) Si J_i et J_j sont deux travaux, satisfaisant chacun à $p_h^{(2)} \geq \max\{p_{h1}^{(1)}, p_{h2}^{(1)}\}$ pour $h \in \{i,j\}$ et si $\max\{t_1 + p_{i1}^{(1)}, t_2 + p_{i2}^{(1)}\} \leq \max\{t_1 + p_{j1}^{(1)}, t_2 + p_{j2}^{(1)}\}$ où t_1 et t_2 indiquent la date de fin d'exécution des travaux déjà ordonnancés sur la machine 1 et sur la machine 2 de l'étage 1 avant de placer les travaux J_i et J_j, alors il faut placer J_i avant J_j dans un ordonnancement optimal. Les auteurs fournissent un algorithme qui permet de construire un ordonnancement optimal en un temps polynomial lorsque la condition C3 est vérifiée.

Ensuite trois heuristiques sont proposées (H_1, H_2 et H_3) et présentées figure 9. Les travaux sont ordonnancés par l'algorithme J utilisant des pseudo-temps d'exécution sur l'étage 1, puis l'affectation est réalisée au plus tôt. Les trois heuristiques ne diffèrent que dans la façon de bâtir les pseudo-temps d'exécution sur une machine fictive qui remplace le premier étage. Les ordonnancements générés sont sans délai si la règle FIFO est utilisée pour l'étage 2. Les bornes supérieures de $\frac{C_{max}^{(Hi)}}{C_{max}^{*}}$ pour $i = 1, 2, 3$ sont respectivement 2, 2 et 1.5.

Algorithme des heuristiques H1, H2 et H3

Etape 1 : *Construction du pseudo-temps d'exécution au premier étage*

Pour l'heuristique H1

$p_i^{(1)} = \max(p_{i1}^{(1)}, p_{i2}^{(1)})$, $\forall i = 1, ..., n$

Pour l'heuristique H2

Si $\sum_{i=1}^{n} p_{i1}^{(1)} \geq \sum_{i=1}^{n} p_{i2}^{(1)}$ *Alors*

$p_i^{(1)} = p_{i1}^{(1)}$, $\forall i = 1, ..., n$

Sinon

$p_i^{(1)} = p_{i2}^{(1)}$, $\forall i = 1, ..., n$

FinSi

Pour l'heuristique H3

$p_i^{(1)} = \frac{1}{2} \times \left(p_{i1}^{(1)} + p_{i2}^{(1)}\right)$, $\forall i = 1, ..., n$

Etape 2 :

Bâtir un pseudo-problème à deux machines fictives uniquement (une par étage) tel que les durées d'exécution des travaux sur la première machine fictive sont égales à $p_i^{(1)}$ et celles sur la deuxième machine égales à $p_i^{(2)}$.

Etape 3 :

Appliquer l'algorithme J de Johnson sur ce pseudo-problème.

Etape 4 :

La solution obtenue pour le pseudo-problème devient la solution pour le problème initial. L'affectation des lots se fait au plus tôt. Un travail ne pourra commencer à s'exécuter sur le deuxième étage que lorsque les deux lots au premier étage auront fini de s'exécuter.

Figure 9. – Algorithme 4 [lee-1993].

Une PSE inspirée des travaux de Ignall et Schrage [ig-1965] est ensuite présentée. La borne inférieure pour un ordonnancement partiel σ, $LB(\sigma)$ est définie par :

$$LB(\sigma) = \max\Bigg\{ t_1^{(2)}(\sigma) + \sum_{i \in N - J(\sigma)} p_i^{(2)}$$

$$+ \min_{i \in N - J(\sigma)} \Big\{ \Big[\max(t_1^{(2)}(\sigma), t_1^{(1)}(\sigma) + p_{i1}^{(1)}, t_2^{(1)}(\sigma) + p_{i2}^{(1)}) - t_1^{(2)}(\sigma) \Big] \Big\};$$

$$TT_{max} + \min_{i \in N - J(\sigma)} p_i^{(2)} \Bigg\}$$

où $J(\sigma)$ est l'ensemble des travaux déjà ordonnancés, $t_j^{(\ell)}(\sigma)$ est la date de fin d'exécution des travaux de σ sur la machine j de l'étage ℓ et $TT_{max} = \max\left\{\sum_{i=1}^{n} p_{i1}^{(1)}; \sum_{i=1}^{n} p_{i2}^{(1)}\right\}$.

• Gupta et Tunc [gu-1994] considèrent le $FH2,(1,PM^{(2)}) \mid S_{nsd}, R_{nsd} \mid C_{max}$. Ils proposent différentes définitions du C_{max} pour un ordonnancement P : le « Makespan machine-based » général ($\max_{1\leq i\leq n}\{C_i^{''(1)}, C_i^{''(2)}\}$), le « Makespan job-based » ($\max_{1\leq i\leq n} C_i^{'(2)}$), et le « Makespan machine-based » dégradé ($\max_{1\leq i\leq n} C_i^{''(2)}$). $C_i^{'(1)}$ et $C_i^{'(2)}$ représentent respectivement les dates de fin d'exécution du travail i à l'étage 1 et 2. Ces dates ne comprennent pas les temps de démontage du travail i sur la machine. $C_i^{''(1)}$ et $C_i^{''(2)}$ représentent les dates de fin de démontage après exécution du travail i respectivement à l'étage 1 et 2. $S_i^{(\ell)}$ (resp. $R_i^{(\ell)}$) représente le temps de montage d'outils (resp. de démontage) avant (après) l'exécution du travail i à l'étage ℓ. En notant h (resp. $i-1$) le travail qui précède immédiatement le travail i sur la machine du deuxième étage (resp. du premier étage), on a :

$$
\begin{aligned}
&C_0^{'(1)} = C_0^{'(2)} = C_0^{''(1)} = C_0^{''(2)} = 0\\
&C_i^{'(1)} = C_{i-1}^{''(1)} + S_i^{(1)} + p_i^{(1)}\\
&C_i^{''(1)} = C_i^{'(1)} + R_i^{(1)}\\
&C_i^{'(2)} = \max\{C_i^{'(1)}; C_h^{''(2)} + S_i^{(2)}\} + p_i^{(2)}\\
&C_i^{''(2)} = C_i^{'(2)} + R_i^{(2)}.
\end{aligned}
$$

Deux bornes et deux théorèmes sont également donnés :

$LB_1 = \sum_{i=1}^{n}(S_i^{(1)} + p_i^{(1)} + R_i^{(1)}) - \max_{1\leq i\leq n}(R_i^{(1)}) + \min_{1\leq i\leq n}(p_i^{(2)})$. Le C_{max} ne peut être inférieur à la somme de la charge sur le premier étage plus la plus petite durée d'exécution au deuxième étage. Cette borne est une borne « job-based » c'est-à-dire qu'on ne tient pas compte du temps de démontage à la fin du dernier travail de l'ordonnancement.

$LB_2 = M^*$, la valeur du C_{max} obtenue en considérant un nombre illimité de machines au deuxième étage [gu-1991]. C'est un cas limite.

THÉORÈME 2 : *Quand le nombre de machines $M^{(2)}$ est illimité, l'ordonnancement obtenu en arrangeant les travaux dans l'ordre des $h(i)$ décroissants (avec $h(i) = p_i^{(2)} - R_i^{(1)}$) minimise le « makespan job-based ».*

THÉORÈME 3 : *Quand le nombre de machines $M^{(2)}$ est illimité, l'ordonnancement obtenu en arrangeant les travaux dans l'ordre des $h'(i)$ décroissants (avec $h'(i) = p_i^{(2)} - R_i^{(1)} + R_i^{(2)}$) minimise le « makespan machine-based ».*

Enfin, les auteurs proposent quatre heuristiques (*cf.* Fig. 10 pour l'algorithme de la première) pour résoudre ce problème. Pour les $F2 \mid S_{nsd}, R_{nsd} \mid C_{max}$ (« machine based » dégradé) ou $F2 \mid S_{nsd}, R_{nsd} \mid C'_{max}$ ($C'_{max} = \max_{1 \leq i \leq n} C_i^{'(2)}$), on connaît deux règles optimales de classement : l'algorithme SH [su-1983] et l'algorithme SG [sz-1987]. Elles sont utilisées dans les heuristiques proposées ici. Les heuristiques suivent le même schéma : dans un premier temps une séquence est déterminée à partir d'un algorithme (SH, SG, LPT ou par $h(i)$ décroissant), puis dans un deuxième temps l'affectation des travaux se fait dans l'ordre de la séquence sans délai. Il semblerait que des ordonnancements sans délai soient générés par ces quatre procédures.

Algorithme de l'heuristique 1

Etape 1 : *Détermination d'une séquence*
Soit $O = (1, 2, 3, ..., n)$ l'ordonnancement obtenu par l'algorithme SH et retenu pour l'étage 1.
Etape 2 : *Affectation des travaux sur le deuxième étage, pour le premier il n'y a pas de difficulté car il n'y a qu'une machine*
Pour $i = 1$ à n *Faire*
Retirer le travail i de O
Soit Ω l'ensemble des machines du deuxième étage dont la date de disponibilité est inférieure à la date de fin d'exécution (comptée avant le démontage) du travail i au premier étage.
Si Ω est vide *Alors*
Affecter le travail i à la machine qui minimise le temps d'attente du travail i (avant de débuter son exécution - sans compter le temps de montage - au deuxième étage).
Sinon
Affecter le travail i sur la machine de Ω libre la dernière.
FinSi
FinPour

Figure 10. – Algorithme 5 [gu-1994].

Dans l'heuristique 2, SH est remplacé par les deux premières étapes de SG qui fournissent p séquences. On applique l'étape 2 sur chacune d'elles.

L'heuristique 3 ne diffère de l'heuristique 1 que par la première étape : O est la séquence obtenue par LPT appliquée au deuxième étage.

Il en est de même pour l'heuristique 4 : O est obtenue en classant les travaux dans l'ordre des $h(i)$ décroissants.

Pour les heuristiques 3 et 4, en cas d'égalité, on favorise le travail ayant la plus petite durée d'exécution au premier étage.

• Pour le $FH2, ((PM^{(\ell)})_{\ell=1}^{2}) \mid\mid C_{max}$, Lee et Vairaktarakis [lee-1994] proposent une heuristique H qui détermine un ordonnancement S (en appliquant J sur $p'_{i1} = \frac{p_i^{(1)}}{M^{(1)}}$ et $p'_{i2} = \frac{p_i^{(2)}}{M^{(2)}}$), puis qui affecte les travaux de S au premier étage en utilisant un algorithme de liste appelé FAM (First Available Machine) et au deuxième étage par l'algorithme LBM (Last Busy Machine). Enfin un réordonnancement est effectué au deuxième étage en classant les travaux par date de fin d'exécution croissante au premier étage et en les plaçant au plus tôt. L'ordonnancement construit n'est pas sans délai car l'affectation obtenue au deuxième étage n'est pas remise en cause et il peut y avoir des travaux qui pourraient être réalisés par d'autres machines moins chargées.

L'algorithme LBM (Fig. 11) consiste à affecter un à un les travaux, dans l'ordre inverse d'une liste S, sur la machine la moins occupée du deuxième étage. C'est en fait un algorithme de liste appliqué au problème inverse.

Algorithme Last Busy Machine (LBM) ou algorithme de liste sur problème inverse

Entrée : T une très grande valeur et S une liste.
Etape 1 :
  Soit $t_j = T$, $\forall j = 1, ..., M^{(2)}$ date à partir de laquelle la machine j du $2^{ème}$ étage n'est plus libre.
Etape 2 :
  Soit i le travail non affecté à une des machines du $2^{ème}$ étage placé le dernier dans la séquence S et m la machine ayant le plus grand t_j. Ordonnancer le travail i sur la machine m.
Etape 3 :
  $t_j = t_j - p_i^{(2)}$, $S = S - \{i\}$
  Si $S \neq \emptyset$ *Alors*
    Aller à l'étape 2
  Sinon FIN

Figure 11. – Algorithme 6 [lee-1994].

Les auteurs fournissent également des bornes inférieures. Considérons $(J1, J2, ..., Jn)$ les n travaux classés par ordre décroissant de leur durée d'exécution au premier étage, et $P_h^{(\ell)}$ la somme des h plus petites durées d'exécution à l'étage ℓ.

Les bornes LB_3 et LB_4 sont définies pour le problème inverse, c'est-à-dire que le nombre de machines au premier étage est égal à celui du deuxième

et inversement (les formules sont donc symétriques).

Si $M^{(1)} \geq M^{(2)}$ Si $M^{(2)} \geq M^{(1)}$

$$LB_1 = \frac{P^{(1)}_{M^{(2)}} + P^{(2)}_n}{M^{(2)}} \qquad LB_2 = \frac{P^{(1)}_{M^{(2)}} + (M^{(2)} - M^{(1)}) \times P^{(1)}_1 + P^{(2)}_n}{M^{(2)}}$$

$$LB_3 = \frac{P^{(1)}_{M^{(1)}} + (M^{(1)} - M^{(2)}) \times P^{(2)}_1 + P^{(1)}_n}{M^{(1)}} \qquad LB_4 = \frac{P^{(2)}_{M^{(1)}} + P^{(1)}_n}{M^{(1)}}.$$

Dans l'expression de LB_1, $\frac{P^{(1)}_{M^{(2)}}}{M^{(2)}}$ représente l'attente minimale et $\frac{P^{(2)}_n}{M^{(2)}}$ la durée d'exécution des travaux au deuxième étage.

Dans l'expression de LB_2, $(M^{(2)} - M^{(1)}) \times P^{(1)}_1$ représente l'inactivité engendrée au deuxième étage pour qu'au moins $M^{(2)} - M^{(1)}$ travaux soient exécutés au premier étage.

Finalement, on a

$$LB = \max\{LB_1, LB_3, C_{LB}\} \qquad \text{ou} \qquad LB = \max\{LB_2, LB_4, C_{LB}\}$$

où C_{LB} est la valeur du C_{max} pour le pseudo-problème défini à l'étape 1 de l'heuristique H. Les auteurs notent que leur heuristique est meilleure que celles rencontrées jusqu'alors. Il semblerait qu'ils n'aient pas tenu compte des travaux exposés dans [bu-1973] qui eux obtiennent en utilisant MJO (Modified Johnson Order) la même borne pour le rapport $\frac{C_{max}}{C^*_{max}}$.

• Gupta, Hariri et Potts s'intéressent au $FH2, (PM^{(1)}, 1) \mid\mid C_{max}$ [gup-1995]. Ils considèrent que les durées d'exécution au premier étage et au deuxième étage sont des entiers.

Ils développent une PSE pour laquelle cinq bornes inférieures, notées LB_1 à LB_5 sont définies et donnent des algorithmes de type heuristique.

$$LB_1 = \max\left\{ \min_{1\leq i\leq n}(p^{(1)}_i) + \sum_{i=1}^{n} p^{(2)}_i, \left\lceil \frac{\sum_{i=1}^{n} p^{(1)}_i}{M^{(1)}} \right\rceil + \min_{1\leq i\leq n}(p^{(2)}_i) \right\}.$$

La borne LB_1 est calculée en $O(n)$.

$$LB_2 = \max_{1 \leq j \leq n} \left\{ \left\lceil \frac{\sum_{i=1}^{j} p_{\sigma(i)}^{(1)}}{M^{(1)}} + \sum_{i=j}^{n} p_{\sigma(i)}^{(2)} \right\rceil \right\}$$

où σ est la séquence des travaux obtenue en appliquant l'algorithme J, en prenant pour durées d'exécution sur la première machine et la deuxième machine : $p_i^{(1)}/M^{(1)}$ et $p_i^{(2)}$. Cette borne est calculée en $O(n \log n)$.

$$LB_3 = \max_{1 \leq j \leq n} \left\{ p_{\pi(j)}^{(1)} + \sum_{i=j}^{n} p_{\pi(i)}^{(2)} \right\}.$$

La séquence π est obtenue en appliquant SPT à l'étage 1. $LB3$ est calculée en $O(n \log n)$.

Soit N_0 un sous-ensemble de l'ensemble N de travaux et n_0 sa cardinalité. Les auteurs définissent une borne inférieure pour ce sous-ensemble par :

$$LB_0(N_0) = \frac{\sum_{i \in N_0} p_i^{(1)}}{M^{(1)}} + \frac{\sum_{i \in N - N_0} \min\{p_i^{(1)}, p_i^{(2)}\}}{M^{(1)}} + \frac{\sum_{h=1}^{M^{(1)}} h \times p_{j_{M^{(1)}-h+1}}^{(2)}}{M^{(1)}}$$

avec $(j_1, j_2, ..., j_{n_0})$ l'ordre des travaux de N_0 tel que $p_{j_1}^{(2)} \leq p_{j_2}^{(2)} \leq \ldots \leq p_{j_{n_0}}^{(2)}$.

LB_4 est définie par $LB_4 = \lceil LB_0(N) \rceil$. Cette borne est calculée en $O(n + M^{(1)} \log M^{(1)})$.

Enfin, la borne LB_5 est définie par $LB_5 = \lceil LB_0(N_0) \rceil$ avec N_0 défini par la procédure présentée figure 12. Cette procédure permet de réduire la cardinalité de N_0 initialement égal à N. Cette borne est calculée en $O(nM^{(1)})$.

Prenons par exemple un $FH2, (P2, 1) \mid\mid C_{max}$ où il y a cinq travaux à ordonnancer (*cf.* Tab. 5).

