Contribution à l'étude des problèmes d'ordonnancement flowshop avec contraintes supplémentaires : Complexité et méthodes de résolution

Ammar Oulamara

Nancy-Université
INPL

Institut National
Polytechnique de Lorraine

Département de formation doctorale en informatique

École doctorale IAEM Lorraine

Contribution à l'étude des problèmes d'ordonnancement flowshop avec contraintes supplémentaires : Complexité et méthodes de résolution

MÉMOIRE

présenté et soutenu publiquement le 24 septembre 2009

pour l'obtention de l'

Habilitation à Diriger des Recherches de l'Institut National Polytechnique de Lorraine

(spécialité informatique)

par

Ammar OULAMARA

Composition du jury

Rapporteurs : Jacques CARLIER, Professeur, l'UTC de Compiègne
Stéphane DAUZÈRE-PÉRÈS, Professeur, Ecole des Mines de St-Etienne
Federico DELLA CROCE, Professeur, Polytechnique de Turin

Examinateurs : Gerd FINKE, Professeur émérite, G-SCOP/INPG, Grenoble
Alain GUINET, Professeur, l'INSA de Lyon
Marie-Claude PORTMANN, Professeur, INPL, Nancy
Jean-Yves MARION, Professeur, INPL, Nancy

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Contribution à l'étude des problèmes d'ordonnancement flowshop avec contraintes supplémentaires : Complexité et méthodes de résolution

Ammar OULAMARA
Laboratoire LORIA UMR 7503
Maître de Conférences
INPL - Ecole des Mines de Nancy

Habilitation à diriger des recherches de l'INPL
Discipline : Informatique

Table des matières

Première partie

Synthèse scientifique

Avant-propos

Mes travaux de recherche sont entièrement consacrés aux structures combinatoires et à l'optimisation dans les systèmes de production et de services. Après ma thèse de doctorat soutenue en juin 2001 au laboratoire Leibniz de Grenoble sur les problèmes d'ordonnancement dans un atelier de type flowshop, puis un passage éclair (moins de 10 mois) dans l'entreprise TempoSoft (spécialisée dans la planification de personnel) comme ingénieur en recherche et développement, j'ai intégré l'Institut National Polytechnique de Lorraine en tant que maître de conférence. En recherche j'ai rejoint le projet MACSI (Modélisation, Analyse et Conduite des Systèmes Industriels) de l'INRIA Lorraine et du LORIA (Laboratoire Lorrain de Recherche en Informatique et ses Applications), projet bi-localisé à Nancy et à Metz, cela jusqu'en décembre 2006 date de fin du projet. Les activités de MACSI se positionnaient dans le cadre de la conception et de la conduite des systèmes de production. Les bases théoriques sont celles des systèmes à événements discrets et de l'optimisation discrète. Les activités de recherche au sein de MACSI étaient organisées suivant trois axes complémentaires :

1. Modélisation et spécification des systèmes industriels comprenant la modélisation en entreprise (modèles descriptifs), la modélisation comportementale (modèles analytiques) et la synthèse de commande.

2. Evaluation des performances et dimensionnement des systèmes à événements discrets stochastiques, en utilisant à la fois les méthodes analytiques pour des systèmes particuliers et des méthodes génériques d'optimisation pour les systèmes généraux.

3. Organisation et gestion de la production, avec comme spécialités : l'ordonnancement prédictif et réactif, le pilotage, l'agencement ainsi que l'étude de politiques de maintenance dans les ateliers de production.

Puis à partir de janvier 2007 avec mes collègues de Nancy, et suite à nos travaux dans MACSI, nous avons monté l'équipe de recherche ORCHIDS (Operations Research for Complex Hybrid Decision Systems) du laboratoire LORIA. L'objectif principale de l'équipe est de concevoir des méthodes et des outils d'aide à la décision efficaces et performants pour des systèmes ou des réseaux complexes en s'appuyant tout particulièrement sur les outils et les modèles de la recherche opérationnelle. Le champ d'application principal est la chaîne logistique (production, stockage, transport, etc.) et dans une perspective plus générale tout réseau où circulent des flux (informations et produits), l'objectif étant l'optimisation et l'organisation des supports physiques et/ou leur pilotage. Les activités de recherche de la jeune équipe ORCHIDS sont organisées suivant deux axes, à savoir,

1. Modélisation et optimisation des unités de production et des systèmes de transports (agence-

ment d'ateliers, localisation de sites, planification et ordonnancement des tâches de production, de maintenance et de transport) en intégrant des contraintes industrielles pertinentes.

2. Modèles et outils pour l'organisation de la négociation et la coopération au sein des chaînes logistiques et des systèmes complexes.

Les fondements scientifiques de l'équipe sont principalement issue de la recherche opérationnelle et de l'optimisation. Ce recentrage est voulu par les membres de l'équipe pour constituer un noyau visible de la thématique RO.

Mes travaux de recherche concernent essentiellement l'organisation et la gestion de production, particulièrement l'optimisation des systèmes de production du point de vue de l'ordonnancement. Ils sont axés d'une part sur la recherche de nouveaux résultats fondamentaux, et d'autre part sur l'étude des problématiques industrielles. Ces résultat sont aussi le fruit de collaboration avec plusieurs personnes auxquelles je souhaite exprimer ici ma gratitude. Je tiens à remercier chaleureusement les thésards que j'ai co-encadré, julien fondrevelle, Zerouk Mouloua et actuellement Adrien Bellanger, avec lesquels les discussions été toujours fructueuses. J'ai eu également la change de travailler avec des personnes talentueuses passionnées par la recherche, particulièrement je remercie, dans l'ordre alphabétique Ali. Allahverdi, Gerd. Finke, Michail. Kovalyov, Wieslaw Kubiak, Djamal Rebaine, Marie-Claude Portmann, Ameur Soukhal. Enfin je termine cet avant-propos par un remerciement aux collègues avec qui je discutais souvent des problématiques intéressantes, lors de nos rencontres dans des congrès nationaux ou internationaux.

1

Introduction

1.1 Introduction

Le processus d'ordonnancement occupe une place importante dans l'industrie et les services. Il s'attache à l'allocation des ressources aux tâches sur des périodes données de temps, et dont l'objectif est d'optimiser un ou plusieurs critères. Les ressources et les tâches peuvent prendre différentes formes en pratique. Les ressources peuvent être des machines, des ouvriers ou tout simplement des outils de fabrication dans un atelier de production, elles peuvent être des unités de calcul ou des serveurs dans un environnement informatique. Les tâches sont des opérations sur des produits à fabriquer sur les machines dans un système de production, elles peuvent être des programmes à exécuter ou simplement des requêtes à transmettre sur des serveurs dans un environnement informatique. Quelque soit le domaine d'application, l'ordonnancement contribue à l'élaboration de meilleures stratégies à adopter pour l'optimisation et l'utilisation des ressources.

Dans l'industrie manufacturière, les modules d'ordonnancement sont associés aux outils de gestion de production, et sont intégrés dans des logiciels de type ERP (*Entreprise Resource Planning*) comme SAP ou JD Edouards, sous forme de modules dits APS (*Advanced Planning Systems*). Les outils APS sont classés par domaine d'applications, comme l'industrie d'assemblage (automobile, engins de genie civile, etc.). Il est tout à fait certain qu'un même APS peut difficilement se révéler efficace pour des applications très différentes, à cause des configurations particulières des unités de production, de certaines caractéristiques techniques des produits ou de certaines contraintes de production plus ou moins complexes. Ainsi, un ordonnancement éventuellement élaboré par un APS, est généralement corrigé manuellement par le service de contrôle de production. A cause de ce faible potentiel des APS actuels, des logiciels plus spécialisés en ordonnancement (comme Ortems, Incoplan, Ordosoftware, etc.) ont pu se faire une place sur le marché des progiciels. Ceci, en développant des outils plus élaborés qui tiennent compte de situations industriels complexes, et aussi grâce à leurs canaux de dialogue avec les ERPs. Evidement la prise en compte de nouvelles contraintes industrielles dans ces logiciels passe préalablement par l'étude et la modélisation de ces contraintes pour pouvoir proposer des méthodes efficaces d'aide à la décision. Les entreprises proposant ces logiciels passent souvent par des partenariats

avec le monde universitaire[1], comme le montre le nombre croissant de thèses CIFRE dans le domaine.
Dans ce mémoire, nous nous intéressons aux problèmes d'ordonnancement dans les systèmes de production. Dans la littérature classique de l'ordonnancement[2], les problèmes sont classés par familles de problèmes selon la structure de l'atelier, on distingue :

- Atelier à une machine : l'atelier est composé soit d'une seule machine (un bloc/outil de production) ou l'essentiel de l'atelier peut être représenté par une seule machine (machine goulot par exemple). Pour ce modèle chaque tâche est constituée d'une seule opération qui doit être effectuée sur cette machine.
- Atelier à plusieurs machines parallèles : chaque tâche est constituée d'une seule opération qui peut être effectuée par plusieurs machines. Les machines peuvent être identiques ou différentes. Pour ce type d'ateliers, en plus du problème de succession des tâches sur les machines, il faut aussi décider de leur affectation sur les ressoures.
- Atelier flowshop : les machines sont disposées en ligne et toutes les tâches ont la même gamme opératoire, c'est-à-dire, chaque tâche doit passer sur chaque machine et cela dans le même ordre pour toutes les tâches.
- Atelier jobshop : les machines sont disposées en ligne, mais chaque tâche a sa propre gamme opératoire, c'est-à-dire, chaque tâche dispose de son propre ordre de passage sur les machines.
- Atelier openshop : ici la numérotation des machines est immatériel. La gamme de fabrication des tâches n'est pas fixée à l'avance, elle est déterminée lors du traitement des tâches.
- Atelier flexible : l'atelier peut être de type flowshop ou jobshop, mais chaque étage est composé de plusieurs machines parallèles identiques ou différentes.

Indépendamment de la structure de l'atelier, les problèmes d'ordonnancement sont également caractérisés par les critères à optimiser. Ces critères sont exprimés sous forme de fonctions mathématiques de type min-max ou mini-somme. Le critère le plus considéré dans la littérature reste de loin la durée total de l'ordonnancement ou le makespan en anglais. Il consiste à minimiser la date de fin de traitement de la dernière tâche à réaliser. Ce critère est très utile en pratique, à titre d'exemple, pour un fonctionnement en "travail posté" d'un atelier de production, il est toute à fait naturel de minimiser la date de fin de fabrication des OFs. Un autre critère important dans les ateliers de production est le flot total ou la somme des dates de fin des tâches, ce critère permet de diminuer les encours de production et de réduire les stocks. Par ailleurs, d'autres critères basés sur la notion du retard des tâches sont utilisés si les tâches sont soumises à des échéances, on trouve par exemple le critère du plus grand retard, le critère de la somme des retards, et le critère du nombre de tâches en retard. Pour les critères sous forme de sommes, il est aussi possible d'attribuer à chaque tâche un poids pour signifie son importance ou son urgence par rapport aux autres tâches, l'objectif est alors de minimiser la somme pondéré.

[1] Exemple de la thèse CIFRE de Fredy Deppner de notre équipe de recherche avec la société INCOTEC, celle-ci a conduit à la mise au point d'une approche de résolution adaptée aux problèmes industriels rencontrés dans la pratique, que ce soit au niveau de la taille des données à traiter ou au niveau du nombre et de la diversité de contraintes incorporées dans le modèle.

[2] Nous parlons ici des problèmes classiques d'ordonnancement les plus considérés et étudiés dans la littérature, le lecteur intéressé à d'autres problèmes d'ordonnancement avec diverses applications hors ateliers de production peut consulter le livre complet sur l'ordonnancement *'Handbook of Scheduling : Algorithms, Models, and Performance Analysis'* édité par J. Leung, et publié par CRC Press, Boca Raton, FL, USA, 2004.

Les modèles les plus répondus dans l'industrie sont les ateliers de type flowshop ou jobshop ou ateliers flexibles, avec des contraintes très complexes de fabrication. Dans la majorité de nos travaux de recherche sur l'ordonnancement, nous avons pris en compte de nouvelles contraintes industrielles ou des contraintes qui reflètent le mieux possible des situations industrielles. Ainsi, à chaque nouvelle problématique étudiée, la démarche adoptée consiste à modéliser, éventuellement simplifier, le problème, puis nous vérifions si cette modélisation s'apparente à un ordonnancement déjà répertorié dans la littérature. Dans le cas négatif nous nous appliquons à l'étude de la complexité du problème pour pouvoir orienter le type de méthodes de résolution à utiliser. Notre démarche de recherche évolue selon deux directions, d'une part nous privilégions la recherche de nouveaux résultats fondamentaux, d'autre part nous construisons des méthodes de résolution efficaces et rapides qui peuvent se mesurer à la taille réelle des instances industrielles.

1.2 Orientation de l'activité de recherche

L'orientation de l'activité de recherche ainsi que le choix des thèmes étudiés ou à étudier est souvent dictée par des rencontres industrielles et académiques. Dans notre cas, ce choix est généralement guidé par le souci de répondre d'une manière positive à des demandes industrielles ou de contribuer à recherche de réponses aux questions fondamentales sur des problèmes difficiles.

Dans ce mémoire, j'ai choisi de distinguer trois thèmes. Les résultats[3] obtenus pour chaque thème dépendent à la fois de la difficulté des problématiques étudiées, du temps qui leur est imparti et des circonstances et des opportunités d'encadrement des étudiants.

Comme il est mentionné dans la section précédente, mes travaux de recherche en ordonnancement sont principalement axées sur les ateliers de type flowshop avec prise en compte de contraintes supplémentaires, proche de la réalité inustrielle. Je souhaite détailler ici mes travaux sur les contraintes suivantes : (i) prise en compte de contraintes de groupement des tâches, connues sous le terme anglais, *batch scheduling*, (ii) prise en compte de contraintes temporelles sur la succession d'exécution des tâches, connues sous le nom de *time-lags*, (iii) prise en compte de la détérioration des tâches.

L'ordre de description des thèmes n'est nullement arbitraire, en fait, il reflète l'importance des thèmes et les résultats obtenus. La prise en compte de contraintes de groupement des tâches est un vaste thème en ordonnancement, du fait des applications sous-jacentes et de la diversité des problèmes. Ce thème fait suite à mes travaux de thèse. Le second thème constitue un autre axe de mes travaux de recherche, initié par le co-encadrement de la thèse de Julien Fondrevelle sur ce sujet. L'étude de ce thème a été initiée par les collaborations avec INCOTEC et, est motivée par le nombre important de problèmes qui peuvent être modélisés par des contraintes temporelles. Le troisième thème traite des problèmes de détérioration des tâches, notamment avec des application dans la sidérurgie[4]. Notre contribution à ces trois thèmes concerne d'une part l'étude de la complexité de la structure combinatoire de ces problèmes, et d'autre part la mise en oeuvre de méthodes d'optimisation efficaces pour la résolution.

Après cette brève introduction, le reste de ce mémoire est organisé selon les types de résultats recherchés et obtenues. Ce mémoire est décomposé en six chapitres. Le chapitre 2 présente les

[3] Afin de faciliter la lecture de ce document, on indiquera par une police différente comme ici, nos résultats de recherche par rapport à ceux existant dans la littérature.

[4] Le détail des applications et des différents modèles sont présentés dans le chapitre 2.

trois thèmes (problématiques) étudiés, pose les contraintes spécifiques à chaque problématique, décrit les applications sous-jacentes, et présente les enjeux pratiques ainsi que les critères utilisés pour l'optimisation. Le chapitre 2 concerne l'étude de la complexité des problèmes d'ordonnancement d'une manière générale, et de la complexité des problèmes particuliers présentés dans ce mémoire. Pour chaque thème, nous rappelons les différents résultats existants dans la littérature, et nous mettons l'accent sur les nouveaux résultats que nous avons obtenus pour les différents problèmes. Le chapitre 4 est consacré aux problèmes polynomiaux ainsi qu'aux méthodes de resolution avec guarantie de performances et les schémas d'approximation pour les problèmes difficiles. Le chapitre 5 expose le schéma général de construction de méthodes exactes de type séparation et évaluation pour les problèmes traités, ainsi que les résultats obtenus par ces méthodes. Le chapitre 6 clôt ce mémoire par une conclusion générale, ainsi que les perspectives et les orientations de recherche que nous souhaitons engagé dans un avenir proche ainsi que quelques réflexions sur de nouvelles voies de recherche.