Étape 1 : $N_0 = (1, 2, 3, 4, 5)$. On trouve $T = 10 - 0.5 \times 6 = 7$, $LB = 0$.

I^{re} itération : Étapes 2 et 3

$j = 2$, $N - N_O = \{2\}$, $N_0 = \{1, 3, 4, 5\}$, $LB_5 = \left\lceil \frac{16}{2} + \frac{1}{2} \times \min(1, 4) + \frac{1}{2} \times (1 \times 2 + 2 \times 2) \right\rceil = \lceil 11, 5 \rceil = 12$, $LB = 12$.

Procédure de calcul LB5

Etape 1 :
- $N_0 = \{1, 2, ..., n\} = N$
- Calculer $T = \frac{\sum_{i=1}^{n} p_i^{(1)}}{M^{(1)}} - (1 - \frac{1}{M^{(1)}}) \times \max_{1 \leq i \leq n} p_i^{(1)}$
- *Jusqu'à l'instant T on est certain que toutes les machines du premier étage sont occupées dans un ordonnancement optimal.*
- $LB = 0$

Etape 2 :
- Choisir $j \in N_0$ tel que $p_j^{(2)} = \min_{i \in N_0} p_i^{(2)}$
- *Si* $\sum_{i \in N - N_0} p_i^{(1)} + p_j^{(1)} < T$ *Alors*
  - Retirer j de N_0 (j *est donc maintenant dans* $N - N_0$)
- *Sinon*
  - Aller à l'étape 4
- *FinSi*

Etape 3 :
- Calculer $LB_5 = \lceil LB_0(N_0) \rceil$
- *Si* $LB_5 < LB$ *Alors*
  - Aller à l'étape 4
- *Sinon*
  - $LB = LB_5$
  - Aller à l'étape 2
- *FinSi*

Etape 4 :
- Fin

Figure 12. – Algorithme 7 [gup-1995].

TABLEAU 5
Données pour le calcul de LB_5.

N° du travail	$p_i^{(1)}$	$p_i^{(2)}$
1	1	2
2	4	1
3	6	7
4	3	2
5	6	3

2e itération : Étapes 2 et 3

$j = 1$, $N - N_O = \{1, 2\}$, $N_0 = \{3, 4, 5\}$, $LB_5 = \lceil \frac{15}{2} + \frac{1}{2} \times (\min(1, 2) + \min(4, 1)) + \frac{1}{2} \times (1 \times 3 + 2 \times 2) = \lceil 12 \rceil = 12$, $LB = 12$.

3e itération : Étapes 2 et 4

$j = 4$, Fin car on ne peut plus extraire de travaux de N_0 sans violer le test de l'étape 2.

La PSE développée par les auteurs utilise pour stratégie de recherche « la profondeur d'abord ». Chaque nœud de l'arbre est constitué par une

solution d'ordonnancement partiel sur les machines du premier étage. Pour déterminer la valeur finale du critère on applique le théorème suivant :

THÉORÈME 4 : *Pour un ordonnancement donné au premier étage, un ordonnancement optimal au deuxième étage est trouvé en $O(n \log n)$ en classant les travaux dans l'ordre non décroissant de leur date de fin d'exécution au premier étage.*

Une borne supérieure initiale pour la PSE est déterminée par une des heuristiques énoncées ci-après. Toutes ces heuristiques semblent fournir des ordonnancements sans délai. Cependant les auteurs ne fournissent aucune information à ce sujet.

La figure 13 présente les algorithmes de construction de séquences, J1 et J2, qui vont être utilisés dans la conception de plusieurs heuristiques. Dans les cas les plus simples, l'affectation des travaux au premier étage est faite selon un algorithme de liste utilisant une séquence σ (obtenue par J1 ou J2) et l'ordonnancement sur la machine du deuxième étage utilise la règle FIFO : c'est le cas de l'heuristique L.

Algorithmes de l'heuristique J1 et J2 de construction d'une séquence

Etape 0 :
- Pour J1
  - $\emptyset$
- Pour J2
  - $p_i^{(1)} = p_i^{(1)}/M^{(1)}\ \forall i = 1, ..., n$ et $p_i^{(2)}$ inchangés

Etape 1 :
- Soit Ω tel que $\Omega = \{i \mid p_i^{(1)} \leq p_i^{(2)}\}$
- Bâtir une séquence σ en appliquant la règle SPT aux travaux de Ω

Etape 2 :
- Compléter la séquence σ en appliquant la règle LPT aux travaux de $N - \Omega$

Figure 13. – Algorithme 8 [gup-1995].

Les algorithmes L′ et M′ (*cf.* Figs. 14 et 15) construisent au moyen d'algorithmes de liste des solutions du problème inverse, puis retournent au problème initial. L'algorithme R effectue a posteriori un calage à gauche de l'ordonnancement ainsi obtenu.

Les algorithmes L′ et M′ peuvent générer des temps d'inactivité sur les machines. Pour remédier à cela, les auteurs fournissent l'algorithme R qui permet de recaler l'ensemble des travaux au plus tôt. Les algorithmes L, G, L′ et M′, avec en entrée les ordonnancements issus des algorithmes J1 ou J2, ont fait l'objet d'une étude de performance dans le pire des cas.

Algorithme L' : List Scheduling for Inverse Problem

Entrée : un ordonnancement σ a priori

Etape 1 : *On traite le problème inverse (P') du problème (P) i.e les étages sont inversés*

- $p_i'^{(1)} = p_i^{(2)}$, $p_i'^{(2)} = p_i^{(1)}$, $\forall i = 1, ..., n$
- $\sigma'(i) = \sigma(n - i + 1)$, $\forall i = 1, ..., n$
- $T = 0$
- $T_j = 0$, $\forall j = 1, ..., M^{(1)}$, date de disponibilité de la machine j du deuxième étage de P'
- $\pi'_j = \emptyset$, $\forall j = 1, ..., M^{(1)}$, séquence des travaux affectés à la machine j du $2^{ème}$ étage de P'
- $\pi_j = \emptyset$, $\forall j = 1, ..., M^{(1)}$, séquence des travaux affectés à la machine j du 1^{er} étage de P
- i = 1

Etape 2 : *Affectation d'un travail au premier étage du problème inverse*

Séquencer le travail placé en $i^{ème}$ position dans σ' sur la machine du premier étage, à l'instant T. On pose $T = T + p'^{(1)}_{\sigma'(i)}$, et $C'^{(1)}_{\sigma'(i)} = T$

Etape 3 : *Affectation du même travail au $2^{ème}$ étage du problème inverse*

Sélectionner la machine j du deuxième étage (pour le problème inverse) dont le T_j est le plus petit

$T_j = \max\{T, T_j\} + p'^{(2)}_{\sigma'(i)}$, $C'^{(2)}_{\sigma'(i)} = T_j$, $\pi'_j = (\pi'_j, \sigma'(i))$

Si $i < n$ Alors

- i=i+1
- Aller à l'étape 2

FinSi

Etape 4 : *Inversion de la solution obtenue avec calage au plus tôt sur le problème direct, avec n_j le nombre de travaux affectés à la machine j*

- $C_{max} = \max_{1 \leq j \leq M^{(1)}} T_j$
- $C_i^{(1)} = C_{max} - C_i'^{(2)} + p_i^{(1)}$ et $C_i^{(2)} = C_{max} - C_i'^{(1)} + p_i^{(2)}$, $\forall i = 1, ..., n$
- $\pi_j(i) = \pi'_j(n_j - i + 1)$, $\forall i = 1, ..., n_j$ et $\forall j = 1, ..., M^{(1)}$

Figure 14. – Algorithme 9 [gup-1995].

TABLEAU 6

Étude de la complexité des algorithmes et du rapport C_{max}/C^*_{max}.

Algorithme	Algorithme en entrée	Temps de calcul	ratio C_{max}/C^*_{max}
L	σ quelconque	$O(n \log n)$	2
L	J2	$O(n \log n)$	$2 - \frac{1}{M^{(1)}}$
G	J1	$O(nM^{(1)})$	$3 - \frac{2}{M^{(1)}}$
L'	J2	$O(n \log M^{(1)})$	$2 - \frac{1}{M^{(1)}}$
M'	σ quelconque	$O(n \log n)$	$\emptyset$

Ces algorithmes sont ensuite combinés. La liste des heuristiques testées est la suivante : J1L, J1G, J1L', J1M', J1L'R, J1M'R, J2L, J2G, J2L', J2M', J2L'R, J2M'R (l'algorithme G est présenté figure 6). Les heuristiques J2L'R et J2M'R sont celles qui fournissent les meilleurs résultats. Dans la PSE, la borne inférieure utilisée est la plus grande parmi celles

Algorithme M' : Modified List scheduling for inverse Problem

Entrée : un ordonnancement σ a priori
Etape 1 : Inversion du problème IDENTIQUE À L'ALGORITHME L' (CF. FIGURE 14)
Etape 2 : Répéter l'étape 2 de l'algorithme L' (cf. Figure 14) pour i variant de 1 à n
Etape 3 : *Affectation des travaux au deuxième étage du problème inverse*
  Sélectionner la machine j qui a le plus petit T_j.
  Choisir le travail i qui n'est pas encore ordonnancé sur le deuxième étage tel que $C_i^{'(1)} \leq T_j$ et de plus grand $p_i^{'(2)}$
  Si un tel travail n'existe pas *Alors*
    Sélectionner le travail i non encore ordonnancé sur le $2^{ème}$ étage de plus petit $C_i^{'(1)}$
  FinSi
  Ordonnancer le travail i sur la machine j
  $T_j = \max\{C_i^{'(1)}, T_j\} + p_i^{'(2)}$, $C_i^{'(2)} = T_j$ et $\pi'_j = (\pi'_j, i)$
  Si $i < n$ *Alors*
    $i = i + 1$
    Aller à l'étape 3
  FinSi
Etape 4 : Retour au problème direct, IDENTIQUE À L'ALGORITHME L' (CF. FIGURE 14)

Figure 15. – Algorithme 10 [gup-1995].

données précédemment. Un théorème permet de limiter le nombre de nœuds parcourus.

THÉORÈME 5 : *Pour le problème de flow-shop hybride* $FH2, (PM^{(1)}, 1) \mid\mid C_{max}$ *tout ordonnancement dans lequel le travail* j *précède le travail* i *sur une même machine du premier étage, avec* $p_i^{(1)} \leq p_j^{(1)}$ *et* $p_i^{(2)} \geq p_j^{(2)}$, *est dominé par l'ordonnancement dans lequel* i *précède* j *sur cette machine.*

La figure 16 présente l'algorithme de la PSE proposée dans [gup-1995].

• Vignier, Billaut et Proust se sont intéressés aux problèmes $FH2, ([P, Q, R]M, [P, Q, R]M) \mid DT^{(1,2)}, split \mid C_{max}$ dans [vi-1995]. Ils proposent des théorèmes permettant de trouver un ordonnancement optimal dans des cas particuliers ainsi que des bornes inférieures et supérieures. L'ensemble de leur contribution est résumé ci-dessous.

Ils considérent que lorsqu'une quantité de travail est affectée à la machine j de l'étage 1, alors elle est affectée à la machine j de l'étage 2. On a $M^{(1)} = M^{(2)} = M$. La contrainte d'intégrité des tailles des sous-lots est relaxée. Dans tout ce qui suit, $v_{ij}^{(\ell)}$ représente le temps pour exécuter une unité du travail i sur la machine j à l'étage ℓ, q_{ij} est la quantité du travail i affectée à la machine j, $o_{ij}^{(\ell)}$ est le travail placé à la $i^{ème}$ position sur la machine j de l'étage ℓ et sera noté o_i lorsqu'il est inutile de préciser la machine et l'étage.

Algorithme de la PSE

Un noeud définit une séquence partielle et sa date de fin sur chacune des machines du premier étage. L'ordonnancement final (i.e sur les deux étages) est déterminé en utilisant le théorème 5. Cette définition de noeud rejoint, dans une certaine mesure, la construction de l'arbre de recherche présentée dans [br-1991] (cf. paragraphe 3.2). Cet algorithme est déduit de l'article.

Etape 1 : *Initialisation*
- Calculer une borne supérieure UB initiale, par l'une des heuristiques.
- Racine=Δ
- $\Omega = \{1, 2, ..., n\}$, ensemble des travaux à ordonnancer.
- Noeud-Courant={Racine}
- $L = \Delta$, ensemble des noeuds à explorer.

Etape 2 :
- Créer $M^{(1)}$ fils à partir de Noeud-Courant et tels que $p_{i_1} \leq p_{i_2} \leq ... \leq p_{i_{M^{(1)}}}$
- Les insérer dans L
- *Pour* tous les éléments de L *Faire*
  - calculer la borne inférieure LB_i de l'élément i
  - *Si* $LB_i > UB$ *Alors*
    - Supprimer ce noeud de L
  - *FinSi*
- *Fin Pour*

Etape 3 : *Branchement*
- *Si* L n'est pas vide *Alors*
  - Choisir le noeud N dont la LB est la plus petite parmi les noeuds les plus profonds.
  - Noeud-courant=N
- *Sinon*
  - Editer la solution et FIN
- *FinSi*

Etape 4 : *Séparation*
- $\Omega = \{1, 2, ..., n\}$−Noeud-courant
- *TantQue* $\Omega \neq \emptyset$ *Faire*
  - Générer une branche pour le travail i sur la machine libre le plus tôt
  - Noeud-Courant = Noeud-Courant $\cup\{i\}$
  - *Si* la condition de dominance est violée pour un travail déjà affecté sur cette machine *Alors*
    - Supprimer la branche créée
  - *Sinon*
    - Calculer la borne inférieure de Noeud-Courant
    - *Si* | Noeud-Courant |= n *Alors*
      - Calculer l'ordonnancement sur les deux étages en utilisant le théorème 5
      - LB =valeur du C_{max} obtenue
    - *FinSi*
    - *Si* $LB < UB$ *Alors*
      - Mettre Noeud-Courant dans L
    - *Sinon*
      - Le supprimer
    - *FinSi*
  - *FinSi*
- *FinTantQue*
- Aller à l'étape 3

Figure 16. – Algorithme 11 [gup-1995].

Cas trivial : machines identiques tous les étages confondus : Problème P1

Ce cas revient à considérer que $v_{pj}^{(1)} = v_{pj}^{(2)} = v_p$, $\forall p = 1, ..., n$ et $\forall j = 1, ..., M$.

Le problème s'écrit :

$$\text{Minimiser } C_{max} = \max_{1\leq j\leq M}\left\{\max_{1\leq r\leq n}\left[\sum_{i=1}^{r} v_{o_i}\times q_{o_i j} - \sum_{i=1}^{r-1} v_{o_i}\times q_{o_i j}\right]\right\} + \sum_{i=1}^{n} v_{o_i}\times q_{o_i j}.$$

THÉORÈME 6 : *Un ordonnancement optimal à ce problème est fourni, par exemple, par l'algorithme J de S. M. Johnson appliqué à l'ensemble des travaux i,* $\forall i = 1,...,n$, *avec* $p_{i1} = v_i \times \frac{Q_i}{M}$ *et* $p_{i2} = v_i \times \frac{Q_i}{M}$. *Cette solution est dupliquée sur tous les couples* $(M_j^{(1)}, M_j^{(2)})$, $\forall j = 1,...,M$.

De nombreuses solutions équivalentes, relativement à C_{max}, peuvent être fournies par J (selon sa programmation) mais elles ne sont pas équivalentes quant à leurs autres qualités.

Le cas encore plus trivial où v_p =contante, $\forall p = 1,...,n$ n'a pas été traité.

Cas des machines identiques par étage : Problème P2

On a maintenant $v_{pj}^{(1)} = v_p^{(1)}$, $v_{pj}^{(2)} = v_p^{(2)}$, $\forall p = 1,...,n$ et $\forall j = 1,...,M$.

Le problème s'écrit :

$$\text{Minimiser } C_{max} = \max_{1\leq j\leq M}\left\{\max_{1\leq r\leq n}\left[\sum_{i=1}^{r} v_{o_i}^{(1)}\times q_{o_i j}^{(1)} - \sum_{i=1}^{r-1} v_{o_i}^{(2)}\times q_{o_i j}^{(2)}\right] + \sum_{i=1}^{n} v_{o_i}^{(2)}\times q_{o_i j}^{(2)}\right\}$$

sous

$$\sum_{j=1}^{M} q_{pj}^{(1)} = Q_p \quad \forall p = 1,...,n$$

$$\sum_{j=1}^{M} q_{pj}^{(2)} = Q_p \quad \forall p = 1,...,n$$

$$q_{pj}^{(1)} = q_{pj}^{(2)} = q_{pj} \quad \forall p = 1,...,n.$$

THÉORÈME 7 : *Dans ce cas, la solution optimale est obtenue en prenant* $q_{ij}^{(1)} = q_{ij}^{(2)} = \frac{Q_i}{M}$, $\forall j = 1,...,M$, *et en appliquant l'algorithme J sur un ensemble de travaux i dont les temps d'exécution sont* $p_{i1} = v_i^{(1)} \times \frac{Q_i}{M}$ *et* $p_{i2} = v_i^{(2)} \times \frac{Q_i}{M}$, $\forall i = 1,...,n$. *Les M problèmes sont identiques.*

Cas des machines quelconques au premier étage, identiques au second étage : Problème P3

Avec $v_{pj}^{(1)}$ quelconque et $v_{pj}^{(2)} = v_p^{(2)}$, $\forall j = 1, ..., M$ et $\forall p = 1, ..., n$. On impose que $o_{ij}^{(1)} = o_{ij}^{(2)} = o_{ij}$, $\forall i = 1, ..., n$ et $\forall j = 1, ..., M$. L'ordre des travaux est le même sur les deux étages.

Le problème s'écrit :

$$\text{Minimiser } C_{max} = \max_{1 \le j \le M} \left\{ \max_{1 \le r \le n} \left[\sum_{i=1}^{r} v_{o_{ij}j}^{(1)} \times q_{o_{ij}j}^{(1)} - \sum_{i=1}^{r-1} v_{o_{ij}j}^{(2)} \times q_{o_{ij}j}^{(2)} \right] + \sum_{i=1}^{n} v_{o_{ij}j}^{(2)} \times q_{o_{ij}j}^{(2)} \right\}$$

sous

$$\sum_{j=1}^{M} q_{pj}^{(1)} = Q_p \quad \forall p = 1, ..., n$$

$$\sum_{j=1}^{M} q_{pj}^{(2)} = Q_p \quad \forall p = 1, ..., n$$

$$q_{pj}^{(1)} = q_{pj}^{(2)} = q_{pj} \quad \forall p = 1, ..., n \text{ et } \forall j = 1, ..., M.$$

Soit X_{kij} une variable bivalente qui vaut 1 si le travail k est placé en i^{ème} position sur la machine j (quelque soit l'étage), et 0 sinon.

THÉORÈME 8 : *Le problème se résout par : Minimiser Z, avec Z sans restriction de signe*

sous

$$Z \ge \max_{1 \le j \le M} \left\{ \max_{1 \le r \le n} \left[\sum_{i=1}^{r} \sum_{k=1}^{n} X_{kij} \times v_{kj}^{(1)} \times q_{kj} - \sum_{i=1}^{r-1} \sum_{k=1}^{n} X_{kij} \times v_k^{(2)} \times q_{kj} \right] + \sum_{i=1}^{n} v_i^{(2)} \times q_{ij} \right\}$$

$$\sum_{j=1}^{M} q_{pj} = Q_p \quad \forall p = 1, ..., n$$

$$\sum_{i=1}^{n} X_{kij} = 1 \quad \forall k = 1, ..., n \text{ et } \forall j = 1, ..., M$$

$$\sum_{k=1}^{n} X_{kij} = 1 \quad \forall i = 1, ..., n \text{ et } \forall j = 1, ..., M$$

qui est un problème, a priori non linéaire, mixte (les variables sont bivalentes).

Cette modélisation a été mise en œuvre avec AMPL [vi-1995].

THÉORÈME 9 : *Dans ce cas, prenons* $q_{ij}^{(1)} = q_{ij}^{(2)} = q_{ij} = \frac{Q_i}{v_{ij}^{(1)} \times \Upsilon_i}$, *avec* $\Upsilon_i = \sum_{j=1}^{M} \frac{1}{v_{ij}^{(1)}}$. *On considère* M *problèmes* $(M_j^{(1)}, M_j^{(2)})$ *de type flow-shop à 2 machines avec* $p(j)_i^{(1)} = \frac{Q_i}{\Upsilon_i}$ *et* $p(j)_i^{(2)} = \frac{v_i^{(2)}}{v_{ij}^{(1)}} \times \frac{Q_i}{\Upsilon_i}$ *qu'on résout chacun par l'algorithme J. La solution trouvée est une borne supérieure à notre problème. Sa durée d'exécution totale,* C_{max}, *est donnée par* $C_{max} = \max_{1 \leq j \leq M}\{C(j)_{max}\}$.

Exemple : Soit un flow-shop hybride à 2 étages, avec 3 machines par étage. Les données sont fournies dans le tableau 7. En appliquant J sur les trois sous-problèmes, on obtient les résultats présentés tableau 8. Le diagramme de Gantt correspondant est présenté figure 17.

TABLEAU 7
Données de l'exemple [vi-1995].

Travail	Q_i	$v_{i1}^{(1)}$	$v_{i2}^{(1)}$	$v_{i3}^{(1)}$	$v_i^{(2)}$	Υ_i	$\frac{Q_i}{\Upsilon_i}$	q_{i1}	q_{i2}	q_{i3}
J1	2	2	4	1	2	$\frac{7}{4}$	$\frac{8}{7}$	$\frac{4}{7}$	$\frac{2}{7}$	$\frac{8}{7}$
J2	5	3	5	10	1	$\frac{19}{30}$	$\frac{150}{19}$	$\frac{50}{19}$	$\frac{30}{19}$	$\frac{15}{19}$
J3	6	2	1	2	4	2	3	$\frac{3}{2}$	3	$\frac{3}{2}$
J4	3	1	3	1	1	$\frac{7}{3}$	$\frac{9}{7}$	$\frac{9}{7}$	$\frac{3}{7}$	$\frac{9}{7}$

Si la quantité du travail J4 affectée à la machine 2 du premier étage est diminuée de $\frac{3}{7}$, alors on obtient le diagramme de Gantt présenté figure 18.

THÉORÈME 10 : *La solution obtenue par le théorème 7 est une borne inférieure au problème P3, en prenant* $v_i^{(1)} = \min_{1 \leq j \leq M}(v_{ij}^{(1)})$, $\forall i = 1, ..., n$.

TABLEAU 8
Résultats de l'exemple [vi-1995].

	1er sous-problème		2ème sous-problème		3ème sous-problème	
Travail	$p(1)_i^{(1)}$	$p(1)_i^{(2)}$	$p(2)_i^{(1)}$	$p(2)_i^{(2)}$	$p(3)_i^{(1)}$	$p(3)_i^{(2)}$
J1	$\frac{8}{7}$	$\frac{8}{7}$	$\frac{8}{7}$	$\frac{4}{7}$	$\frac{8}{7}$	$\frac{16}{7}$
J2	$\frac{150}{19}$	$\frac{50}{19}$	$\frac{150}{19}$	$\frac{30}{19}$	$\frac{150}{19}$	$\frac{15}{19}$
J3	3	6	3	12	3	6
J4	$\frac{9}{7}$	$\frac{9}{7}$	$\frac{9}{7}$	$\frac{3}{7}$	$\frac{9}{7}$	$\frac{9}{7}$
Résultat :	$(J1, J4, J3, J2)$		$(J3, J2, J1, J4)$		$(J1, J4, J3, J2)$	

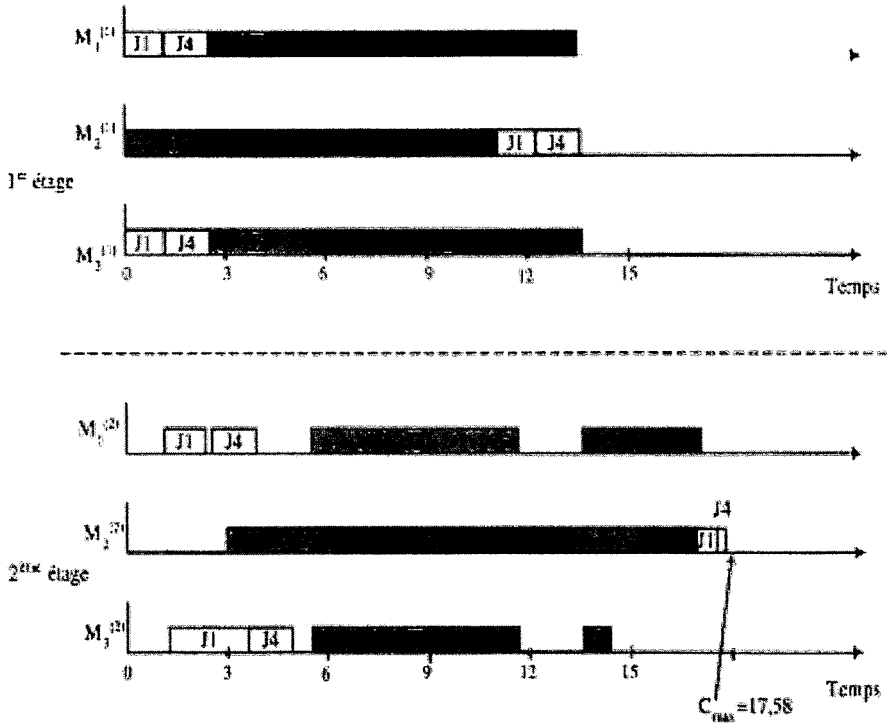

Figure 17. – Résultat obtenu en appliquant le théorème 9.

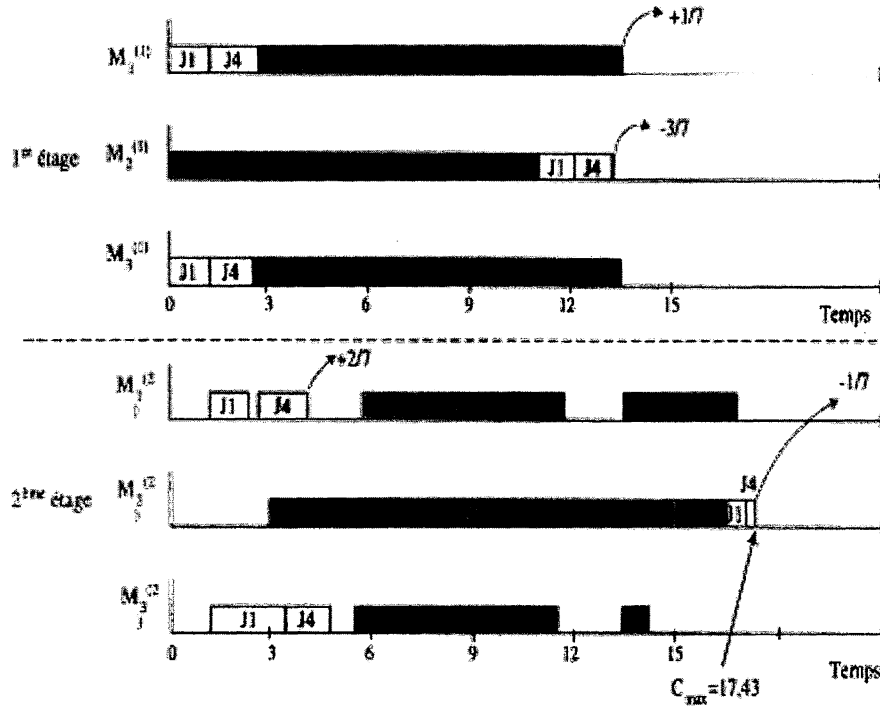

Figure 18. – Diagramme de Gantt quand les quantités du travail J4 sont modifiées.

Cas où les machines $M_j^{(1)}$ et $M_j^{(2)}$ sont proportionnelles : Problème P4

On a $\frac{v_{ij}^{(1)}}{v_{ij}^{(2)}} = k_i = constante$, $\forall i = 1, ..., n$, $\forall j = 1, ..., M$, donc :

$$k_i = \frac{\sum_{j=1}^{M} \frac{1}{v_{ij}^{(2)}}}{\sum_{j=1}^{M} \frac{1}{v_{ij}^{(1)}}} = \frac{\Upsilon_i^{(2)}}{\Upsilon_i^{(1)}} \text{ où } \Upsilon_i^{(\ell)} = \sum_{j=1}^{M} \frac{1}{v_{ij}^{(\ell)}} \forall \ell = 1..2.$$

Le problème s'écrit :

$$\text{Minimiser } C_{max} = \max_{1 \le j \le M} \left\{ \max_{1 \le r \le n} \left[\sum_{i=1}^{r} v_{o_i j}^{(1)} \times q_{o_i j}^{(1)} - \sum_{i=1}^{r-1} v_{o_i j}^{(2)} \times q_{o_i j}^{(2)} \right] + \sum_{i=1}^{n} v_{o_i j}^{(2)} \times q_{o_i j}^{(2)} \right\}$$

sous

$$\sum_{j=1}^{M} q_{pj}^{(1)} = Q_p \quad \forall p = 1, ..., n$$

$$\sum_{j=1}^{M} q_{pj}^{(2)} = Q_p \quad \forall p = 1, ..., n$$

$$q_{pj}^{(1)} = q_{pj}^{(2)} = q_{pj} \quad \forall p = 1, ..., n \text{ et } \forall j = 1, ..., M.$$

THÉORÈME 11 : *Dans ce cas, la solution optimale est obtenue en prenant* $q_{ij}^{(1)} = \frac{Q_i}{\sum_{h=1}^{M} \frac{v_{ij}^{(1)}}{v_{ih}^{(1)}}}$ *et* $q_{ij}^{(2)} = \frac{Q_i}{\sum_{h=1}^{M} \frac{v_{ij}^{(2)}}{v_{ih}^{(2)}}}$, $\forall j = 1, ..., M$ *et en appliquant l'algorithme J sur un ensemble de travaux* i *dont les temps d'exécution sont* $p_{i1} = \frac{Q_i}{\sum_{h=1}^{M} \frac{1}{v_{ih}^{(1)}}} = \frac{Q_i}{\Upsilon_i^{(1)}}$ *et* $p_{i2} = \frac{Q_i}{\sum_{h=1}^{M} \frac{1}{v_{ih}^{(2)}}} = \frac{Q_i}{\Upsilon_i^{(2)}}$, $\forall i = 1, ..., n$. *Les* M *problèmes sont identiques.*

Cas de machines quelconques aux deux étages : Problème P5

On impose $o_{ij}^{(1)} = o_{ij}^{(2)} = o_{ij}$, $\forall i = 1, ..., n$ et $\forall j = 1, ..., M$. On notera que ceci ne veut pas dire que le travail placé en i^{ème} position sur $(M_j^{(1)}, M_j^{(2)})$ se trouvera forcément à cette même position sur $(M_{j'}^{(1)}, M_{j'}^{(2)})$.

Sa modélisation analytique est :

$$\text{Minimiser } C_{max} = \max_{1 \le j \le M} \left\{ \max_{1 \le r \le n} \left[\sum_{i=1}^{r} v_{o_{ij}j}^{(1)} \times q_{o_{ij}j}^{(1)} - \sum_{i=1}^{r-1} v_{o_{ij}j}^{(2)} \times q_{o_{ij}j}^{(2)} \right] + \sum_{i=1}^{n} v_{o_{ij}j}^{(2)} \times q_{o_{ij}j}^{(2)} \right\}$$

sous

$$\sum_{j=1}^{M} q_{pj}^{(1)} = Q_p \quad \forall p = 1, ..., n$$

$$\sum_{j=1}^{M} q_{pj}^{(2)} = Q_p \quad \forall p = 1, ..., n$$

$$q_{pj}^{(1)} = q_{pj}^{(2)} = q_{pj} \quad \forall p = 1, ..., n \text{ et } \forall j = 1, ..., M.$$

Soit X_{kij} une variable bivalente qui vaut 1 si le travail k est placé en i^{ème} position sur la machine j des deux étages et 0 sinon.

THÉORÈME 12 : *Ce problème se résout par :*
Minimiser Z, avec Z sans restriction de signe
sous

$$Z \geq \max_{1\leq j\leq M}\left\{ \max_{1\leq r\leq n}\left[\sum_{i=1}^{r}\sum_{k=1}^{n} X_{kij}\times v_{kj}^{(1)}\times q_{kj} - \sum_{i=1}^{r-1}\sum_{k=1}^{n} X_{kij}\times v_{kj}^{(2)}\times q_{kj}\right] + \sum_{k=1}^{n} v_{kj}^{(2)}\times q_{kj}\right\}$$

$$\sum_{j=1}^{M} q_{kj} = Q_k \quad \forall k = 1,...,n$$

$$\sum_{i=1}^{n} X_{kij} = 1 \quad \forall k = 1,...,n \ \ et \ \ \forall j = 1,...,M$$

$$\sum_{k=1}^{n} X_{kij} = 1 \quad \forall i = 1,...,n \ \ et \ \ \forall j = 1,...,M.$$

Cette modélisation a été mise en œuvre avec AMPL [vi-1995].

THÉORÈME 13 : *Dans ce type de problème, si on prend* $q_{ij}^{(1)} = q_{ij}^{(2)} = \frac{Q_i}{v_{ij}^{(1)}\times\Upsilon_i^{(1)}}$ *où* $\Upsilon_i^{(1)} = \sum_{j=1}^{M}\frac{1}{v_{ij}^{(1)}}$ *et si on considère M problèmes* $(M_j^{(1)}, M_j^{(2)})$ *de type flow-shop à deux machines avec* $p(j)_i^{(1)} = \frac{Q_i}{\Upsilon_i^{(1)}}$ *et* $p(j)_i^{(2)} = \frac{v_{ij}^{(2)}}{v_{ij}^{(1)}}\times\frac{Q_i}{\Upsilon_i^{(1)}}$, *qu'on résout chacun par l'algorithme J, alors la solution trouvée est une solution approchée du* $FH2,((RM)^2) \mid DT^{(1,2)}, split \mid C_{max}$. *Sa durée d'exécution totale est* $C_{max} = \max_{1\leq j\leq M}\{C(j)_{max}\}$.

Une autre solution approchée peut être obtenue en prenant $q_{ij}^{(1)} = q_{ij}^{(2)} = \frac{Q_i}{v_{ij}^{(2)}\times\Upsilon_i^{(2)}}$ et en appliquant de la même façon J sur $p(j)_i^{(1)} = \frac{v_{ij}^{(1)}}{v_{ij}^{(2)}}\times\frac{Q_i}{\Upsilon_i^{(2)}}$ et $p(j)_i^{(2)} = \frac{Q_i}{\Upsilon_i^{(2)}}$.