2

Problèmes, applications et aperçu de la littérature

2.1 Introduction

Dans ce chapitre, nous présentons les différentes contraintes prises en compte dans les problèmes d'ordonnancement traités dans ce mémoire. Les sections 2, 3 et 4 sont respectivement dédiées aux contraintes de groupement des tâches, contraintes d'écarts temporels et contraintes de détérioration de tâches. Pour chaque section, nous présentons l'intérêt de ces contraintes, les différents modèles existants ainsi que les applications associées, puis nous terminons chaque section par un aperçu de la littérature.

2.2 Groupement de tâches

L'étude des problèmes d'ordonnancement avec groupement de tâches occupe une large place dans la littérature. L'intérêt suscité par ces problèmes est dû aux nombreuses applications dans l'industrie et les services. Le papier *"Scheduling with batching : A review"* publié par Potts et Kovalyov [130] dans EJOR (*European Journal of Operational Research*) a été sélectionné en 2007 comme l'un des 30 articles les plus influants publiés dans ce journal entre 1975 et 2005, ceci pour son intérêt pour la communauté scientifique.

La finalité principale du groupement des tâches (traitement par batches ou *batching* en anglais) réside dans le taux d'efficacité du traitement des tâches, et le gain en temps. Par exemple, pour réduire la facture énergétique dans la sidérurgie, la température des fours est amplifiée après avoir rempli ces fours de produits. Egalement, il peut être intéressant de regrouper les tâches, lorsque les machines nécessitent un temps de nettoyage ou de changement d'outils à chaque exécution, par exemple, dans les ateliers de peinture dans l'industrie automobile (les véhicules de même couleur sont alignés les uns derrière les autres), dans ce cas, il est utile de réduire la somme des temps de changement en regroupant l'exécution des tâches en famille (similitude entre les tâches).

On distingue deux grandes familles de machines à traitement par batches, à savoir les max-batch machines (*parallel batching machine*), notées par *p-batch machines* et les sum-batch machines

(*serial batching machines*), notées *s-batch machines*.

2.2.1 p-batch machine

Une p-batch machine peut traiter plusieurs tâches simultanément, selon sa capacité, dans un même batch. Quand la capacité de la p-batch machine est suffisante pour traiter toutes les tâches simultanément, on dit que la capacité est infinie, si cette capacité n'est pas suffisante, on dit que la capacité est finie. Cette capacité s'exprime soit en nombre de tâches à exécuter simultanément soit selon une grandeur physique telle que le poids ou le volume des tâches. Les tâches peuvent être identiques (mêmes durées opératoire) ou non identiques (durées opératoires différentes). La figure 2.1 présente les différentes caractéristiques des tâches dans les modèles d'ordonnancement en présence de p-batch machines. La durée opératoire d'un batch sur une p-batch machine est égale à la plus grande durée opératoire des tâches appartenant à ce batch. Les tâches d'un même batch ne sont disponibles que lorsque le batch est intégralement traité.

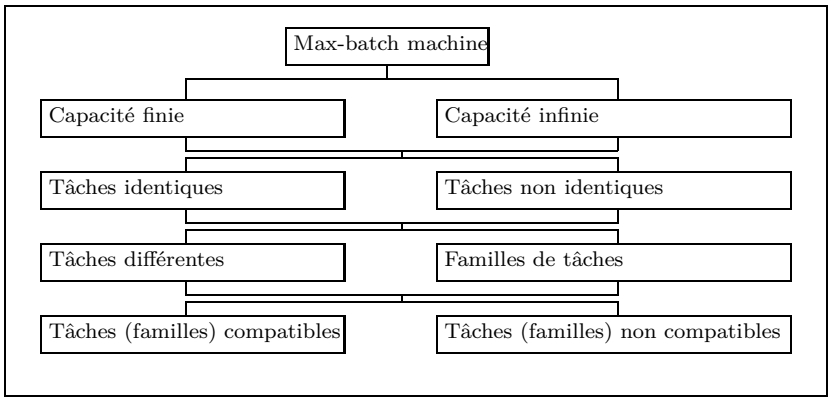

FIG. 2.1 – Caractéristiques des modèles sur une p-batch machine

Des applications diverses des p-batch machines sont décrites dans la littérature, par exemple, dans l'industrie des semi-conducteurs, Ahmadi et al. [3], dans la sidérurgie, Finke et al. [57], dans l'industrie du pneumatique, Oulamara et al. [126]. Dans certaines situations, les tâches regroupées dans le même batch doivent être compatibles, c'est-à-dire, elles doivent partager les mêmes propriétés techniques pour pouvoir être traités ensemble. La compatibilité entre les tâches est alors représentée par un graphe de compatibilité. Boudhar [21], [19] a considéré le cas d'un graphe général de compatibilité et quelques graphes particuliers. Finke et al. [57] ont présenté deux exemples sur la nécessité de compatibilité entre les tâches, le premier concerne l'utilisation des fours en sidérurgie et le deuxième concerne la fabrication d'équipements (métalliques) de bureaux. Par ailleurs, Oulamara et al. [126] ont présenté une application dans l'industrie du pneumatique, où la compatibilité, cette fois-ci, est associée aux durées opératoires des tâches.

2.2.2 s-batch machine

Une s-batch machine peut traiter plusieurs tâches en séquentiel. Les tâches peuvent être différentes ou regroupées en familles [153]. Sur ce type de machine, un temps de réglage est nécessaire chaque fois qu'on change de tâche si toutes les tâches sont différentes, ou seulement entre deux

tâches de familles différentes si les tâches sont regroupées en familles. Dans le cas où les tâches sont regroupées en plusieurs familles, le temps de changement peut être, soit indépendant des tâches à réaliser, ou bien il dépend, soit de la tâche réalisée, soit de la tâche qui va être réalisée, soit des deux tâches à la fois. Même quand toutes les tâches forment une seule famille, il est possible d'exécuter cette famille de tâches en plusieurs batches, alors un temps de changement est nécessaire sur la machine avant l'exécution de chaque batch, ce problème est dit *batch sizing problem*, Coffman et al. [49]. La figure 2.2 présente les différentes caractéristiques des tâches dans les modèles d'ordonnancement en présence de s-batch machines.

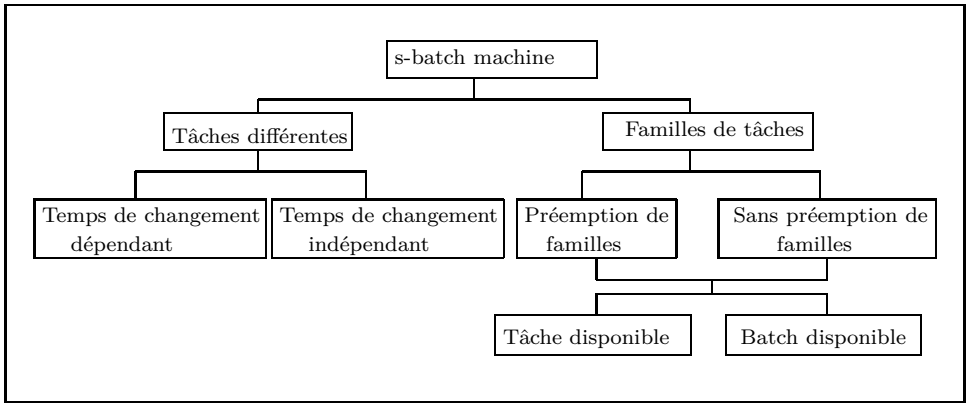

FIG. 2.2 – Caractéristiques des modèles sur une s-batch machine

La durée opératoire d'un batch sur une s-batch machine est égale à la somme des durées opératoires des tâches appartenant à ce batch. Une tâche de ce batch, est soit disponible après sa réalisation alors le modèle est dit *tâche disponible*, soit disponible seulement après la réalisation de toutes les tâches du batch alors le modèle est dit *batch disponible*. Par exemple, on considère un ensemble de tâches placé sur un support (palette) et transporté par un robot, ici l'exécution d'une tâche consiste à la charger sur la machine et à la reposer sur le support après sa réalisation. Une tâche n'est disponible pour le traitement sur les autres machines que lorsque toutes les tâches du même support sont réalisées. Dans cet exemple, un batch représente l'ensemble des tâches posées sur le même support.

Le temps de changement sur la s-batch machine peut être sans anticipation ou avec anticipation. Dans le premier cas, noté *'ns' nonanticipatory setup time*, l'exécution du temps de changement nécessite la présence du batch sur la machine, alors que dans le second cas, noté *'as' anticipatory setup time*, la présence du batch n'est pas nécessaire, Baker [11]. Ces deux modèles ont une signification, par exemple, quand les tâches ont des dates d'arrivée différentes, dans ce cas le modèle *'ns'* impose au *setup time* du batch contenant ces tâches à ne commencer que lorsque toutes ses tâches soient disponibles, alors que le modèle *'as'* s'affranchit de cette contrainte. Une autre différence entre ces deux modèles peut être observée dans les problèmes d'ateliers, par exemple, dans un flowshop à deux s-batch machines, un temps de changement est dit avec anticipation sur la deuxième machine, s'il peut être exécuté avant que le batch correspondant soit disponible sur cette machine, et il est dit sans anticipation si l'exécution de ce temps de changement ne peut être effectué avant que le batch soit disponible sur la deuxième machine (voir les figures 2.3 et 2.4). Les s-batch machines sont, par exemple, utilisées dans l'industrie

plastique [131].

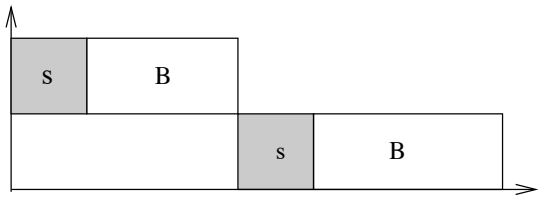

FIG. 2.3 – Temps de changement sans anticipation

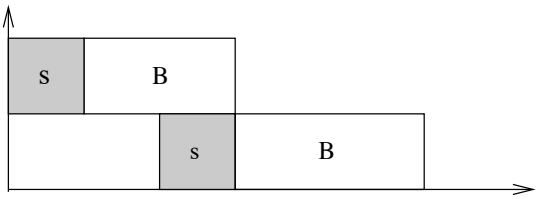

FIG. 2.4 – Temps de changement avec anticipation

2.2.3 Etat-de-l'art

Plusieurs problèmes d'ordonnancement intégrants les p-batch et les s-batch machines sont étudiés dans la littérature. Des état-de-l'arts sur ces problèmes ont été successivement présentés par Potts et Van Wassenhove [131] Wabster et Baker [153], Potts et Kovalyov [130].

Brucker et al. [23] ont étudié la complexité des problèmes d'ordonnancement sur une p-batch machine dans les cas où la capacité est respectivement, infinie et finie. Ils ont utilisé la programmation dynamique pour montrer que certains problèmes sont polynomiaux. Par exemple, lorsque la capacité de la p-batch machine est infinie, minimiser le flot total, le flot total pondéré et le retard algébrique sont polynomiaux avec une complexité de $O(n^2)$. Par ailleurs, les problèmes dont les critères sont le nombre de tâches en retard et le nombre pondéré de tâches en retard sont respectivement de complexité $O(n^3)$ et $O(n^2W)$, où W est la somme des pondérations de toutes les tâches. Quand la capacité de la p-batch machine est finie, Brucker et al. [23] ont développé un algorithme basé sur la programmation dynamique pour le critère du flot total avec une complexité de $O(n^{b(b-1)})$, où b représente la capacité de la p-batch machine, Poon et al. [128] ont amélioré ce résultat pour des grandes valeurs de b et ont proposé un algorithme en $O(n^{6b})$. Brucker et al. [23] ont aussi montré que les problèmes de minimisation du retard algébrique, du nombre de tâches en retard sont NP-difficiles au sens fort, alors que minimiser la somme pondérée des dates de fin des tâches reste un problème ouvert. D'autres auteurs [99], [149], [39], [158] se sont intéressés aux problèmes d'ordonnancement d'une p-batch machine avec d'autres contraintes supplémentaires. En présence de compatibilité entre les tâches, Boudhar et Finke [21] ont considéré le cas d'une p-batch machine avec un graphe de compatibilité général. Ils ont montré que si la capacité de la p-batch machine est égale à 2, la minimisation du makespan est polynomial et revient à un problème de couplage de poids maximum, le problème devient NP-difficile dès que la capacité de la p-batch machine est supérieure à 2. Boudhar a considéré d'autres structures de graphes, à savoir le graphe biparti [19], et le split graphe [18], [20]. Finke et al. [57] ont considéré un cas très intéressant où les comptabilités entre les tâches sont représentées par un graphe d'intervalle, ils ont présenté deux applications industrielles pour justifier le graphe d'intervalle, puis ils ont considéré plusieurs situations où la capacité de la p-batch machine est respectivement finie et infinie, durées opératoires des tâches respectivement unitaires et quelconque. Ils ont montré en particulier que la minimisation du makespan (capacité de la machine infinie et durées opératoires quelconques) est polynomial, et proposent un algorithme de type programmation dynamique en $O(n^3)$. La prise en compte de la compatibilité entre les

tâches est très liée à l'ordonnancement chromatique dans les graphes complémentaires (graphes d'incompatibilité), [52].

Pour le problème de p-batch machines parallèles, Brucker et al. [23] ont montré que pour tout critère régulier et sans restriction sur la capacité des machines, il existe une solution optimale où, sur chaque machine, les batches sont séquencés dans l'ordre croissant de leur durée opératoire. Ainsi, pour un nombre constant de machines, plusieurs algorithmes de programmation dynamique sur une machine peuvent être généralisés pour les problèmes à plusieurs machines.

Concernant les ordonnancements d'ateliers, Potts et al. [129] ont traité de la complexité de la minimisation du makespan dans les ateliers flowshop, jobshop et openshop en présence de deux p-batch machines. Quand la capacité des deux machines est infinie, ils ont montré que l'openshop est soluble en $O(n)$ où l'ordonnancement optimal contient au plus deux batches, alors que le jobshop est soluble en $O(nlogn)$, où la solution optimale contient au plus trois batches sur chaque machine. Si l'une des machines a une capacité finie, le problème d'openshop et du jobshop deviennent NP-difficiles. Dans le chapitre 3, nous reviendrons en détail sur les résultats sur le flowshop.

Les problèmes d'ordonnancement sur les s-batch machines sont traités dans la littérature dans le cadre de redimensionnement des lots de production, *lot-sizing ou batch sizing problem* d'une ou plusieurs familles de produits. Ici la taille des batches est une valeur discrète, à l'opposé du lot streaming où la taille des batches peut être continue, [43], [90]. La plupart des travaux sur une s-batch machine traitent les critères du flot total, du flot total pondéré, en présence d'une ou plusieurs familles de tâches. Nous décrivons ici un aperçu des travaux de la littérature sur une seule famille de tâches à ordonnancer en plusieurs batches. Les travaux traitant de plusieurs familles de tâches sont par exemple traités par Monma et Potts [106], Potts et Van Wassenhove [131]. Potts et Kovalyov [130] et Allahverdi et al. [8] ont successivement présenté des *surveys* incluant les travaux sur une s-batch machine et plusieurs familles de tâches à ordonnancer.