Les problèmes de type $FH2,([P,Q,R]M,[P,Q,R]M) \mid S_{nsd}, R_{nsd}, DT^{(1,2)}, split \mid C_{max}$ et $FH2,([P,Q,R]M,[P,Q,R]M) \mid d_i, S_{nsd}, R_{nsd}, DT^{(1,2)}, split \mid T_{max}$ restent à étudier.

• Dans [ch-1995], une étude permet de comparer explicitement les heuristiques présentées dans [gu-1988], [bu-1973], [sr-1989] et un algorithme de liste dont la liste initiale est obtenue de façon arbitraire, pour le problème

$FH2,(PM^{(1)},1) \mid\mid C_{max}$. Il est montré que le ratio de performance dans le pire des cas pour l'algorithme de liste et de l'heuristique de [bu-1973] ne peut jamais dépasser 2. Pour les deux autres, le ratio dans le pire des cas est compris entre $2-\frac{1}{M^{(1)}}$ et $3-\frac{2}{M^{(1)}}$. En revanche de façon expérimentale, les performances en moyenne sont meilleures que dans le pire des cas. L'algorithme de liste a un temps de calcul plus rapide et un ratio dans le pire des cas meilleur que [gu-1988] et [sr-1989] mais en moyenne plus mauvais.

• Li s'intéresse au $FH2,(1,PM^{(2)}) \mid S_{nsd}, s_{nsd}, split^{(2)}$, recouvrement $\mid C_{max}$ [li-1996] : il existe des temps de réglage indépendants des séquences, pour la mise en route d'une nouvelle famille f de produits i ($S_f^{(\ell)}$), $i \in J_f$ et, à l'intérieur d'une famille, pour un nouveau produit i ($s_{fi}^{(\ell)}$). Il donne la valeur optimale du C_{max} lorsque le nombre de machines au deuxième étage est illimité et que deux conditions de dominance – raisonnables – existent sur les temps de réglage et sur les temps unitaires de fabrication. L'auteur fournit deux heuristiques : FORWARD et BACKWARD (*cf.* figure 19) qui tendent en fait à réduire les temps de réglage. Elles nécessitent un séquencement initial donné par une des six règles : SPT1, SPT2, LPT2, AVG1, AVG2 ou AVG3. AVG1 est la meilleure règle de pré-classement : les familles sont classées par $\frac{S_f^{(2)}+\sum_{i\in J_f}(s_{fi}^{(2)}+v_{fi}^{(2)}\times Q_{fi})}{|J_f|}$ décroissant (temps moyen de production d'un lot par famille) et, à l'intérieur d'une famille f par $s_{fi}^{(2)}+v_{fi}^{(2)}\times Q_{fi}$ décroissant. Les facteurs considérés dans les expérimentations sont le nombre de machines à l'étage 2, le nombre de familles, le nombre de lots par famille, la taille des lots, les temps de réglage majeurs à l'étage 1, les temps de réglage majeurs à l'étage 2 et les temps de réglage mineurs aux deux étages. En conclusion, BACKWARD est plus efficace.

Lorsque la fabrication des lots ne peut plus être éclatée et que le recouvrement entre les deux étages n'est plus autorisé, l'auteur mentionne les adaptations des deux heuristiques précédentes. Il donne ensuite une modélisation analytique d'une borne inférieure classique en ne considérant que le deuxième étage. Celle-ci se révèle efficace si la taille des lots est relativement faible et si les rapports $\frac{v_{fi}^{(2)}}{v_{fi}^{(1)}}$ sont relativement grands (la production au premier étage a peu de poids...).

• Dans [vi-1996b], les auteurs proposent une modélisation analytique et deux heuristiques pour résoudre le $FH2,(PM^{(1)},1) \mid pmtn^{(1)}, d_i^{(2)} \mid T_{max}$. La première, H1, résout le problème en deux étapes. Dans un premier

Algorithme de BACKWARD

Etape 1 :

Soit un ordonnancement initial des familles et des lots, sur la machine du premier étage, obtenu par une des règles de séquencement initial. Soit C_{max} une borne supérieure à la durée d'achèvement total et soit $\Delta = \frac{x \times C_{max}}{100}$ (2 % selon l'auteur). Soit $CM_j^{(2)} = C_{max}$, $\forall j = 1, ..., M^{(2)}$, la date initiale de disponibilité de chacune des machines de l'étage 2

Etape 2 : *Ordonnancement des lots et des familles à l'étage 2*

Prendre le lot i de la famille f disponible le plus tard à l'étage 1, et non encore placé à l'étage 2

Soit Ω l'ensemble des machines qui travaillent déjà des produits du même type et qui peuvent encore en faire d'autres (i.e prévoir de rajouter en amont, les temps de montage mineur et majeur, ultérieurement, lorsqu'on décidera d'y arrêter cette fabrication)

Si Ω est vide *Alors*

Soit Ω l'ensemble des machines qui travaillent des produits de la même famille et qui peuvent encore en faire d'autres (i.e prévoir de rajouter en amont, les temps de montage mineur et majeur, ultérieurement, lorsqu'on décidera d'y arrêter cette fabrication)

FinSi

Si $|\Omega| = 1$ *Alors*

Assigner le lot à cette machine, notée m^*

Sinon

Choisir celle qui est utilisée le plus tard

FinSi

Si $CM_{m^*}^{(2)} - v_{fi}^{(2)} \times Q_{fi} \geq C_{fi}^{(1)}$ et si on peut positionner, devant, les temps de réglage complet *Alors*

Ranger le lot complet

$C_{m^*}^{(2)} = C_{m^*}^{(2)} - v_{fi}^{(2)} \times Q_{fi} - s_{fi}^{(2)}[-S_f^{(2)}]^{Note}$

Aller à l'étape 3

Sinon

Positionner le maximum possible de produits sur m^* c'est-à-dire calculer x_{fi} avec $x_{fi} \leq Q_{fi}$ et $CM_{m^*}^{(2)} - v_{fi}^{(2)} \times x_{fi} \geq C_{fi}^{(1)}$ et à condition qu'on puisse placer les temps de réglages nécessaires, devant, ultérieurement

$Q_{fi} = Q_{fi} - x_{fi}$, Mise à jour de $C_{fi}^{(1)}$

$C_{m^*}^{(2)} = C_{m^*}^{(2)} - v_{fi}^{(2)} \times x_{fi}[-s_{fi}^{(2)}[-S_f^{(2)}]]^{Note}$

Aller à l'étape 2

FinSi

Etape 3 :

Si tous les lots de toutes les familles sont placés ou si la séquence est infaisable *Alors*

Aller à l'étape 4

Sinon

Aller à l'étape 2

FinSi

Etape 4 :

Si plusieurs essais ont été faits (une vingtaine selon l'auteur) *Alors*

Donner le dernier ordre retenu

FinSi

Si l'ordre est possible *Alors*

$C_{max} = C_{max} - \Delta$ et $CM_j^{(2)} = C_{max}, \forall j = 1, ..., M^{(2)}$ et recalcul de Δ

Aller à l'étape 2

FinSi

Si l'ordre est impossible *Alors*

$C_{max} = C_{max} + \Delta$ et $CM_j^{(2)} = C_{max}, \forall j = 1, ..., M^{(2)}$ et recalcul de Δ

Aller à l'étape 2

FinSi

Note : *En fait, il faudra garder mémoire des différentes valeurs avec ou sans les temps de réglage. Cela dépendra de ce que l'on mettra comme autre sous-lot, devant, à l'itération suivante*

Figure 19. – Algorithme 12 [li-1996].

temps, les travaux sont classés selon EDD au deuxième étage, déterminant ainsi des dates dues au premier étage. Ensuite, l'écart maximum (L_{max}) au premier étage est minimisé. Dans un deuxième temps le retard maximum est minimisé sur la machine du deuxième étage (en considérant des dates de début au plus tôt). La deuxième, H2, se décompose en trois étapes. La première détermine une séquence en appliquant la règle EDD (les durées d'exécution au premier étage ne sont pas prises en compte). La séquence ainsi obtenue définit donc les dates de début d'exécution des travaux au deuxième étage à partir de la date 0. La deuxième étape consiste à calculer les décalages minimaux qu'il faut leur ajouter pour permettre aux travaux de s'exécuter au premier étage. Le calcul de ces décalages repose sur un théorème, établi par les auteurs, et basé sur les conditions de satisfiabilité de Horn [ho-1974]. Enfin, dans la troisième étape les algorithmes de Horn et de Mac Naughton [na-1959] sont appliqués afin d'obtenir l'affectation des travaux au premier étage.

Dans ce même papier, le problème $FH2, (PM^{(1)}(t), 1) \mid pmtn^{(1)}, d_i^{(2)} \mid T_{max}$ est aussi abordé. Une modélisation analytique et une méthode de résolution sont proposées. Cette heuristique est similaire à H2, en prenant en compte les calendriers de fonctionnement des machines.

• Dans [gu-1996], les auteurs donnent d'abord une modélisation en programmation linéaire entière à variables 0-1 pour le $FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid\mid C_{max}$. Mais ils proposent des heuristiques de résolution pour le problème à 2 étages.

$$
\begin{cases}
MinZ = C_{max} & \\
\text{sous} & \\
\sum_{\substack{i=0 \\ i\neq j}}^{n} \sum_{p=1}^{M^{(\ell)}} x(i,j,p,\ell) = 1 & \forall j \in [1,n],\ \ell \in [1,k] \\
\sum_{j=0}^{n} x(i,j,p,\ell) \leq 1 & \forall i \in [0,n],\ p \in [1, M^{(\ell)}],\ \ell \in [1,k] \\
\sum_{\substack{i=0 \\ i\neq h}}^{n} x(i,h,p,\ell) - \sum_{\substack{j=0 \\ j\neq h}}^{n} x(h,j,p,\ell) = 0 & \forall h \in [1,n],\ p \in [1, M^{(\ell)}],\ \ell \in [1,k] \\
C_j^{(\ell)} \geq C_i^{(\ell)} + \sum_{p=1}^{M^{(\ell)}} x(i,j,p,\ell) \times p_j^{(\ell)} & \\
+ \left\{ \left(\sum_{p=1}^{M^{(\ell)}} x(i,j,p,\ell) \right) - 1 \right\} \times HV, & \forall i \in [0,n],\ j \in [1,n], \ell \in [1,k] \\
C_j^{(\ell)} \geq C_j^{(\ell-1)} + p_j^{(\ell)} & \forall j \in [1,n],\ \ell \in [2,k] \\
C_j^{(\ell)} \leq C_{max} & \forall j \in [1,n],\ \ell \in [1,k] \\
C_j^{(\ell)} \geq 0 & \forall j \in [1,n],\ \ell \in [1,k]
\end{cases}
$$

avec $x(i,j,p,\ell) = 1$ si le travail j est ordonnancé immédiatement après le travail i sur la machine p de l'étage ℓ, $x(0,j,p,\ell) = 1$ si le travail j est le premier à s'exécuter sur la machine p de l'étage ℓ, $x(i,0,p,\ell) = 1$ si le travail i est le dernier à s'exécuter sur la machine p de l'étage ℓ, $C_i^{(\ell)}$ la date de fin d'exécution du travail i à l'étage ℓ. HV est une très grande valeur.

Un exemple de résolution avec le logiciel AMPL est fourni en annexe de [vi-1995].

Trois bornes inférieures sont proposées pour le cas particulier $k = 2$.

LB_1 est égale au C_{max} obtenu en appliquant l'algorithme J de S. M. Johnson sur $p_i^{(1)} = \frac{p_i^{(1)}}{\max\{M^{(1)},M^{(2)}\}}$ et $p_i^{(2)} = \frac{p_i^{(2)}}{\max\{M^{(1)},M^{(2)}\}}$.

$$LB_2 = \min_{1\leq i\leq n} p_i^{(1)} + \max\left\{ \frac{1}{M^{(2)}} \sum_{i=1}^{n} p_i^{(2)}, max_{1\leq i\leq n} p_i^{(2)} \right\}$$

LB_3, valeur duale de LB2 obtenue sur le problème inverse, est égale à

$$\max\left\{ \frac{1}{M^{(1)}} \sum_{i=1}^{n} p_i^{(1)}, \max_{1\leq i\leq n} p_i^{(1)} \right\} + \min_{1\leq i\leq n} p_i^{(2)}.$$

Dans un troisième temps, ils proposent des heuristiques pour le problème $FH2, ((PM^{(\ell)})_{\ell=1}^{2}) \mid\mid C_{max}$, dont la structure est présentée figure 20. Une première phase détermine une séquence, la deuxième affecte les travaux, étage par étage, dans l'ordre de celle-ci. Plusieurs règles de séquencement sont proposées :

1. la règle de S. M. Johnson (le travail i est ordonnancé devant le travail j si $\min(p_i^{(1)}, p_j^{(2)}) \leq \min(p_j^{(1)}, p_i^{(2)})$),
2. la règle SPT appliquée aux $p_i^{(1)}$,
3. la règle SPT appliquée aux $p_i^{(1)} + p_i^{(2)}$,
4. la règle SPT appliquée aux $p_i^{(2)}$,
5. la règle LPT, en trois versions similaires à celles ci-dessus.

Les ordonnancements obtenus ne devraient pas être sans délai car on place les travaux un par un.

Comme le font remarquer les auteurs, la règle de Johnson semble intéressante pour ce problème puisque les travaux ayant une durée d'exécution plus faible au premier étage sont placés en début d'ordonnancement et ceux ayant une durée d'exécution plus grande au deuxième étage sont placés en fin d'ordonnancement. Or l'utilisation de la règle SPT permet de minimiser la moyenne des dates de fin d'exécution des travaux au premier étage et comme nous avons pu le remarquer lors

Structure des heuristiques de Guinet et al

Etape 1 : *Détermination de l'ordonnancement*
  Appliquer une des règles de séquencement
Etape 2 : *Réalisation de l'affectation des travaux*
  Pour $j = 1$ à n *Faire*
    Soient $CM^{(1)}$ *et* $CM^{(2)}$ *la plus petite date de disponibilité des machines du premier et du deuxième étage*
    Si $CM^{(1)} + p_j^{(1)} \geq CM^{(2)}$ *Alors*
      Le travail j est affecté sur la machine qui détermine $CM^{(1)}$.
      Au deuxième étage, il est affecté sur la machine qui minimise sa date d'achèvement. En cas de choix possible, prendre celle qui est libre le plus tard.
    Sinon
      Le travail j est affecté sur la machine du deuxième étage qui détermine $CM^{(2)}$.
      Au première étage, il est affecté sur la machine qui minimise son temps d'attente.
    FinSi
    Mettre à jour $CM^{(1)}$ et $CM^{(2)}$.
  FinPour

Figure 20. – Algorithme 13 [gu-1996].

de l'étude de [sr-1989], LPT apporte une bonne solution au problème de la minimisation de la durée totale d'achèvement pour le $PM \mid\mid C_{max}$ au deuxième étage.

3.2. Les problèmes à k étages

• Salvador [sa-1973] présente une PSE pour la résolution du $FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid no - wait \mid C_{max}$. Une première phase calcule pour une séquence fixée, les dates d'initialisation des travaux qui minimisent le C_{max}. Ces dates sont déterminées à l'aide d'un graphe orienté et par une méthode de programmation dynamique. Puis, la PSE détermine la séquence de permutation optimale pour le critère considéré. Enfin une dernière phase permet de déterminer un ordonnancement optimal à partir des résultats des deux premières.

• Wittrock [wi-1985] et [wi-1988] traite d'un problème de flow-shop hybride dans lequel les travaux peuvent ne pas passer par tous les étages. Le temps nécessaire pour réaliser un travail dépend de son type et de la machine. Les machines sont toutes identiques à chaque étage et il existe des stocks inter-étages limités de capacité $b^{(\ell)(\ell+1)}$ – ces files caractérisent des stocks gérés en FIFO –. Le problème peut donc se noter $FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid b^{(\ell)(\ell+1)} \mid \max(rendement)$ et $\min(encours)$. Dans [wi-1985], l'auteur s'intéresse à un problème d'ordonnancement de type périodique. Le problème général est décomposé en trois sous problèmes : un problème d'affectation des opérations aux machines, un problème de

séquencement sur chaque machine et un problème de détermination de la date à laquelle les pièces apparaissent dans les files d'attente. Le problème d'affectation a pour but de minimiser à chaque étage, le chargement de la machine goulet. Les problèmes de séquencement et de détermination des dates d'arrivée dans les stocks ont pour but d'équilibrer le taux de chargement des machines y compris des stocks (sans modifier l'affectation réalisée) en utilisant une heuristique qui modifie de manière dynamique le taux de chargement. Dans [wi-1988], l'auteur modélise de la même façon un problème d'ordonnancement non périodique.

- Une PSE est proposée dans [br-1991] pour le problème FHk, $((PM^{(\ell)})_{\ell=1}^{k}) \mid\mid C_{max}$. C'est une adaptation de celle proposée dans [br-1975] qui fait référence pour ce problème.

L'arbre de recherche est constitué de $n \times k$ niveaux (n pour chaque étage) et de deux types de nœuds : des nœuds ronds et des nœuds carrés. Un nœud correspond à un travail. En parcourant l'arbre à partir de la racine :

- lorsqu'on rencontre un nœud rond, cela signifie que le travail correspondant est placé en séquence sur la machine courante,
- lorsqu'on rencontre un nœud carré, cela signifie que le travail correspondant est placé en tête sur la machine suivante (éventuellement sur la première machine de l'étage suivant). Si on reste au même étage, cette machine est identique à la précédente.