Coffman et al [49], [48] sont les premiers à avoir étudié le problème d'ordonnancement sur une s-batch machine. Dans [49], ils ont montré que la minimisation du flot total est polynomial et soluble en $O(nlgn)$. Albers et Brucker[5] ont montré que la somme pondérée des dates de fin des tâches est NP-difficile sauf quand les durées opératoires sont égales, où le problème est polynomial soluble en $O(n^2)$. Ils ont considéré aussi la minimisation du flot total en présence de contraintes de précédence. D'autres auteurs ont traité les problèmes d'ordonnancement sur une s-batch machine avec des contraintes supplémentaires et des critères d'optimisation différents, Hochbaum and Landy [82] ont considéré le critère du nombre de tâches en retard, Yuan et al. [159] ont traité le même problème avec des contraintes de précédence en plus, Baptiste [13] a examiné plusieurs critères réguliers en présence de dates d'arrivée et durées opératoires des tâches constantes, Gerodimos et al. [68] ont abordé le critère du nombre de tâches en retard, Ng et al. [119] ont examiné l'objectif du retard algébrique des tâches et des contraintes de précédence. Cheng et Kovalyov [40] ont analysé le cas spécial où la taille des batches est limité à k tâches, ils ont montré que la minimisation du flot total est polynomial, aussi ils ont étudié la complexité d'autres critères réguliers. Pour les s-batch machines parallèles, la majorité des problèmes d'ordonnancement sont NP-difficiles, du fait déjà, que le cas de machines classiques sont NP-difficile. Cheng et al. [36] ont proposé un algorithme de programmation dynamique en $O(m.n^{m+1})$ pour le critère du flot total, cet algorithme est basé sur les idées de Coffman et al. [49] pour une seule machine.

Pour les ateliers de production (flowshop, jobshop et openshop) les batches ont une autre caractéristique supplémentaire, à savoir la consistance et la nonconsistance. Un batch est dit *consistant*, si son contenu n'est pas modifié lors de son traitement sur les différentes machines, alors qu'il est *nonconsistant*, si le contenu du batch est remis en question à chaque traitement sur une machine. Glass et al. [71] ont étudié le problème de minimisation du makespan dans un openshop avec deux s-batch machines, ils ont montré qu'il existe une solution optimale composée de batches consistants. Ils ont aussi montré que la solution optimale contient un, deux ou trois batches. Malheureusement même avec ces deux propriétés qui caractérisent la structure de la solution optimale, la recherche de la solution optimale contenant deux batches est un problème NP-difficile au sens ordinaire. Gribkovskaia et al. [73] ont montré que pour le même problème, s'il existe une solution optimale avec trois batches alors elle peut être trouvée en $O(n)$. Glass et al. [71] ont étudié le problème du flowshop, nous revenons en détail sur ces problèmes dans le chapitre 3. Les ateliers de type jobshop en présence de s-batch machines sont très peu considérés dans la littérature. Mosheiov et Oron [113] ont proposé un algorithme en $O(n)$ pour la minimisation du makespan en présence de deux machines et des durées opératoires constantes. Sotskov et al. [144] ont considéré le cas de plusieurs machines et des critères à la fois réguliers et non-réguliers. Pour la résolution, ils ont proposé plusieurs techniques d'insertion combinées avec beam-search. D'autres travaux axés sur la résolution du job shop intégrant des contraintes supplémentaires sont publiées dans la littérature, Low et al. [102],

2.3 Contraintes d'écarts temporels

Les contraintes d'écarts temporels (*time lags* en anglais), connues aussi dans la littérature comme des contraintes de délais ou contraintes d'attentes, sont une généralisation des contraintes de précédence classiques. Dans le cas classique, une tâche peut débuter juste après la fin de la tâche qui la précède. La présence de contraintes d'écarts temporels oblige une tâche à ne commencer qu'après un certain temps après le début ou la fin de la tâche qui la précède. Dans ce cas, le temps d'attente est minoré, appelé aussi *time-lag min* (voir la figure 2.5). Il est parfois possible que la second tâche doit absolument commencer avant un temps donné après le début ou la fin de la première tâche, l'attente est alors majoré, appelé ici *time-lag max* (voir la figure 2.6). Le dernier cas regroupe les deux situations, où l'attente est à la fois minorée et majorée, dans ce cas l'attente se situe dans un intervalle de temps.

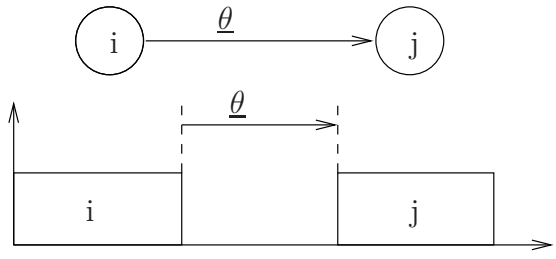

FIG. 2.5 – Contrainte avec time-lag min

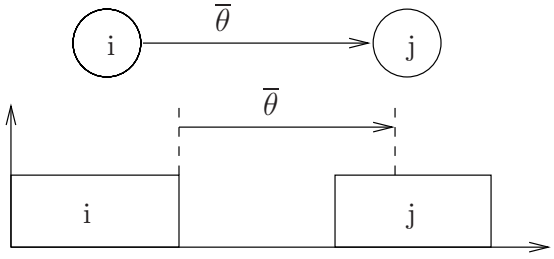

FIG. 2.6 – Contrainte avec time-lag max

Contrairement aux problèmes d'ordonnancement de projets, où depuis longtemps les contraintes temporels sont prises en compte, par exemple, les travaux de B. Roy dans les années soixante sur les contraintes de potentiel [139],[138] ; les contraintes de délais n'ont été intégrées que plus

tard dans les problèmes d'ordonnancement d'ateliers. Une première étude incluant les time-lags est dû à Mitten [105] en 1959, mais c'est à partir des années 80 que les premiers travaux sur les contraintes d'écarts temporels sont réellement apparus dans la littérature de l'ordonnancement sous différentes appellations, [141], [103], [142].

L'accroissement des travaux avec la prise en compte des time-lags est attribué aux nombreuses applications industrielles. Deppner [53] a présenté dans sa thèse une large liste de ces applications, Fondrevelle [60] a aussi présenté dans sa thèse d'autres applications industrielles et un état de l'art complet sur les modèles d'ordonnancement avec time-lags. Nous citons ici quelques applications, par exemple, dans l'agro-alimentaire et l'industrie du froid. Hodson et al. [83] ont exposé des contraintes de fabrication, où pour réduire le risque de contamination des produits alimentaires durant le processus de production, un temps maximal de 30 mn est imposé entre la cuisson des produits et leur surgélation. Dans l'agriculture, Foulds et Wilson [64] ont décrit les contraintes temporelles affectant l'enchaînement des différentes tâches lors de la récolte. Dans l'industrie chimique et pharmaceutique, Chu et Proth [46] ont présenté une application dans un laboratoire d'analyses médicales automatisé, où les time-lags sont imposés entre l'enchaînement des différentes opérations, ceci pour la stabilisation des produits due aux réactions chimiques. Dans la sidérurgie, Caumond et al. [31] ont modélisé la situation où certaines tâches sur des produits nécessitent une température précise, les time-lags correspondent au réchauffement ou au refroidissement entre les tâches. On trouve aussi des applications en génie civil, Bartusch et al. [14], Heilmann [81].

Depuis les années 90, plusieurs études publiées dans la littérature portent sur les problèmes d'ordonnancement d'ateliers. Les problèmes à une machine font un peu exception, en raison de la signification du time-lag ici. En fait, on distingue deux modèles pour une machine : (i) le modèle où une tâche est composée d'une seule opération à traiter, ici les time-lags sont plutôt utilisés pour généraliser les contraintes de précédence classiques entre les tâches, (ii) une tâche est composée d'un certain nombre d'opérations à réaliser sur une machine, par exemple, le problème de *coupled-task* [142], les opérations d'une même tâche sont liées entre elles par des relations de précédence sous forme de chaînes, les time-lags min et max sont définis entre les opérations de la même tâche.