La position du nœud dans l'arbre détermine donc l'affectation du travail et sa position en séquence sur la machine. Un ensemble de règles est mis au point pour la génération des nœuds :

- Il y a au maximum $n - M^{(\ell)} + 1$ nœuds carrés au premier niveau de l'arbre à chaque étage ℓ.
- Si pour l'étage ℓ, un nœud carré ou rond i est apparu à un niveau, il ne pourra être repris.
- Un nœud carré i ne peut être mis au niveau j si ce niveau contient déjà des nœuds carrés r où $r > i$ (ce sont des possibilités qui ont déjà été testées).
- Pour un étage ℓ donné, il ne peut pas y avoir plus de $M^{(\ell)}$ nœuds carrés (on ne peut pas utiliser plus de machines qu'il n'y en a).
- Aucun chemin ne se terminera s'il contient moins de $M^{(\ell)}$ nœuds carrés sauf si $n < M^{(\ell)}$.

Brah et Hunsucker définissent une borne inférieure composée à la fois d'une borne basée sur les travaux et d'une borne basée sur les machines. Le calcul

de ces bornes est obtenu en généralisant les raisonnements habituels sur le problème du flow-shop. Soient $ACT(S^{(\ell)}(A'))$ et $MCT(S^{(\ell)}(A'))$ définies par les relations 3.2.1 et 3.2.2. $LBM(S^{(\ell)}(A'))$ est égal à $ACT(S^{(\ell)}(A')) + \min_{i \in N-A'} \sum_{\ell'=\ell+1}^{k} p_i^{(\ell')}$ si $ACT(S^{(\ell)}(A')) \geq MCT(S^{(\ell)}(A'))$, et sinon à $MCT(S^{(\ell)}(A')) + \min_{i \in A'} \sum_{\ell'=\ell+1}^{k} p_i^{(\ell')}$. C' est la borne inférieure calculée sur les machines où N est l'ensemble des travaux à ordonnancer. A est un ensemble de travaux déjà ordonnancés sur l'étage ℓ, $A' = A \cup \{i\}$, $S^{(\ell)}(A)$ est l'ordonnancement de l'ensemble A sur l'étage ℓ, $S^{(\ell)}(A')$ représente un ordonnancement formé par la concaténation du travail i à $S^{(\ell)}(A)$, $C[S^{(\ell)}(A), j]$ représente la date de fin de la séquence sur la machine j de l'étage ℓ et $C[S^{(\ell)}(A), j]_{i \in j_1}$ représente la date de fin de la séquence sur la machine j de l'étage ℓ lorsque le travail i est affecté à la machine j_1 de l'étage ℓ. L'algorithme de la PSE est donné figure 21.

$$C[S^{(\ell)}(A'), j]_{i \in j} = \max \left\{ C[S^{(\ell)}(A), j], C[S^{(\ell-1)}(A'), j_1]_{i \in j_1} \right\} + p_i^{(\ell)}$$

ou

$$C[S^{(\ell)}(A'), j]_{i \in j' \neq j} = C[S^{(\ell)}(A), j]$$

$$ACT(S^{(\ell)}(A')) = \frac{\sum_{j=1}^{M^{(\ell)}} C[S^{(\ell)}(A'), j]}{M^{(\ell)}} + \frac{\sum_{i \in N-A'} p_i^{(\ell)}}{M^{(\ell)}} \quad \text{(Average Completion Time)} \quad (3.2.1)$$

$$MCT(S^{(\ell)}(A')) = \max_{1 \leq j \leq M^{(\ell)}} C[S^{(\ell)}(A'), j] \quad \text{(Maximum Completion Time)} \quad (3.2.2)$$

$LBJ(S^{(\ell)}(A')) = \min_{1 \leq j \leq M^{(\ell)}} C[S^{(\ell)}(A'), j] + \max_{i \in N-A'} \sum_{\ell'=\ell}^{k} p_i^{(\ell')}$ est la borne calculée sur les travaux.

Exemple : Considérons un flow-shop hybride à deux étages avec $M^{(1)} = 2$ et $M^{(2)} = 1$. Les durées d'exécution sont données dans le tableau 9. L'arbre généré est celui présenté figure 22. Le diagramme de Gantt associé est présenté figure 23.

PSE de Brah et Hunsucker

Etape 1 :
  $C_{max} = \infty$
Etape 2 :
  Générer $n - M^{(1)} + 1$ noeuds carrés.
  Calculer les bornes inférieures pour tous ces noeuds.
  Mettre ces noeuds dans la liste L par valeurs croissantes des bornes inférieures.
Etape 3 :
  Retirer le premier noeud de la liste L.
Etape 4 :
  Si le noeud est un noeud terminal *Alors*
    Aller à l'étape 6
  Sinon
    Aller à l'étape 5
  FinSi
Etape 5 :
  Générer les noeuds possibles et nécessaires de niveau immédiatement inférieur, à partir des règles de génération.
  Calculer les bornes inférieures associées à ces noeuds et élaguer si nécessaire.
  Ajouter ces noeuds par valeur décroissante du numéro de niveau et, en cas d' égalité, par valeur croissante de la borne dans la liste L.
  Aller à l'étape 3.
Etape 6 :
  Si la borne inférieure du noeud courant est inférieure à C_{max} *Alors*
    Le chemin devient la nouvelle solution.
    Modification de C_{max}
  FinSi
  Si L est vide *Alors*
    Aller à l'étape 7.
  Sinon
    Aller à l'étape 3.
  FinSi
Etape 7 :
  Fin et édition du chemin et de la valeur du C_{max}.

Figure 21. – Algorithme 14 [br-1991].

TABLEAU 9
Données de l'exemple de la PSE de [br-1991].

Étage	J1	J2	J3
1	1	2	3
2	3	4	2

• Le problème traité dans [ra-1992a] est un problème identique au précédent. Cependant les auteurs prétendent déterminer une solution optimale pour un flow-shop hybride « de permutation ». On peut penser que les travaux arriveront tous dans le même ordre sur tous les étages, mais cela implique forcément l'introduction de temps d'inactivité sur les machines, donc une éventuelle augmentation de la valeur du critère. La procédure proposée par

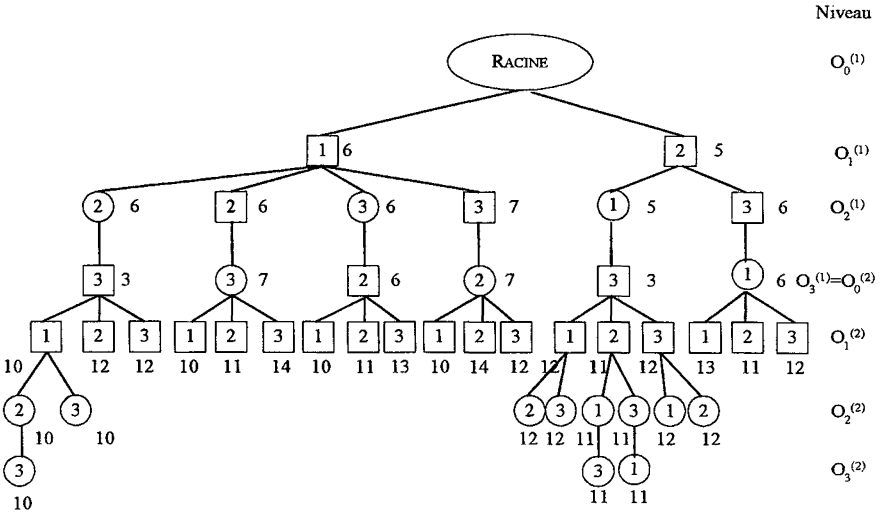

Figure 22. – Arbre de recherche correspondant à l'exemple de la PSE de [br-1991].

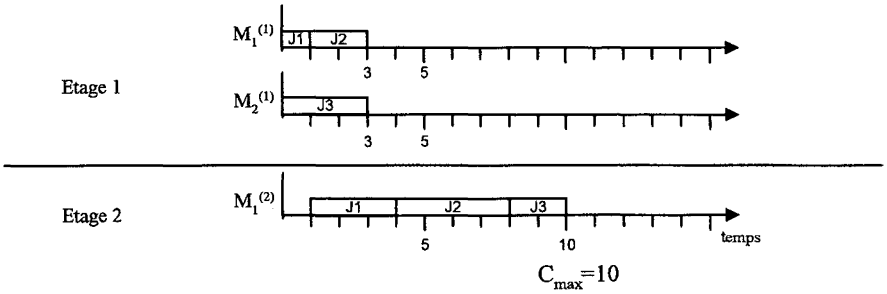

Figure 23. – Diagramme de Gantt correspondant à l'exemple de la PSE de [br-1991].

les auteurs revient à considérer un flow-shop à k machines et à déterminer un ordonnancement pour ce flow-shop. Puis on affecte sur chaque étage les travaux au plus tôt. Il faut être conscient que « l'optimalité » de cette PSE n'est vraie que relativement à l'espace restrictif des solutions explorées. En effet, nous avons comparé les résultats obtenus par cette PSE et ceux obtenus par celle de Brah et Hunsucker. Cette comparaison met clairement en évidence que la PSE de Rajendran et Chaudhuri n'est qu'une heuristique et explique les temps de calcul faibles donnés dans l'article. Aucune étude comparative avec les bornes inférieures existantes ni avec la PSE de [br-1991] n'est fournie. L'algorithme de la PSE est présentée figure 24 et celui du calcul de la borne inférieure figure 25. Il ne s'agit en fait que d'une arborescence permettant d'explorer les $n!$ ordres possibles pour les

PSE de Rajendran et Chaudhuri

Soient ORD la liste des noeuds classés par borne inférieure croissante et CRE celle des noeuds créés non encore parcourus.

Etape 1 :
- Créer un noeud fictif ne contenant aucun travail ordonnancé.
- Créer à partir de celui-ci C_n^P noeuds possédant chacun P travaux. A ce niveau de l'algorithme, l'ordonnancement partiel est déterminé par les P travaux.
- Mettre ces noeuds dans la liste CRE.
- *Remarque : il est clair qu'il n'y a pas de problème d'affectation puisqu'on place P travaux par étage. On a donc toujours plus, ou autant, de machines que de travaux par étage.*
- Mettre à jour les dates de disponibilités des machines.
- *Tant Que* la liste CRE n'est pas vide *Faire*
  - Retirer un noeud de la liste CRE. *Cet élément devient le noeud courant*
  - Calculer la borne inférieure de ce noeud (cf. Figure 25).
  - Placer ce noeud dans la liste ORD à sa place (par borne inférieure croissante). En cas d'égalité, favoriser le noeud possédant le plus grand nombre de travaux. S'il y a encore égalité, placer le dernier noeud créé en tête.
- *Fin Tant Que*

Etape 2 :
- *Tant Que* le premier noeud de la liste ORD ne possède pas (n-1) éléments *Faire*
  - Enlever le premier noeud, noté P_{noeud}, de la liste ORD.
  - *Pour* chaque travail i non encore ordonnancé *Faire*
    - Créer un nouveau noeud, en ajoutant aux travaux de P_{noeud} le travail i.
    - Placer ce nouveau noeud dans la liste CRE
  - *Fin Pour*
  - *Tant Que* la liste CRE n'est pas vide *Faire*
    - Retirer un noeud de la liste CRE. *Cet élément devient le noeud courant*
    - Calculer la borne inférieure de ce noeud (cf. Figure 25)
    - Placer ce noeud dans la liste ORD à sa place (par borne inférieure croissante). En cas d'égalité, favoriser le noeud possédant le plus grand nombre de travaux. S'il y a encore égalité, placer le dernier noeud créé en tête.
  - *Fin Tant Que*
- *Fin Tant Que*

Etape 3 :
- L'ordonnancement associé au premier noeud de la liste ORD constitue la solution.

Figure 24. – Algorithme 15 [ra-1992a].

travaux, sachant que les travaux sont ensuite placés un à un pour toutes leurs opérations au plus tôt sur les machines de tous les étages. A un nœud, on a placé $|\ \sigma\ |$ travaux de la séquence σ considérée, et on en déduit la borne inférieure $LB(\sigma)$ (*cf.* Fig. 25).

On a σ un ordonnancement partiel, N l'ensemble des travaux non ordonnancés, P le plus petit nombre de machines tous étages confondus, $R_j^{(\ell)}(\sigma)$ la date à laquelle la machine j de l'étage ℓ est disponible après avoir traité l'ordonnancement partiel σ. $R_\bullet^{(\ell)}(\sigma)$ est la date de disponibilité minimale à l'étage ℓ après avoir traité l'ordonnancement partiel σ.

• Rajendran et Chaudhuri ([ra-1992b]) se consacrent aussi au même problème mais en considérant le total « flow-time » comme critère à minimiser. Pour la même PSE, ils établissent une nouvelle borne inférieure.

Procédure de calcul de la borne inférieure de la PSE de Rajendran et Chaudhuri

Etape 1 :

$S^{(\ell)}$ *définit la date à laquelle les travaux non ordonnancés peuvent débuter leur exécution à l'étage* ℓ. *Ainsi un travail qui n'est pas fini à un niveau donné, ne peut pas commencer son exécution aux niveaux supérieurs.*

$$S^{(\ell)} = \max\left\{ R_\bullet^{(1)}(\sigma) + \min_{i\in N}\sum_{h=1}^{\ell-1} p_i^{(h)}, R_\bullet^{(2)}(\sigma) + \min_{i\in N}\sum_{h=2}^{\ell-1} p_i^{(h)}, ..., R_\bullet^{(\ell-1)}(\sigma) + \min_{i\in N} p_i^{(\ell-1)} \right\}$$

$\forall \ell = 2, ..., k$ et $S^{(1)} = 0$

Etape 2 :

La borne inférieure de la durée totale de tous les ordonnancements commençant par σ, à partir de l'étage ℓ, est : $L'^{(\ell)}(\sigma) = \max(R_\bullet^{(\ell)}(\sigma), S^{(\ell)}) + \max\left\{\frac{1}{M^{(\ell)}}\sum_{i\in N} p_i^{(\ell)}, \max_{i\in N}(p_i^{(\ell)})\right\} + \min_{i\in N}\left(\sum_{h=\ell+1}^{k} p_i^{(h)}\right)$

Cette borne se compose de trois termes : à l'étage ℓ, *la plus grande date de début au plus tôt pour les travaux non ordonnancés, une borne inférieure du temps nécessaire pour effectuer les travaux non encore ordonnancés, et enfin la durée minimale pour qu'un travail non ordonnancé puisse finir son exécution sur l'ensemble des étages.*

Etape 3 :

Calculer la borne inférieure de la duréee totale sur le dernier étage

$$L'^{(k)}(\sigma) = \max\left\{ \max_{1\le j\le M^{(k)}}(R_j^{(k)}(\sigma)), \max(R_\bullet^{(k)}(\sigma), S^{(k)}) + \max\left(\frac{1}{M^{(k)}}\sum_{i\in N} p_i^{(k)}, \max_{i\in N} p_i^{(k)}\right)\right\}$$

Etape 4 :

Calculer la borne inférieure de ce noeud par $LB(\sigma) = \max_{1\le\ell\le k}(L'^{(\ell)}(\sigma))$

Figure 25. – Algorithme 16 [ra-1992a].

Dans une deuxième partie, ils donnent une heuristique, présentée figure 26, pour résoudre ce même problème à deux étages.

• Un problème de flow-shop hybride très particulier est abordé dans [hu-1992] : le nombre de travaux dans le système de production est borné par une valeur pré-déterminée. Il est dit contraint. Pour les $FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid contraint, d_i \mid \bar{T}$ et $FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid contraint, d_i \mid \sum_i U_i$, une simulation sous SIMAN permet de tester six règles de priorité : SPT, LPT, FIFO, LIFO, MWR et LWR. Elles sont utilisées pour classer les travaux dans les files d'attente à chaque étage. Les tests montrent que pour le retard moyen, la règle FIFO est la meilleure et que pour le nombre de travaux en retard, les résultats varient selon le nombre de travaux maximum dans le système. La règle SPT donne des résultats intéressants pour les deux critères. La variation du nombre d'étages et de machines ne semble pas avoir d'effets significatifs.

• Sawik [saw-1993] aborde le $FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid b^{(\ell)(\ell+1)}, a^{(\ell)(\ell+1)} \mid C_{max}$, où $a^{(\ell)(\ell+1)}$ est un décalage temporel minimum à respecter, entre les opérations ℓ et $\ell+1$ [sz-1983] et nous propose une heuristique. La

Heuristique de Rajendran et Chaudhuri

Etape 1 :
- Constituer $\Psi = \{i \in N \mid R_\bullet^{(1)}(\sigma) + p_i^{(1)} \leq R_\bullet^{(2)}(\sigma)\}$
- *Si* Ψ est vide *Alors*
  - Aller à l'étape 3
- *Sinon*
  - Aller à l'étape 2
- *FinSi*

Etape 2 :
- Soit q le travail non ordonnancé ayant la plus petite durée au deuxième étage (en cas d'égalité celui qui a en plus la plus petite durée au premier étage)
- Ajouter le travail q à l'ordonnancement σ (à chaque étage on place au plus tôt)
- Enlever le travail q de N.
- Aller à l'étape 4.

Etape 3 :
- Soit q le travail non ordonnancé ayant la plus petite durée au premier étage (en cas d'égalité celui qui a en plus la plus petite durée au deuxième étage)
- Ajouter le travail q à l'ordonnancement σ (à chaque étage on place au plus tôt)
- Enlever le travail q de N.