Concernant le modèle à une machine avec contraintes de précédence, Wikum et al. [154] ont présenté la première étude sur la généralisation des contraintes de précédence classiques en incluant des time-lags min et max. Leurs papier expose essentiellement des résultats de complexité pour le cas de graphes sous forme de chaînes. La présence à la fois de time-lags min et max rend la majorité des problèmes NP-difficile, même dans le cas de simples chaînes composées de deux tâches. Brucker et Knust [25] ont montré de nombreux résultats de complexité pour le problème avec time-lags min, un résultat intéressant de leur papier, est la réduction des problèmes de machines parallèles avec préemption au problème à une machine et time-lags min, ils ont présenté quelques cas particuliers qui sont polynomiaux. Dans la même continuité de l'étude des problèmes de précédence de type chaînes, Munier et Sourd [116] ont étudié le cas de time-lags min, pour le critère de C_{max} et durées opératoires des tâches et time-lags min fixes, ils ont proposé un algorithme polynomial pour trouver la valeur de la solution optimale, mais pseudo-polynomial pour trouver l'ordonnancement optimal, cela est dû au codage des données. Brucker et al. [26] ont proposé d'autres résultats de complexité et complète l'étude du problème traité par Munier et Sourd [116] en fournissant un algorithme polynomial pour le critère de la durée totale de

l'ordonnancement. Pour les méthodes de résolution des problèmes à une machine, on trouve des travaux sur les méthodes exactes de type séparation et évaluation, Balas et al. [12], Lourenço [47] pour la prise en compte de time-lags min, Chu et Proth [46], Brucker et al. [24], Sheen et Liao [143] pour la prise en compte de time-lags min et max, Hurink et Keuchel [85] ont proposé une méthode de recherche local de type tabu pour le cas de time-lags min-max.

Pour le second modèle, où chaque tâche est composée de deux opérations, on trouve les travaux de Yu [156] et Yu et al. [157] sur le critère de makespan en présence de time-lags min. Ils ont montré que ce problème est NP-difficile même quand les deux opérations de la même tâches sont identiques, le problème reste difficile si toutes les durées opératoires des tâches sont unitaires et les time-lags quelconques. En présence simultanée des time-lags min et max, le problème revient au cas de Coupled-task, introduit par Shapiro [142], et largement étudié dans la littérature. Orman et Potts [120] ont présenté une étude approfondie pour le critère du makespan, la majorité de leurs résultats sont des résultats de complexité. Le seul problème resté ouvert pour quelques années est celui où les durées opératoires des premières opération et les durées opératoires des second opérations sont respectivement égales et les time-lags exacts identiques. Ce problème a été résolu polynomialement plus tard par Ahr et al. [4] en utilisant le plus court chemin dans un graphe représentatif des différents profils de la machine lors de l'ordonnancement final.

Les problèmes de machines parallèles avec time-lags sont essentiellement étudiés dans l'environnement informatique, où les time-lags min correspondent aux délais de communication entre les processeurs. Un état de l'art sur les problèmes avec délais de communication est présenté par Chrétienne et Picouleau [45]. Munier et al. [115] ont considéré le cas de time-lags min entre les tâches et comme objectif le critère du flot total, ils se sont focalisés sur les performances théoriques des algorithmes de liste. Boudhar et al. [**?**] ont traité le cas de machines parallèles avec préemption des tâches, le time-lag min représente ici le temps de transport d'une tâche interrompue sur une machine vers une autre machine pour continuer son traitement. Ils ont montré que la minimisation du Cmax est NP-difficile et présentent des heuristiques gloutonnes pour la résolution.

Pour les ateliers de type flowshop, jobshop et openshop, les time-lags sont définis entre les opérations de la même tâche, c'est-à-dire, entre la fin d'une opération et le début de l'opération suivante (*stop-start time-lag*). On trouve dans la littérature d'autres définitions de time-lags comme, de début à début (*start-start time-lag*) ou de fin à fin (*stop-stop time-lag*). Ces trois modèles sont équivalent à condition que les durées opératoires des tâches soient fixes et connues à l'avance. Notre contribution à l'étude des problèmes d'ordonnancement avec time-lags concerne essentiellement les ateliers flowshop. Nous différons la présentation des résultats de la littérature sur ce type d'ateliers, ainsi que nos contributions au chapitre 3.

Pour les ateliers de type openshop, les résultats de la littérature concernent essentiellement le problème à deux machines et time-lags min avec le makespan comme critère à optimiser. La majorité sont des résultats de complexité. Rayward-Smith et Rebaine [132] ont montré la NP-complétude du cas où les time-lags sont constants. Ce problème reste NP-difficile même si les durées opératoires de toutes les tâches sont unitaires. Rebaine et Strusevich [134] ont étudié le cas particulier où les time-lags min ne dépendent pas des tâches, ils ont proposé des heuristiques avec garantie de performances, Yu [156] a considéré dans sa thèse d'autres cas particuliers.

La littérature sur le jobshop en présence de time-lag est très réduite, Dell'Amico [51] a considéré le cas de deux machines et time-lags min avec minimisation du makespan. Il a montré que ce

problème est équivalent au cas de flowshop général à deux machines et time-lags. Or, comme le problème de flowshop est déjà NP-difficile (voir chapitre 3) alors le problème de jobshop est aussi NP-Difficile. Deppner [53] a proposé des approches de résolution basées sur des heuristiques de construction, Caumond et al. [32] ont présenté un algorithme mémétique pour la minimisation du makespan, De Bontridder [17] considère la minimisation de la somme pondérée des retards avec des time-lags min et des dates de disponibilité et un graphe de précédences entre les tâches, et a proposé une méthode de recherche tabou pour la minimisation du makespan.

2.4 Détérioration de tâches

L'ordonnancement avec la prise en compte de la détérioration des tâches couvre une partie des applications industrielles. La détérioration d'une tâche exprime ici le fait que la durée opératoire d'une tâche est une fonction croissante dépendante de la date de début d'exécution de la tâche. Ainsi la durée opératoire d'une tâche j s'écrit sous forme $p_j = F_j(t_j)$ où F_j est une fonction croissante et t_j est l'instant de début d'exécution de j. Par exemple, en sidérurgie il faut exécuter une série d'opérations sur la matière en fusion, celle-ci se refroidit avec le temps, et la durée d'exécution des opérations s'allonge avec ce refroidissement [28]. Un autre exemple, concerne l'entretien préventif des machines, où la durée de l'intervention dépend du dernier entretien effectuée. On trouve aussi d'autres applications dans les services, par exemple le temps ou l'effort nécessaire pour éteindre un incendie de feux est fortement liés à l'instant de début de l'intervention. D'autres applications sont cités par Kunnathur et Gupta [96], Mosheiov [109]. Plusieurs fonctions de détérioration sont considérées dans la littérature, on trouve les cas les plus fréquents, comme les fonctions linéaires de type $F_j(t_j) = a_j + b_j \times t_j$ (a_j est la durée intrinsèque et b_j est le taux de détérioration) ou linéaires par morceaux de style $F_j(t_j) = a_j$ si $t_j \leq A_j$ et $F_j(t_j) = a_j + b_j \times t_j$ si $t_j > A_j$ (A_j date critique à partir de laquelle la tâche commence à se détériorer), la fonction F est dite uniformement lineaire si $A_j = A, \forall j$. Les fonctions plus compliquées comme les fonctions quadratiques, exponentielles, ou simplement non linéaires, sont rarement considérées dans la littérature . Gawiejnowicz [66], Aidaee et Wormer [6], Cheng et al. [38] ont successivement présenté des *surveys* sur les problèmes d'ordonnancement avec détérioration de tâches.