Etape 4 :
- Mise à jour des dates de disponibilité des machines.
- *Si* N est vide *Alors*
  - FIN
- *Sinon*
  - Aller à l'étape 1
- *FinSi*

Figure 26. – Algorithme 17 [ra-1992b].

durée d'exécution d'une pièce de type i à l'étage ℓ est $v_i^{(\ell)}$. On note : $Q = \sum_{i=1}^{n} Q_i$ (Q_i est le nombre total de produits i), i_h l'identité de la h^{ième} pièce dans la séquence $(i_{[1]}, i_{[2]}, ..., i_{[Q]})$, $t_h^{(\ell)}$ sa date de début d'exécution à l'étage ℓ, $r_h^{(\ell)}$ sa date de début d'exécution au plus tôt. $I_h^{(\ell)}$ est l'inactivité engendrée sur l'étage ℓ par la pièce h (somme du temps d'attente avant de débuter la fabrication de la pièce h et de la durée de blocage d'une machine jusqu'à ce que la pièce h puisse la quitter). $X_{bh}^{(\ell)}$ le temps total écoulé quand la b^{ième} place du stock, entre les étages ℓ et $\ell + 1$, devient disponible après le traitement des h premières pièces ($b = 1, ..., b^{(\ell)(\ell+1)}$) et $x_h^{(\ell)} = \min_{1 \leq b \leq b^{(\ell)(\ell+1)}} X_{bh}^{(\ell)}$. $b(h)^{(\ell)}$ est le numéro de la place disponible le plus tôt dans le stock, entre les étages ℓ et $\ell + 1$, après le traitement des $h - 1$ premières pièces sur l'étage ℓ. $Y_{jh}^{(\ell)}$ est le temps total écoulé quand la machine j de l'étage ℓ devient disponible après le traitement des h premières pièces et $y_h^{(\ell)} = \min_{1 \leq j \leq M^{(\ell)}} Y_{jh}^{(\ell)}$. $M(h)^{(\ell)}$ est le numéro de la machine ayant la date de disponibilité la plus petite après le traitement des $h - 1$ premières pièces.

Le raisonnement se fait pièce par pièce. À chaque itération, l'ordonnancement complet d'une pièce est réalisé (*i.e.* sur tous les étages). On sélectionne la pièce qui minimise le temps total d'inactivité. L'algorithme RITM (« Route Idle Time Minimization ») est donné figure 27.

Algorithme RITM

Etape 0 : *Détermination d'une pondération pour chaque pièce*

Choisir cst_1 et $cst_2 > 1$ deux constantes

Classer les pièces par $\bar{v}_i$ décroissant où $\bar{v}_i = \sum_{\ell=1}^{k}(1 + cst_1 \times (\ell - 1)) \times v_i^{(\ell)}$

A la $h^{ième}$ pièce, on associe un poids ρ_h égal à cst_2^{n-h}

$X_{b0}^{(\ell)} = 0, \forall \ell = 1, ..., k-1$ et $b = 1, ..., b^{(\ell)(\ell+1)}$

$Y_{j0}^{(\ell)} = 0, \forall \ell = 1, ..., k$ et $j = 1, ..., M^{(\ell)}$

$b(1)^{(\ell)} = 1, \forall \ell = 1, ...k-1$

$M(1)^{(\ell)} = 1, \forall \ell = 1, ...k$

$\sigma = \emptyset$, $h = 1$

Etape 1 : *Sélection d'une pièce*

Pour $i \in N - \sigma$ *Faire*

$t_i^{(\ell)} = \max\{r_i^{(\ell-1)} + a^{(\ell)(\ell+1)}, y_{i-1}^{(\ell)}\}, \forall \ell = 1, ..., k-1$

$C_i^{(\ell)} = t_i^{(\ell)} + v_i^{(\ell)}, \forall \ell = 1, ..., k$

$r_i^{(\ell)} = \max\{C_i^{(\ell)}, x_{i-1}^{(\ell+1)} - a^{(\ell)(\ell+1)}\}, \forall \ell = 1, ..., k-2$

$r_i^{(k)} = C_i^{(k)}$

$I_i^{(\ell)} = (t_i^{(\ell)} - y_{i-1}^{(\ell)}) + (r_i^{(\ell)} - C_i^{(\ell)})$

FinPour

Sélectionner la $h^{ième}$ pièce dans les produits de type i qui minimise le temps total d'inactivité pondéré c'est-à-dire $\min_{i \in N-\sigma} \frac{\sum_{\ell=1}^{k} I_i^{(\ell)}}{\rho_i}$.

Etape 2 : *Complément de l'ordonnancement partiel par la $h^{ième}$ pièce , c'est-à-dire calculs de $t_h^{(\ell)}$, $C_h^{(\ell)}$, $r_h^{(\ell)}$ sur la machine $M(h)^{(\ell)}$ $\forall \ell = 1, ..., k$ et à la place $b(h)^{(\ell)}$ $\forall \ell = 1, ..., k-1$ du stock*

Etape 3 :

Pour le type de produit i auquel appartenait la $h^{ième}$ pièce *Faire*

$Q_i = Q_i - 1$

FinPour

Si $Q_i = 0$ *Alors*

$\sigma = \sigma \cup \{i\}$

FinSi

Si $\sigma = N$ *Alors*

FIN

Sinon

Calculer $X_{bh}^{(\ell)}$, $Y_{jh}^{(\ell)}$, $\forall \ell = 1, ..., k-1$ (resp k), $\forall j = 1, ..., M^{(\ell)}$ et $\forall b = 1, ..., b^{(\ell)(\ell+1)}$

Pour chaque étage ℓ *Faire*

Déterminer $b(h+1)^{(\ell)}$ par $\min_{1 \le b \le b^{(\ell)(\ell+1)}} X_{bh}^{(\ell)}$

Déterminer $M(h+1)^{(\ell)}$ par $\min_{1 \le j \le M^{(\ell)}} Y_{jh}^{(\ell)}$

FinPour

$h = h + 1$

Aller à l'étape 1

FinSi

Figure 27. – Algorithme 18 [saw-1993].

Cet algorithme peut donc prendre en compte l'arrivée de pièces en temps réel.

• Les problèmes de flow-shop hybride contraint $FHk,((PM^{(\ell)})_{\ell=1}^{k}) \mid contraint, r_i \mid \bar{F}$ et $FHk,((PM^{(\ell)})_{\ell=1}^{k}) \mid contraint, r_i \mid F_{max}$ sont à nouveau abordés dans [hu-1994]. Hunsucker et Shah y étudient les mêmes règles de classement : la règle SPT donne les meilleures performances et LPT les plus mauvaises. La règle FIFO donne également de très bons résultats pour des systèmes congestionnés voire très congestionnés.

• Vandevelde [va-1994] traite le $FHk,((PM^{(\ell)})_{\ell=1}^{k}) \mid\mid C_{max}$. Pour cela, elle relaxe le problème initial en un problème à machines parallèles, en se focalisant sur un étage goulet. Puis elle établit un ensemble de bornes inférieures et deux bornes supérieures, en utilisant étage par étage un algorithme de Carlier [ca-1987]. Les bornes supérieures donnent de bonnes approximations. En revanche, la relaxation du problème semble trop restrictive car il n'est pas toujours possible d'identifier un seul étage goulet. L'auteur suggère alors de relaxer le problème en considérant 2 étages goulets.

• Lee et Vairaktarakis [lee-1994] abordent le même problème et suggèrent une heuristique, H', pour le résoudre. Celle-ci, présentée figure 28, est bâtie un peu dans le même esprit que CDS [ca-1970].

Algorithme de l'heuristique H'

Etape 1 :
  Pour r=1, ..., k/2 *Faire*
    Appliquer l'algorithme H de Lee et Varaktarakis (1994) sur les étages $2r-1$ et $2r$. S_r est l'ordonnancement obtenu.
  FinPour
Etape 2 :
  Concaténer tous les ordonnancements obtenus.
  Retirer les temps d'inactivité non nécessaires (la mise en oeuvre de cette étape n'apparaît pas évidente).

Figure 28. – Algorithme 19 [lee-1994].

Les auteurs montrent que :

$$\frac{C_{max}^{(H')}}{C_{max}^{*}} \leq k - \frac{1}{\max\{M^{(1)}, M^{(2)}\}} - \ldots - \frac{1}{\max\{M^{(k-1)}, M^{(k)}\}}.$$

• Santos, Hunsucker et Deal proposent dans [sa-1995] une borne inférieure pour le $FHk,((PM^{(\ell)})_{\ell=1}^{k}) \mid\mid C_{max}$. On pose $RS(i,\ell) = \sum_{\ell'=\ell+1}^{k} p_i^{(\ell')}$ et RSA la liste ordonnée par ordre croissant des $RS(i,\ell)$ (les éléments de

cette liste sont notés $RSA(i,\ell)$ et donc $RSA(1,\ell) \leq RSA(2,\ell) \leq ... \leq RSA(n,\ell)$). De même les auteurs définissent $LS(i,\ell) = \sum_{\ell'=1}^{\ell-1} p_i^{(\ell')}$ ainsi que la liste ordonnée LSA des valeurs croissantes de $LS(i,\ell)$. La borne inférieure s'exprime pour chaque étage ℓ par :

$$LB^{(\ell)} = \frac{1}{M^{(\ell)}} \times \left\{ \sum_{j=1}^{M^{(\ell)}} LSA(j,\ell) + \sum_{i=1}^{n} p_i^{(\ell)} + \sum_{j=1}^{M^{(\ell)}} RSA(j,\ell) \right\}$$

$$LB^{(0)} = \max_{1 \leq i \leq n} \left\{ \sum_{\ell=1}^{k} p_i^{(\ell)} \right\}.$$

Les erreurs relatives moyennes par rapport à la valeur optimale sont toujours inférieures à 8 % (résultats expérimentaux obtenus par rapport à la solution optimale obtenue par la PSE de Brah et Hunsucker).

• Sawik [saw-1995] traite le $FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid b^{(\ell)(\ell+1)} = 0, a^{(\ell)(\ell+1)} \mid C_{max}$. L'heuristique proposée est identique à celle dans [saw-1993] : seules les informations relatives aux pondérations et aux stocks sont supprimées.

• Guinet, Botta et Solomon [gu-1995] proposent une modélisation en programmation linéaire entière 0-1 pour le problème $FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid d_i \mid T_{max}$. Elle ne diffère de celle proposée dans [gu-1996] que par la fonction objectif ($minZ = T_{max}$) et par la relation ($C_i^{(k)} - d_i \leq T_{max}$). Les auteurs proposent trois heuristiques qui se décomposent toutes les trois en deux phases. La première est une phase d'ordonnancement basée soit sur la procédure CDS [ca-1970], soit sur la procédure NEH [na-1983] (en créant dans les deux cas un $k+1^{ème}$ étage et en posant $p_i^{(k+1)} = -d_i$ et $M = k+1$ le nombre de machines dans le flow-shop), soit sur celle de Townsend [to-1977] (avec $M = k$). La deuxième phase permet d'affecter les travaux sur les machines des différents étages selon deux règles : la première consiste à affecter un travail de sorte que sa date de fin d'exécution soit minimum, en prenant la machine qui a été libérée la dernière, la deuxième règle consiste à affecter le travail à la dernière machine libre de manière à terminer ce travail au plus tôt dans l'atelier suivant et répéter ceci pour chaque étage (cette dernière nécessiterait quelques éclaircissements).

• Le $FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid b^{(\ell,\ell+1)}, S^{(\ell,\ell+v)} \mid C_{max}$ est traité dans [he-1996]. Il s'agit d'un problème qui présente un intérêt fondamental dans

l'industrie. $S^{(\ell,\ell+v)}$ représente un nombre de supports nécessaires entre l'étage ℓ et $\ell+v$. Les stocks inter-étage sont exprimés en capacité de travaux et non pas en nombre de travaux. Les auteurs proposent une heuristique qui dans une première étape détermine une séquence et qui, dans une deuxième étape, affecte au plus tôt les travaux dans l'ordre de cette séquence. Les travaux sont ensuite classés par ordre croissant de leur date de fin croissante. La deuxième étape est répétée jusqu'à ce que l'affectation des travaux soit réalisée au dernier étage.

Quatre règles initales sont testées :

- classement par $ind(i)$ défini par $ind(i) = \sum_{\ell=1}^{k}(k-(2\times\ell-1))p_i(\ell)$;
- utilisation de l'algorithme de CDS en utilisant $p_{i1}^{(j)} = \sum_{\ell=1}^{j} p_i^{(\ell)}$ et $p_{i2}^{(2)} = \sum_{k-j+1}^{k} p_i^{(\ell)}$ pour bâtir le pseudo-problème j sur lequel l'algorithme J sera utilisé ;
- utilisation de l'algorithme de Dannenbring en utilisant

$$p_{i1} = \sum_{\ell=1}^{k}(k-\ell+1)\times p_i^{(\ell)} \qquad \text{et} \qquad p_{i2} = \sum_{\ell=1}^{k}\ell\times p_i^{(\ell)},$$

- utilisation de la procédure NEH.

Les résultats montrent que les résultats sont meilleurs quand la séquence initiale est obtenue par l'adaptation de Dannenbring.

• Dans [bo-1996], des problèmes de flow-shop hybrides à k étages incluant un certain nombre de contraintes sont abordés (recouvrement, temps de montage et de démontage non dépendant de la séquence, contraintes de précédence, ...). Les critères considérés sont le C_{max} et le L_{max}. Les heuristiques développées sont très proches de celles mises en œuvre par [gu-1995], [he-1996] (par exemple). Elles ne sont pas détaillées ici. Signalons que dans chacun des cas, signalés ci-dessus, l'auteur fournit les formules analytiques de calcul du $C_i^{(\ell)}$, très utiles pour l'établissement des procédures « $makespan(x)$ ».

• Une amélioration de la PSE de Brah et Hunsucker est proposée dans [po-1996]. D'une part les bornes inférieures sont améliorées. D'autre part un algorithme génétique, basé sur un codage direct ternaire [dj-1996], est introduit dans la PSE afin d'améliorer à certains étages la valeur de la borne supérieure. Cet algorithme tient compte de l'ensemble des décisions prises

dans l'arbre et construit une série de populations constituées de solutions complètes.

• Vignier, Billaut et Proust étudient le $FHk,((PM^{\ell}(t))_{\ell=1}^{k}) \mid r_i^{(1)}, \tilde{d}_i^{(k)}, pmtn, T_{max} = 0 \mid C_{max}$ [vi-1996a]. Chaque machine a un calendrier de fonctionnement et le critère à minimiser est la date d'achèvement total parmi les ordonnancements sans retard, s'il en existe. Une méthode exacte, basée sur la PSE de Brah et Hunsucker, est présentée. Le calcul des bornes inférieures du C_{max} est le même que dans [po-1996]. Un graphe biparti permet de calculer une borne inférieure au retard maximum.

• Le $FHk,((PM^{(\ell)})_{\ell=1}^{k}) \mid\mid \bar{C}$ est traité dans [vi-1996c]. Les auteurs proposent une méthode exacte, toujours basée sur l'architecture présentée dans [br-1991]. En revanche quatre nouvelles bornes sont proposées. Ils montrent que LB_2 n'est pas dominée par LB_1 mais que LB_4 domine LB_3 qui n'est pas présentée ici. Soient $CM_j^{(\ell)}$ la date de disponibilité de la machine j à l'étage ℓ, EST la plus petite date de début au plus tôt des travaux non encore ordonnancés à l'étage ℓ (*i.e.* appartenant à $N-A$); elle est égale à $\max\{\min_{j\in M^{(\ell)}} CM_j^{(\ell)}, \min_{i\in N-A} C_i^{(\ell-1)}\}$. $\eta^{(\ell)}(x)$ est la $x^{ième}$ plus petite durée d'exécution à l'étage ℓ, parmi les travaux appartenant à $N-A$.

$$LB1 = \sum_{i\in A}\left\{C_i^{(\ell)} + \sum_{\ell'=\ell+1}^{k} p_i^{(\ell')}\right\} + \sum_{h\in N-A}\left\{C_h^{(\ell-1)} + \sum_{\ell'=\ell}^{k} p_h^{(\ell')}\right\}$$

$$LB2 = \sum_{i\in A}\left\{C_i^{(\ell)} + \sum_{\ell'=\ell+1}^{k} p_i^{(\ell')}\right\} + \mid N-A \mid \times\left\{EST + \min_{h\in N-A}\left(\sum_{\ell'=\ell}^{k} p_h^{(\ell')}\right)\right\}$$

$$LB4 = \sum_{i\in A}\left\{C_i^{(\ell)} + \sum_{\ell'=\ell+1}^{k} p_i^{(\ell')}\right\} + \mid N-A \mid \times EST + \sum_{h\in N-A}\sum_{\ell'=\ell}^{k} p_h^{(\ell')} + \sum_{\delta=1}^{div^{(\ell)}-1} \delta \times M^{(\ell)} \times \eta^{(\ell)}(\delta) + mod^{(\ell)} \times div^{(\ell)} \times \eta^{(\ell)}(div^{(\ell)})$$

avec $\mid N-A \mid = div^{(\ell)} \times M^{(\ell)} + mod^{(\ell)}$.

3.3. Applications du flow-shop hybride en milieu industriel

• Paul [pa-1979] propose une simulation sur un problème de flow-shop hybride à deux étages modélisant la production de bouteilles de verre

(des fours au premier étage et des machines de moulage au second). Paul réduit ce problème à un problème d'affectation sur M machines parallèles identiques en ignorant les fours du premier étage. Cette simulation a pour objet de comparer plusieurs politiques existantes dans l'entreprise à la règle SPT qui se révèle être la meilleure. Cette règle est appliquée sur les travaux classés par niveau de stock afin de ne pas toujours choisir le travail ayant la plus petite taille. Le nombre de travaux en rupture de stock ou en retard et le temps moyen de stockage sont les deux critères abordés.

• Dans un contexte industriel identique, un Système Interactif d'Aide à la Décision est présenté dans [pr-1995]. Des temps de préparation dépendant des séquences (S_{sd}) existent au premier étage, ainsi que des recouvrements entre le premier et le deuxième étage; le « splitting » est autorisé. Les tailles des sous-lots ne sont pas données. Un coût global de production est à optimiser. Ce logiciel est opérationnel depuis 1977 dans un grand groupe industriel français.

• Narasimhan et Panwalkar [na-1984], dans un contexte de fabrication de câbles, proposent l'heuristique CMD (Cumulative Minimum Deviation) pour un flow-shop hybride à deux étages en vue de minimiser la somme des temps morts et les en-cours au deuxième étage, avec le « splitting » imposé au deuxième étage (noté $FH2,(1,Q2) \mid split^{(2)} \mid$ somme des temps morts + en-cours au deuxième étage). La machine du premier étage fonctionne suffisamment rapidement pour qu'il n'y ait pas d'attente au deuxième étage. De plus $p_i^{(1)} = 0.4 \times p_{i1}^{(2)} + 0.6 \times p_{i2}^{(2)}$. Les auteurs proposent de comparer SPT, LPT, MD (Minimum Deviation) et CMD. CMD se révèle la meilleure. Ils définissent les quantités $\Delta_i = p_{i1}^{(2)} - p_i^{(1)}$ et $\lambda_i = p_{i2}^{(2)} - p_i^{(1)}$. Les règles SPT et LPT sont appliquées sur les durées d'exécution au premier étage. S'il y a indétermination alors les travaux sont classés par $\mid \Delta_i \mid + \mid \lambda_i \mid$ croissant. La règle MD consiste à classer les travaux par $\mid \Delta_i \mid + \mid \lambda_i \mid$ croissant. S'il y a indétermination pour la $i^{ème}$ position alors on choisit le travail h tel que $\mid \lambda_{[i-1]} \mid + \mid \Delta_h \mid \geq 0$. Enfin, la règle CMD place à la $i^{ème}$ position le travail qui minimise la quantité ajoutée à Z, où Z = temps d'inactivité + en-cours au deuxième étage. S'il y a indétermination, on applique la même règle que dans MD. En 1987, les auteurs généralisent la règle CMD à la règle GCMD pour un problème de flow-shop hybride avec un nombre différent de machines au premier et au deuxième étage, supérieur ou égal à 1 [na-1987]. Kadipasaoylu, Xiang et Khunawola [ka-1997] ont repris ces travaux d'une part en rajoutant l'étude de

deux règles de séquencement (valeur ajoutée au travail) dans le cas statique, d'autre part en effectuant une étude supplémentaire dans le cas dynamique lorsque les travaux disposent de dates de fin souhaitées. Au passage ils ont montré l'équivalence des règles SPT et MD dans les travaux originels en raison des conditions d'expérimentation. Les ordonnancements sont évalués d'après de nombreux critères de performance classiques et un économique. Dans le cas statique, les auteurs confirment le bon comportement de GCMD, mettent en évidence qu'une minimisation des temps morts des machines et des temps d'attente des travaux ne conduit pas nécessairement ici à une réduction de la durée totale et que les règles économiques de séquencement ont aussi un bon comportement. Dans le cas dynamique, d'autres règles sont ajoutées pour tenir compte des dates échues des travaux. L'étude montre un bon comportement global de leur part alors que GCMD se révèle beaucoup plus fragile que dans le premier cas. Les règles économiques, ainsi que SPT, ne produisent pas de bons résultats vis-à-vis des retards. Dans tous les cas, LPT est peu performante.

• Un problème de flow-shop hybride à 2 étages modélisant une industrie de papier est étudié dans [sh-1990]. L'approche utilisée est fondée sur la résolution des problèmes d'ordonnancement à machines parallèles. La fonction objectif à minimiser est la somme du coût d'affectation et du coût lié aux changements tout en respectant des contraintes d'équilibrage de chaînes, des dates de début au plus tôt et de fin des travaux, de dépendances technologiques. Le problème peut se noter $FH2, ((RM^{(\ell)})_{\ell=1}^{2}) \mid r_i^{(1)}, d_i^{(2)}, S_{sd}, DT^{(\ell,\ell+1)}, balancing \mid \sum$ Coût. Ce problème est décomposé en un problème d'affectation et de séquencement.

L'algorithme général est donné figure 29. En ce qui concerne la phase d'affectation, l'heuristique proposée est une procédure par construction qui affecte les travaux aux machines en une passe en respectant si possible les dates de fin souhaitées et la charge maximale des machines. Quand cela n'est pas possible on essaie de satisfaire dans un premier temps la contrainte liée à la charge maximale des machines et ensuite les dates de fin souhaitées. Pour la phase de séquencement une heuristique basée sur EDD et sur une méthode exacte est utilisée.

• Un problème de flow-shop hybride à trois étages ($FH3, (P6, P3, 1) \mid d_i \mid C_{max}$ et T_{max}) modélisant une entreprise de chimie fine est traité dans [fo-1996]. Le modèle proposé englobe deux heuristiques : la première permet d'affecter les différents travaux sur les deux premiers étages à partir d'une séquence donnée et la deuxième utilise un algorithme soit de recuit simulé

Algorithme de l'heuristique de [Sherali et al.]

Etape 1 : Initialisation
  Résoudre le sous-problème d'affectation au second étage
  Calculer les dates dues induites pour les utilisées au premier étage.
Etape 2 : Résolution au premier étage
  Résolution du sous problème d'affectation.
  Résolution du sous problème de séquencement.
  Calculer les dates de début au plus tôt au second étage.
Etape 3 : Résolution au deuxième étage
  Résolution du sous problème d'affectation.
  Résolution du sous problème de séquencement.
  Si les contraintes "principales" sont respectées *Alors*
    Fin
  Sinon
    Calculer les dates dues induites pour les utilisées au premier étage.
    Aller à l'étape 2

Figure 29. – Algorithme 20 [sh-1990].

soit de recherche tabou afin d'optimiser l'ordre de passage des travaux sur les différentes machines.

• Un problème de flow-shop hybride avec des temps de montage dépendant de la séquence est traité dans [ag-1995] et modélise une fabrication de tapis. Chaque étage est composé de plusieurs groupes de machines identiques. Dans un premier temps les auteurs donnent la modélisation du problème d'affectation et d'ordonnancement (les tailles de lots sont imposées). Dans un deuxième temps, les temps unitaires de fabrication et les quantités à produire sont donnés et ils modélisent alors le problème de dimensionnement des sous-lots sur un horizon discrétisé. On peut regretter que le résultat de la phase de dimensionnement n'ait pas, semble-t-il, été utilisé en entrée de la modélisation analytique traitant le problème d'affectation et d'ordonnancement (*cf.* [la-1994]...). Les auteurs suggèrent enfin le squelette d'un algorithme non optimal qui se déroule en deux phases. La première permet de déterminer les quantités à produire étage par étage. La deuxième permet d'ordonnancer groupe par groupe les différents travaux. On notera l'absence d'expérimentations de cette méthode.

4. TYPOLOGIE DES PROBLÈMES DE FLOW-SHOP HYBRIDE

En parcourant les branches de l'arbre représenté figure 30, on retient un certain nombre de contraintes pour aboutir aux feuilles qui représentent les problèmes particuliers. Comme on peut le constater, hormis dans les cas

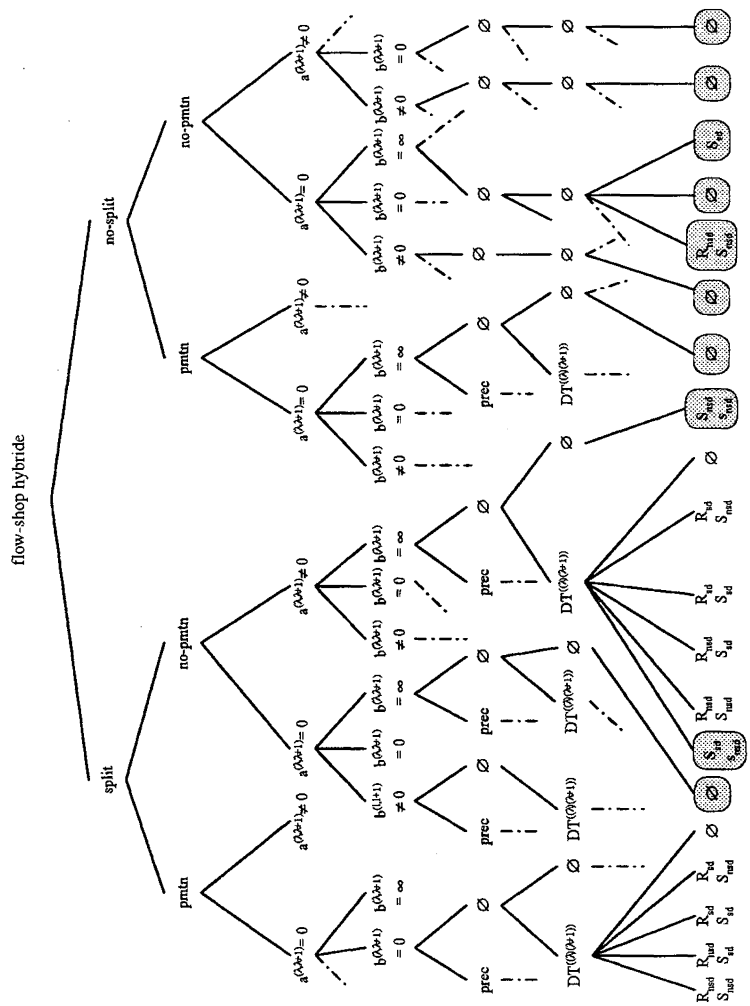

Figure 30. – Typologie des problèmes d'ordonnancement de type flow-shop hybride.

TABLEAU 10
Les problèmes à k étages.

Problème	Référence
$FHk, ((RM^{(\ell)})_{\ell=1}^{k}) \mid S_{sd} \mid C_{max}$	[ag-1995]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid no-wait \mid C_{max}$	[sa-1973]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid\mid C_{max}$	[br-1991]
	[lee-1994]
	[va-1994]
	[sa-1995]
	[po-1996]
	[gu-1996]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid\mid \bar{C}$	[vi-1996c]
$FHk, ((PM^{(\ell)}(t))_{\ell=1}^{k}) \mid r_i^{(1)}, \tilde{d}_i^{(k)}, pmtn, T_{max}=0 \mid C_{max}$	[vi-1996a]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid$ permutation $\mid C_{max}$	[ra-1992a]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid$ permutation $\mid \bar{F}$	[ra-1992a]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid b^{(\ell)(\ell+1)}, a^{(\ell)(\ell+1)} \mid C_{max}$	[saw-1993]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid b^{(\ell)(\ell+1)}=0, a^{(\ell)(\ell+1)} \mid C_{max}$	[saw-1995]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid contraint, d_i \mid \bar{T}$	[hu-1992]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid contraint, d_i \mid \sum_i U_i$	[hu-1992]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid contraint, r_i \mid \bar{F}$	[hu-1994]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid contraint, r_i \mid F_{max}$	[hu-1994]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid d_i \mid T_{max}$	[gu-1995]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid b^{(\ell,\ell+1)}, S^{(\ell,\ell+v)} \mid C_{max}$	[he-1996]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid périodique, b^{(\ell)(\ell+1)} \mid$ max(*rendements*) et min(*encours*)	[wi-1985]
$FHk, ((PM^{(\ell)})_{\ell=1}^{k}) \mid b^{(\ell)(\ell+1)} \mid$ max(*rendements*) et min(*encours*)	[wi-1988]
$FH3, (P6, P3, 1) \mid d_i \mid C_{max}$ et T_{max}	[fo-1996]

industriels, peu de contraintes variées sont considérées habituellement dans la littérature spécialisée.

Les tableaux 10 et 11 présentent un résumé de l'ensemble des problèmes traités que nous avons recensés dans la littérature.

TABLEAU 11
Les problèmes à 2 étages.

Problème	Référence
$FH2, ((PM^{(\ell)})_{\ell=1}^{2}) \mid\mid$ rupture + temps stockage	[pa-1979]
$FH2, ((RM^{(\ell)})_{\ell=1}^{2}) \mid S_{sd}^{(1)}, s_{nsd}^{(2)}, a_i^{(1)(2)}, DT^{(1,2)}, split, ... \mid$ Coût	[pr-1995]
$FH2, ((RM^{(\ell)})_{\ell=1}^{2}) \mid r_i^{(1)}, d_i^{(1)}, S_{sd}, DT^{(\ell,\ell+1)}, balancing \mid \sum$ Coût	[sh-1990]
$FH2, (1, Q2) \mid split^{(2)}, taille\ imposée \mid \sum tempsmorts + encours$	[na-1984]
$FH2, ((PM)^2) \mid\mid C_{max}$	[sr-1989]
	[de-1991]
$FH2, (PM, PM) \mid transport \mid C_{max}$	[la-1987]
$FH2, ((PM^{(\ell)})_{\ell=1}^{2}) \mid\mid C_{max}$	[sh-1972]
	[bu-1973]
	[lee-1994]
	[gu-1996]
$FH2, ((PM^{(\ell)})_{\ell=1}^{2}) \mid\mid \bar{F}$	[ra-1992b]
$FH2, (PM^{(1)}, 1) \mid\mid C_{max}$	[gu-1988]
	[gup-1995]
$FH2, (1, PM^{(2)}) \mid\mid C_{max}$	[sr-1989]
	[gu-1991]
$FH2, (PM^{(1)}, 1) \mid pmtn^{(1)}, d_i \mid T_{max}$	[vi-1996b]
$FH2, (PM^{(1)}(t), 1) \mid pmtn^{(1)}, d_i \mid T_{max}$	[vi-1996b]
$FH2, (1, PM^{(2)}) \mid S_{nsd}, s_{nsd}, split^{(2)}, a^{(1)(2)} \mid C_{max}$	[li-1996]
$FH2, (P2, 1) \mid split^{(1)}, taille\ des\ sous\text{-}lots\ imposée \mid C_{max}$	[lee-1993]
$FH2, (1, PM^{(2)}) \mid S_{snd}, R_{nsd} \mid C_{max}$	[gu-1994]
$FH2, ([P, Q, R]M, [P, Q, R]M) \mid DT^{(1,2)}, split \mid C_{max}$	[vi-1995]

Proust a montré la variété des problèmes de flow-shop résolus par des adaptations de l'algorithme J [pr-1992]. Cette étude confirme l'importance des idées de S. M. Johnson et montre qu'elles sont également intéressantes pour les problèmes de flow-shop hybride. Les heuristiques se décomposent pratiquement toujours en deux étapes : la première se consacre à la détermination d'une séquence et la deuxième au problème d'affectation

(dans l'ordre de la séquence pré-définie). Ceci est restrictif car dans la grande majorité des cas le problème conjoint de dimensionnement en sous-lots n'est pas abordé.

5. CONCLUSION

Cet article présente un panorama des problèmes de flow-shop hybride traités dans la littérature. Cependant nous avons montré que les approches réalisées jusqu'à présent ne concernaient que des cas relativement particuliers de ce problème : elles se focalisent sur les problèmes sous-jacents de séquencement et d'affectation et ignorent à la quasi-unanimité celui du dimensionnement des sous-lots de fabrication. Nous avons également proposé une notation et une typologie. Cette dernière montre à la fois l'ensemble des problèmes déjà abordés et ceux restant à étudier qui sont très nombreux. Enfin nous avons insisté sur l'importance de s'intéresser aux flow-shops hybrides à deux étages afin d'obtenir de très bonnes solutions, voire des solutions optimales, pour construire ensuite de bonnes solutions pour les problèmes à k étages. De nombreux cas industriels auxquels nous sommes confrontés nous obligent à aborder le problème du flow-shop hybride dans sa plus grande globalité.

REMERCIEMENTS

Les auteurs remercient sincèrement les relecteurs, dont le Professeur M-C. Portmann, pour leur travail, leurs remarques et leurs conseils qui ont contribué à la mise au point de cet article.

RÉFÉRENCES

[ag-1995] E. H. Aghezzaf, A. Artiba, O. Moursli et C. Tahon, Hybrid flowshop problems, a decomposition based heuristic approach. In *International Conference on Industrial Ingineering and Production Management (IEPM'95)*, avril 1995, Marrakech (Maroc), FUCAM/IFIP/INRIA, p. 43-56.

[al-1997] F. Alexandre, C. Cardeira, F. Charpillet, Z. Mammeri et M.-C. Portmann, *Compu-search Methodologies II : Scheduling using Genetic Algorithms and Artificial Neural Networks, dans The Planning and Scheduling of Production System*, A. Artiba et S. E. Elmaghraby, Chapman & Hall edition, 1997, p. 301-336.

[ba-1974] K. R. Baker, *Introduction to sequencing and scheduling*, John Willey & Sons, Inc., 1974.

[ba-1987] P. BAPTISTE, C. H. CHO, J. FAVREL et M. ZOUHRI, Une caractérisation analytique des ordonnancements admissibles sous contraintes hétérogènes en flow-shop, *RAIRO-APII*, 1987, *25*, p. 87-102.