La majorité des résultats de la littérature concerne les problèmes à une machine. Gupta et Gupta [75], Browne et Yechiali [22], Gawiejnowicz et Pankowska [67], Mosheiov [108], [109], [110] ont étudié le modèle à une machine et une fonction de détérioration linéaire $p_j = a_j + b_j \times t_j$ pour différents critères. Par exemple, Gupta et Gupta [75] ont traité le cas uniformément linéaire, ils ont montré que la minimisation de tout critère régulier est polynomial, la séquence optimale est donnée par l'ordre décroissant des a_j/b_j. Mosheiov [109] a étudié le cas linéaire où les durées opératoires intrinsèques des tâches sont nulles et les dates de lancement critiques des tâches sont toutes égales. Il a considéré que la machine est disponible à un instant strictement positif, il a montré que la durée totale de l'ordonnancement est indépendante de la séquence des tâches. Kubiak et Van De Velde [95] ont étudié le même problème que Mosheiov [109], ils ont montré que le problème du makespan devient NP-difficile au sens fort si les durées intrinsèques des tâches sont strictement positives, et de manière générale, ils ont montré que dans le cas de fonctions de détérioration générales (il existe au moins une fonction de détérioration qui n'est pas linéaire) la minimisation de n'importe quel critère régulier est NP-difficile au sens ordinaire. D'autres résultats sont publiés dans la littérature sur une machine, exemple Cheng et Ding [37] ont traité

des dates d'arrivée, Bachman et Janiak [9] étudient le critère du plus grand retard algébrique.

Le modèle de machines parallèles est très peu considéré dans la littérature, on trouve ce modèle avec des fonctions de détérioration linéaire $p_j = a_j \times t_j$. Mosheiov [111] et Chen[34], [35] ont respectivement montré que le problème du makespan et du flot total sont NP-complet en présence de deux machines et des détériorations de type $p_j = a_j \times t_j$ pour tout j.

Pour les ateliers de production de type flowshop, jobshop et openshop, on distingue dans la littérature deux modèles. Le premier modèle de détérioration est basé sur les dates de début des tâches, c'est à dire, chaque tâche i a une durée opératoire $p_{i,j}$ sur la machine j donnée par $p_{i,j} = a_{i,j} + b_{i,j} \times t_j$. Mosheiov [112] a considéré le modèle de détérioration de type $p_{i,j} = b_{i,j} \times t_j$ pour les ateliers flowshop, jobshop et openshop. Il a montré que la minimisation du makespan pour le flowshop et l'openshop à deux machines est polynomial, et sont NP-difficiles dès que le nombre de machines est strictement supérieur à deux. Il a montré aussi que le jobshop est NP-difficile même pour deux machines. Kononov et Gawiejnowicz [94] ont montré que le problème de flowshop à deux machines devient NP-difficile si la détérioration des tâches est de type $p_{i,j} = a_{i,j} + b_{i,j} \times t_j$. Le deuxième modèle de détérioration des tâches est introduit par Sriskandarajah et Wagneur [146] et Wagneur et Sriskandarajah [151], [152] où la durée opératoire d'une tâche sur une machine est une fonction croissante du temps écoulé entre sa date de fin sur la première machine et sa date de début sur la machine encours (les durées opératoires des tâches sur la première machine sont constantes, c'est-à-dire, pas de détérioration de la première opération). Sriskandarajah et Wagneur [146] ont montré que la minimisation du makespan dans un flowshop à deux machines et une détérioration linéaire des tâches est NP-difficile au sens fort. Une autre difficulté de ce problème est liée au placement des tâches, c'est-à-dire étant donnée une séquence de tâches qui minimise un critère régulier, trouver le meilleur placement des tâches n'est pas évident, ce problème est dit *Problème Restreint*, connu aussi par *Timing Problem* dans la littérature, [145]. Le problème de placement de tâches est rarement posé dans le cas classique de l'ordonnancement, car dès que la séquence optimale est connue, le placement au plus tôt des tâches donne la meilleure valeur de la fonction objectif. Wagneur et Sriskandarajah [151] ont utilisé un système dynamique pour trouver les dates de lancement des tâches sur la première machine, Finke et Jiang [56] ont proposé une méthode plus simple pour la résolution du problème du *Timing* pour les critères Cmax et Lmax, ceci pour toute fonction continue croissante de détérioration de tâches. Ils ont utilisé le modèle inverse du flowshop (c'est-à-dire en inversant l'ordre des machines). Dans [58], nous avons proposé un autre algorithme pour le problème de *Timing* dont l'objectif est de minimiser le flot total, cet algorithme sera détaillé dans le chapitre 4. A notre connaissance ce deuxième modèle de détérioration des tâches n'est pas considéré pour d'autres types d'ateliers.

2.5 Conclusion

Dans ce chapitre, nous avons passé en revue les différentes contraintes prises en compte dans les problèmes d'ordonnancement que nous avons étudiés. Nous avons présenté l'intérêt de ces contraintes ainsi que différentes applications industrielles. Nous avons exposé les plus importants travaux publiés dans la littérature, toutefois sans être exhaustif. Nous avons volontairement omis de présenter la revue de littérature sur les problèmes de flowshop en présence de différentes contraintes, cette partie sera exposée en détail dans le chapitre 3 avec nos contributions.

3

Complexité des problèmes

3.1 Introduction

En complément de la revue de la littérature présentée dans le chapitre précédent, nous détaillons dans ce chapitre nos travaux de recherche sur le modèle de flowshop en présence de contraintes de groupement des tâches et de contraintes d'écarts temporels. En particulier, nous abordons ici la complexité des problèmes de flowhop intégrants ces contraintes. Nous commençons cette section par un rappel de la théorie de la complexité, puis nous traitons, dans les sections 3 et 4 les contraintes considérées.

3.2 Théorie de la complexité

Dans cette section, nous présentons les concepts fondamentaux pour appréhender la complexité des problèmes combinatoires. Cette présentation englobe quelques notions de base sur la théorie de complexité et ne prétend pas couvrir cette théorie en elle même. L'ouvrage de Garey et Johnson [65] propose une étude complète.

La théorie de la complexité classe les problèmes combinatoires en plusieurs catégories et donne une orientation sur le traitement de ces problèmes. Parmi les classes de problèmes combinatoires, on distingue les problèmes de décision et les problèmes d'optimisation. Un problème de décision consiste à répondre à une question par un '*OUI*' ou par un '*NON*', alors qu'un problème d'optimisation exige une réponse à une question de style *'trouver un objet tel que ...'*, la réponse ici fournit l'objet recherché. Chaque problème d'optimisation peut être décrit sous forme d'un problème de décision, qui n'est pas nécessairement plus facile que le problème d'optimisation. Ainsi, il est possible d'analyser la difficulté d'un problème d'optimisation en considérant sa version de décision. La théorie de la complexité cherche à connaître la difficulté d'une réponse algorithmique à un problème. Autrement dit, peut-il y avoir un algorithme efficace pour répondre à la question de décidabilité (problème de décision) ou de calculabilité (problème d'optimisation). L'efficacité d'un algorithme est mesuré par son temps d'exécution, c'est-à-dire, pour toute instance des données en entrée de l'algorithme, combien d'opérations élémentaires nécessaires pour atteindre le résultat, cette mesure est aussi dite *complexité algorithmique*. Quand ce nombre d'opérations est borné par un polynôme, dépendant de la taille des entrées, alors l'algorithme est dit *polynomial*,

par contre, si ce nombre d'opérations est borné inférieurement par une forme exponentielle de la taille des données, alors l'algorithme est dit *exponentiel*. Donner une réponse algorithmique à un problème de décision ou d'optimisation nécessite, en premier lieu, de connaître la classe de complexité auquelle appartient ce problème. Dans cette section on ne parlera que de trois classes de complexité, à savoir, la classe P, la classe NP et la classe $NP - Complet$.

Définition 1 *La classe P contient tous les problèmes de décision solubles en temps polynomial, c'est-à-dire, les problèmes de décision pour lesquels il existe un algorithme polynomial permettant de répondre à la question par un OUI.*

Définition 2 *La classe NP contient tous les problèmes de décision solubles en temps polynomial sur une machine de Turing non déterministe*[1]*, c'est-à-dire, les problèmes de décision pour lesquels il existe un algorithme tel que si on lui fournit une solution dont la réponse est 'OUI', il est capable de le vérifier en temps polynomial.*

A partir de ces deux définitions, il est clair que $P \subseteq NP$, mais jusqu'à aujourd'hui, rien ne peut affirmer que $P = NP$ ni infirmer que $P \neq NP$. Dans tout ce travail, on suppose que $P \neq NP$. Avant de définir la classe NP-Complet, il est nécessaire de présenter en premier ce que c'est une transformation polynomial.