[be-1982] R. BELLMAN, A. O. ESOGBUE et I. NABESHIMA, *Mathematical aspects of scheduling and applications*, volume 4 of *International Series in Modern Applied Mathematics and Computer Science*, Pergamon Press edition, 1982.

[ro-1996] J-C. BILLAUT et F. ROUBELLAT, A new method for workshop real time scheduling, *Int. J. Prod. Res.*, 1996, *34*, n° 6, p. 1555-1579.

[bl-1994] J. BLAZEWICZ, K. ECKER, G. SCHMIDT et J. WEGLARZ, *Scheduling in Computer and Manufacturing Systems*, Springer-Verlag edition, 1994.

[bo-1996] V. BOTTA-GENOULAZ, *Planification et ordonnancement d'une succession d'ateliers avec contraintes*, PhD thesis, Université Claude Bernard-Lyon 1, décembre 1996.

[br-1991] S. A. BRAH et J. L. HUNSUCKER, Branch and bound algorithm for the flow shop with multiple processors, *European Journal of Operational Research*, 1991, *51*, p. 88-99.

[br-1975] P. BRATLEY, M. FLORIAN et P. ROBILLARD, Scheduling with earliest start and due date constraints on multiple machines, *Naval Research Logistics Quarterly*, 1975, *22*, n° 1, p. 165-173.

[bu-1973] R. E. BUTEN et V. Y. SHEN, A scheduling model for computer systems with two classes of processors, *Sagomore Computer Conference on Parallel Processing*, 1973.

[ca-1970] H. G. CAMPBELL, R. A. DUKEK et M. L. SMITH, A heuristic algorithm for the n job, m machine sequencing problem, *Management Science*, 1970, *16*, n° 10, p. B630-B637.

[Ca-1988] J. CARLIER et P. CHRETIENNE, *Problèmes d'ordonnancement (modélisation/complexité/algorithmes)*, Études et Recherches en Informatique, Masson édition, 1988.

[ca-1987] J. CARLIER, Scheduling jobs with release dates and tails on identical machines to minimize the makespan, *European Journal of Operational Research*, 1987, *29*, p. 298-306.

[ch-1990] Y.-L. CHANG et R. S. SULLIVAN, Schedule generation in a dynamic job shop, *Int. J. Prod. Res.*, 1990, *28*, n° 1, p. 65-74.

[ch-1995] B. CHEN, Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage, *Journal of Operational Research Society*, 1995, *46*, n° 2, p. 234-244.

[co-1994] Collectif, « Ordonnancement et Entreprises : applications concrêtes et outils pour le futur ». In *Journées d'études*, Toulouse (France), CNRS/GdR Automatique/pôle SED/GT3. juin 1994, 315 p.

[de-1991] D. E. DEAL et J. L. HUNSUCKER, The two stage flowshop scheduling problem with m machines at each stage, *Journal of Information and Optimization Sciences*, 1991, *12*, p. 407-471.

[dj-1996] L. DJERID et M.-C. PORTMANN, Genetic algorithm operators restricted to precedent constraints set: genetic algorithm designs with or without branch and bound approach for solving scheduling problems with disjunctive constraints, *Pékin (Chine), IEEE/Systems, Man and Cybernetics (SMC'96)*, octobre 1996, *4*, p. 2922-2927.

[fo-1996] Ph. FORTEMPS, Ch. OST, M. PIRLOT, J. TEGHEM et D. TUYTTENS, A production scheduling case study in a chemical firm using metaheuristics, *European Journal of Operational Research*, 1996, soumis.

[fo-1993] C. FOURE, D. M. GAY et B. W. KERNIGHAN, *AMPL: A Modeling Language for Mathematical Programming*, The Scientific Press edition, 1993.

[go-1993] GOThA, Les problèmes d'ordonnancement, *RAIRO Rech. Opér.*, 1993, *27*, n° 1, p. 77-150.

[gr-1979] R. L. GRAHAM, E. L. LAWLER, J. K. LENSTRA et A. H. G. RINNOOY KAN, Optimization and approximation in deterministic sequencing and scheduling theory: a survey, *Ann. Discrete Math.*, 1979, *5*, p. 287-326.

[gr-1966] R. L. GRAHAM, Bounds for certain multiprocessing anomalies, *The Bell System Technical Journal*, 1966, *45*, p. 1563-1581.

[gu-1995] A. GUINET, V. BOTTA et M. SOLOMON, Minimisation du plus grand retard ou de la plus grande date de fin dans les problèmes d'ordonnancement de type flowshop hybride. In *Journées d'études: « Affectation et Ordonnancement »*, Tours (France), CNRS/GdR Automatique/pôle SED/GT3, septembre 1995, p. 95-111.

[gu-1996] A. GUINET, M. M. SOLOMON, P. K. KEDIA et A. DUSSAUCHOY, A computational study of heuristics for two-stage flexible flowshops, *Int. J. Prod. Res.*, 1996, *34*, n° 5, p. 1399-1415.

[gup-1995] J. N. D. GUPTA, A. M. A. HARIRI et C. N. POTTS, Scheduling a two-stage hybrid flow shop with parallel machines at the first stage, *Mathematics of Industrial Systems (Annals of operations research)*, 1995, à paraître.

[gu-1991] J. N. D. GUPTA et E. A. TUNC, Schedules for a two-stage hybrid flowshop with parallel machines at the second stage, *Int. J. Prod. Res.*, 1991, *29*, n° 7, p. 1489-1502.

[gu-1994] J. N. D. GUPTA et E. A. TUNC, Scheduling a two-stage hybrid flowshop with separable setup and removal times, *European Journal of Operational Research*, 1994, *77*, p. 415-428.

[gu-1988] J. N. D. GUPTA, Two-stage hybrid flowshop scheduling problem, *Operational Research Society*, 1988, *39*, n° 4, p. 359-364.

[ha-1979] R. H. HAYES et S. C. WHEELWRIGHT, Le cycle de vie du processus de production (i) et (ii), *Harvard-L'expansion*, automne 1979, p. 23-32, 99-110.

[he-1996] H. HENTOUS et A. GUINET, A constraint hybrid flowshop problem. In *Workshop on Production Planning and Control (WPPC)*, septembre 1996, Mons, Belgique, FUCaM/EIASM/ICM, p. 303-306.

[ho-1974] W. A. HORN, Some simple scheduling algorithms, *Naval Research Logistics Quarterly*, 1974, *21*, p. 177-185.

[hu-1992] J. L. HUNSUCKER et J. R. SHAH, Performance of priority rules in a due date flow shop, *OMEGA*, 1992, *20*, n° 1, p. 73-89.

[hu-1994] J. L. HUNSUCKER et J. R. SHAH, Comparative performance analysis of priority rules in a constrained flow shop with multiple processors environment, *European Journal of Operational Research*, 1994, *72*, p. 102-114.

[ig-1965] E. IGNALL et L. SCHRAGE, Application of the branch and bound technique to some flow-shop scheduling problems, *Operations Research*, 1965, *13*, n° 3, p. 400-412.

[jo-1954] S. M. JOHNSON, Optimal two and three stage production schedules with setup times included, *Naval Research Logistics Quarterly*, 1954, *1*, n° 1, p. 61-68.

[ka-1997] S. N. KADIPASAOYLU, W. XIANG et B. M. KHUMAWOLA, A comparison of scheduling rules in static and dynamic hybrid flow systems, *Int. J. Prod. Res.*, 1997, *35*, n° 5, p. 1359-1384.

[la-1978] B. J. LAGEWEG, J. K. LENSTRA et A. H. G. RINNOOY KAN, A general bounding scheme for the permutation flow-shop problem, *Operations Research*, 1978, *26*, n° 1, p. 53-67.

[la-1987] M. A. LANGSTON, Interstage transportation planning in the deterministic flow-shop environment, *Operations Research*, 1987, *35*, n° 4, p. 556-564.

[la-1994] J.-B. LASSERRE et S. DAUZÈRE-PÉRÈS, Planification et ordonnancement intégrés dans un atelier de type jobshop, Bulletin de liaison n° 8 du CNRS/GdR Automatique/Ple SED/GT3, avril 1994.

[la-1982] R. LAUMAILLE, *Pratique de Gestion de Production*, Les Editions d'Organisation, 1982.

[la-1989] E. L. LAWLER, J. K. LENSTRA, A. H. G. RINNOOY KAN et D. B. SHMOYS, Sequencing and scheduling: algorithms and complexity, Report BS-R8909, Center for Mathematics and Computer Science, The Netherlands, Amsterdam, 1989, 70 p.

[lee-1993] C. LEE, T. C. E. CHENG et B. M. T. LIN, Minimizing the makespan in the 3-machine assembly flowshop scheduling problem, *Management Science*, 1993, *39*, n° 5, p. 616-625.

[lee-1994] C. Y. LEE et G. L. VAIRAKTARAKIS, Minimizing makespan in hybrid flowshop, *Operations Research Letters*, 1994, *16*, n° 3, p. 149-158.

[le-1992] A. LEGAIT, L'intégration : quels impacts sur la spécification d'un système de gestion de production assistée par ordinateur ?, *PhD thesis*, Université de Lyon I-INSA, 1992.

[le-1979] P. LEMOAL et J. C. TARONDEAU, Un défi à la fonction de la production, *Revue Française de Gestion*, Janvier-Février 1979, p. 9-14.

[le-1994] M.-L. LEVY, P. LOPEZ et B. PRADIN, Characterization of feasible schedules for the flow-shop problem: a decomposition approach. In *European Workshop on Integrated Manufacturing Systems Engineering (IMSE'94)*, Grenoble (France), INRIA, décembre 1994, p. 307-315.

[li-1996] S. LI, A hybrid two-stage flowshop with part family, batch production, major and minor setups, *European Journal of Operational Research*, 1996 à paraître.

[ma-1993] B. L. MACCARTHY et J. LIU, Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling, *Int. J. Prod. Res.*, 1993, *31*, n° 1, p. 59-79.

[na-1959] R. MACNAUGHTON, Scheduling with deadlines and loss functions, *Management Science*, 1959, *6*, p. 1-12.

[mi-1959] L. G. MITTEN, Sequencing n jobs on two machines with arbitrary time lags, *Management Science*, 1959, *5*, n° 3, p. 293-298.

[mo-1983] C. L. MONMA et A. H. G. RINNOOY KAN, A concise survey of efficiently solvable special cases of the permutation flow-shop problem, *RAIRO Rech. Opér.*, 1983, *17*, n° 2, p. 105-119.

[na-1987] S. L. NARASIMHAN et P. M. MANGIAMELI, A comparison of sequencing rules for a two-stage hybrid flowshop, *Decision Sciences*, 1987, *18*, p. 250-265.

[na-1984] S. L. NARASIMHAN et S. S. PANWALKAR, Scheduling in a two stage manufacturing process, *Int. J. Prod. Res.*, 1984, *22*, n° 4, p. 555-564.

[na-1983] M. NAWAZ, E. ENSCORE et I. HAM, A heuristic algorithm for the m machine, n job flowshop sequencing problem, *OMEGA*, 1983, *11*, n° 1, p. 91-95.

[pa-1961] S. PAGE, An approach to the scheduling of jobs on machines, *J. Roy. Statist. Soc.*, 1961, *23*, p. 484-492.

[pa-1979] R. J. PAUL, A production scheduling problem in the glass-container industry, *Operations Research*, 1979, *27*, n° 2, p. 290-302.

[po-1996] M. C. PORTMANN, A. VIGNIER, D. DARDILHAC et D. DEZALAY, Some hybrid flowshop scheduling by crossing branch and bound and genetic algorithms. In *5th International Workshop on Project Management and Scheduling (PMS'96)*, Poznan (Pologne), avril 1996, p. 186-189.

[po-1988] M.-C. PORTMANN, Méthodes de décomposition spatiales et temporelles en ordonnancement de la production, *RAIRO-APII*, 1988, *22*, n° 5, p. 439-451.

[pr-1995] C. PROUST et E. GRUNENBERGER, Planification de production dans un contexte de flow-shop hybride à deux étages : conception et interprogrammation d'ARIANNE 2000, *RAPA*, 1995, *8*, n° 5, p. 715-734.

[pr-1992] C. PROUST, Using Johnson's algorithm for solving flowshop scheduling problems. In *Summer school on Scheduling Theory and its Applications*, Bonas (France), INRIA/C^3/COMETT, octobre 1992, p. 297-342.

[ra-1992b] C. RAJENDRAN et D. CHAUDHURI, A multi-stage parallel processor flowshop problem with minimum flowtime, *European Journal of Operational Research*, 1992, *57*, p. 111-122.

[ra-1992a] C. RAJENDRAN et D. CHAUDHURI, Scheduling in n-jobs, m stage flowshop with parallel processors to minimize makespan, *Int. J. of Production Economics*, 1992, *27*, p. 137-143.

[ri-1995] P. RICHARD, C. CAVALIER, N. JACQUET et C. PROUST, Solving scheduling problems using petri nets and constraints logic programming. In *International Conference on Emerging Technologies and Factory Automation (ETFA'95), Vol. 1*, Paris (France), INRIA/IEEE, octobre 1995, p. 59-68.

[ri-1976] A. H. G. RINNOOY KAN, Machine scheduling problems: classification, complexity and computations, Nijhoff, The Hague, 1976.

[sa-1973] M. S. SALVADOR, A solution to a special class of flow shop scheduling problem. In *Symposium on the Theory of Scheduling and its Applications*, Springer-Verlag, Berlin (Allemagne), 1973, p. 83-91.

[sa-1992] E. SANLAVILLE, Conception et analyse d'algorithmes de liste en ordonnancement préemptif, *PhD thesis*, Université de Paris VI, september 1992.

[sa-1995] D. L. SANTOS, J. L. HUNSUCKER et D. E. DEAL, Global lower bounds for flow shop with multiple processors, *European Journal of Operational Research*, 1995, *80*, p. 112-120.

[saw-1993] T. J. SAWIK, A scheduling algorithm for flexible flow lines with limited intermediate buffers, *Applied Stochastic Models and Data Analysis*, 1993, *9*, p. 127-138.

[saw-1995] T. J. SAWIK, Scheduling flexible flow lines with no in-process buffers, *Int. J. Prod. Res.*, 1995, *33*, n° 5, p. 1357-1367.

[sh-1972] V. Y. Shen et Y. E. Chen, A scheduling strategy for the flowshop problem in a system with two classes of processors. In *Conference on Information and Systems Science*, 1972, p. 645-649.

[sh-1990] H. D. Sherali, S. C. Sarin et M. S. Kodialam, Models and algorithm for a two-stage production process, *Production Planning and Control*, 1990, *1*, n° 1, p. 27-39.

[sr-1989] C. Sriskandarajah et S. P. Sethi, Scheduling algorithms for flexible flow-shops: Worst and average case performance, *European Journal of Operational Research*, 1989, *43*, p. 143-160.

[st-1990] E. F. Stafford et F. T. Tseng, On the Srikar-Ghosh MILP model for the nxm SDST flowshop problem, *Int. J. Prod. Res.*, 1990, *28*, n° 10, p. 1817-1830.

[su-1983] D. R. Sule et K. Y. Huang, Sequency on two and three machines with setup, processing and removal times separated, *Int. J. Prod. Res.*, 1983, *21*, n° 5, p. 723-732.

[sz-1987] W. Szwarc et J. N. D. Gupta, A flow-shop problem with sequence-dependant additive setup times, *Naval Research Logistics Quarterly*, 1987, *23*, p. 619-627.

[sz-1983] W. Szwarc, Flowshop problems with time lags, *Management Science*, 1983, *29*, n° 4, p. 477-481.

[to-1977] W. Townsend, Sequencing n jobs on m machines to minimize maximum tardiness: a branch and bound solution, *Management Science*, 1977, *23*, n° 9, p. 1016-1019.

[va-1994] A. Vandevelde, Minimizing the makespan in a multiprocessor flow shop, *Master's thesis*, avril 1994, 37 p.

[vi-1995] A. Vignier, J.-C. Billaut et C. Proust, Les problèmes de flow-shop hybride : état de l'art, Rapport Interne 155, LI/E3i/Univ. de Tours, mai 1995, 100 p.

[vi-1996a] A. Vignier, J.-C. Billaut et C. Proust, Solving k-stage hybrid flowshop scheduling problems. In *Multiconference on Computational Engineering in Systems Applications (CESA'96), Symposium on Discrete Events and Manufacturing Systems (IEEE-SMC /IMACS)*, Lille (France), 1996, p. 250-258.

[vi-1996b] A. Vignier, J.-C. Billaut, C. Proust et V. Tkindt, Resolution of some two stage hybrid flowshop scheduling problems, Pékin (Chine), IEEE/Systems, Man and Cybernetics (SMC'96), octobre 1996, *4*, p. 2934-2941.

[vi-1996c] A. Vignier, D. Dardilhac, D. Dezalay et C. Proust, A branch and bound approach to minimize the total completion time in a k-stage hybrid flowshop. In *5th Internationnal Conference on Emerging Technologies and Factory Automation (ETFA'96)*, Hawaii (USA), IEEE/SICE, novembre 1996, *1*, p. 215-220.

[wi-1985] R. J. Wittrock, Scheduling algorithms for flexible flow lines, *IBM J. Res. Develop.*, juillet 1985, *29*, n° 4, p. 401-412.

[wi-1988] R. J. Wittrock, An adaptable scheduling algorithm for flexible flow line, *OpSearch*, 1988, *36*, n° 3, p. 445-453.