Définition 3 *Une transformation d'un problème π_1 en problème π_2 notée $\pi_1 \mapsto \pi_2$, est dite polynomiale, s'il existe un algorithme polynomial qui transforme n'importe quelle instance I_1 du problème π_1 à une instance I_2 du problème π_2, telle que la réponse à l'instance I_1 est OUI si et seulement si la réponse à l'instance I_2 est OUI. Cette transformation est aussi appelée réduction.*

Définition 4 *Un problème π est dit dans la classe $NP - Complet$, si π appartient à la classe NP et tous les problèmes de la classe NP peuvent être transformés en problème π.*

Comme la transformation polynomiale est transitive, il suffit de chercher un seul problème π' de la classe NP qui peut être réduit polynomialement au problème π, afin que ce dernier devienne $NP - Complet$.

Les questions qui se posent à présent sont (i) comment exploiter ces différentes notions pour l'analyse des problèmes combinatoires en général, et les problèmes d'ordonnancement en particulier, (ii) quel est l'intérêt de cette analyse. La réponse à la première question est toute simple, en effet, pour un problème d'optimisation, il suffit d'analyser le problème de décision équivalent. Si ce problème de décision appartient à la classe P, alors le problème d'optimisation est polynomial, si le problème de décision est NP-Complet, alors on dit que le problème d'optimisation est NP-Difficile. En revanche la réponse à la seconde question est moins facile, néanmoins les conséquences pratiques de cette analyse ne souffre d'aucune ambiguïté. En fait, si un problème d'optimisation appartient à la classe P alors il est soluble en temps polynomial, maintenant si on prouve qu'un problème d'optimisation appartient à la classe NP-Difficile, alors il en résulte une explosion combinatoire, de l'ordre de l'exponentielle, pour la recherche de la solution optimale. A partir de là, des méthodes de résolution approchées doivent être utilisées pour les problèmes de grande taille, comme les heuristiques, les algorithmes d'approximation, les metaheuristiques, etc.

[1] Voir dans Garey et Johnson [65] pour la définition d'une machine de Turing non déterministe.

3.3 Groupement de tâches

Cette section traite de la complexité des problèmes d'ordonnancement de type flowshop en présence de machines classiques ou de machines à traitement par batches (p-batch machines et s-batch machines). Avant de présenter les travaux de la littérature et nos contributions, nous commençons par exposer les notations utilisées.

On considère un ensemble de N tâches à ordonnancer. Chaque tâche j est décrite par sa durée opératoire sur la machine i par $p_{i,j}$. Pour certains problèmes, cette durée opératoire est donnée par un intervalle de temps $[a_{i,j}, b_{i,j}]$ comme nous le verrons dans le cas de compatibilité entre les tâches. Une machine i peut être (i) une machine classique, capable de traiter une tâche à la fois, (ii) une p-batch machine, capable de traiter plusieurs tâches simultanément avec une capacité finie ou infinie (iii) une s-batch machine, susceptible de traiter plusieurs tâches en séquentiel dans des batches.

La majorité des travaux de la littérature considère des ateliers à deux machines, avec au moins une des machines qui est *une batching machine*. Nous utilisons, ici, la traditionnelle notation utilisée pour les problèmes d'ordonnancement en trois champs, $\alpha|\beta|\gamma$, Graham et al. [72], où la présence des *batching machines* est représentée dans le champs β. Par exemple, si la *ième* machine est une p-batch machine, on ajoute $p-batch(i)$ dans la notation, par ailleurs, si cette machine a une capacité finie, on ajoute $b < n$ où b représente la capacité de la machine. Dans le cas d'une s-batch machine on précise si le setup time est avec anticipation as ou sans anticipation ns. La compatibilité entre les tâches est ajoutée dans la notation selon le type de graphes, par exemple, si la compatibilité est un graphe d'intervalle, on note cela par $G = INT$.

3.3.1 Modèles avec p-batch machines

Ahmadi et al. [2] sont les premiers à étudier le problème de flowshop en présence d'au moins une p-batch machine. Ils ont supposé que toutes les tâches ont la même durée opératoire sur la p-batch machine. Pour le critère du makespan, il ont montré que le problème $F2\,|\,p-batch(1)$, $b < n$, $p_{1,j} = t\,|\,C_{max}$ (donc par symétrie également le problème $F2\,|\,p-batch(2)$, $b < n$, $p_{2,j} = t\,|\,C_{max}$) est polynomial de complexité $O(nlogn)$. La solution est trouvée par la règle *Full batch-LPT*, i.e. la solution est composée de batches saturés sur la première machine, et sur la deuxième machine, les tâches sont ordonnancées suivant l'ordre décroissant de leur durée opératoire, quant au modèle symétrique, il est résolu par la règle *Full batch-SPT*. Ahmadi et al. [2] ont également montré que le problème $F2\,|\,p-batch(1,2)$, $b_1, b_2 < n$, $p_{1,j} = t_1$, $p_{2,j} = t_2\,|\,C_{max}$ est polynomial et ont proposé un algorithme optimal en $O(n^3)$. Ils ont considéré les trois problèmes précédents pour le critère du flot total, et ont proposé deux algorithmes basés sur la programmation dynamique, respectivement pour les problèmes $F2\,|\,p-batch(2)$, $b < n$, $p_{2,j} = t\,|\,\sum C_i$ et $F2\,|\,p-batch(1,2)$, $b_1, b_2 < n$, $p_{1,j} = t_1$, $p_{2,j} = t_2\,|\,\sum C_i$, la complexité de ces algorithmes est $O(n^3)$. Contrairement au critère du C_{max}, la symétrie n'est pas préservée pour le critère du flot total, ainsi Ahmadi et al. [2] ont montré que le problème $F2\,|\,p-batch(1)$, $b < n$, $p_{1,j} = t\,|\,\sum C_i$ est NP-difficile au sens fort en utilisant la réduction du problème *numerical three-dimensional matching* qui est NP-complet au sens fort (Garey et Johnson [65]). Notons que le problème $F2\,|\,p-batch(1)$, $b < n$, $p_{1,j} = t\,|\,\sum C_i$ est aussi étudié par Hoogeveen et Van de Velde [84]. Ils ont proposé une borne inférieure basée sur la relaxation lagrangienne et ont construit avec cette borne une heuristique avec une performance garantie de $\frac{5}{3}$.

Le modèle de flowshop avec durées opératoires quelconques des tâches est considéré par Potts et al. [129]. Ils ont montré que la séquence optimale du problème $F2 \,|\, p-batch(1,2),\ b_1 = b_2 = n \,|\, C_{max}$ (la capacité des deux machines est infinie) est composée d'au plus deux batches, et ont proposé un algorithme polynomial, noté $F2(n,n)$, qui donne la solution optimale en $O(nlogn)$. Quand l'une des deux machines est une machine classique et l'autre est une p-batch machine avec une capacité infinie, la minimisation du makespan reste polynomial, en $O(nlogn)$. Toutefois, lorsque les deux machines sont des p-batch machines et au moins l'une des deux machines a une capacité finie, la minimisation du makespan est NP-difficile au sens ordinaire. Potts et al. [129] ont utilisé la réduction du problème de PARTITION connu pour être NP-difficile au sens ordinaire (Garey et Johnson [65]).

Dans Oulamara et Finke [124], nous avons considéré le modèle de flowshop sans attente avec deux p-batch machines, les capacités des deux machines sont identiques (comme le traitement des batches se fait sans attente, si les machines n'ont pas la même capacité, il suffit de considérer la plus petite des deux capacités). Nous avons montré que le problème $F2|p-batch(1,2), no-wait|Cmax$ (la capacité des deux machines est infinie) la solution optimale contient au plus deux batches et nous avons proposé un algorithme polynomial qui trouve cette solution en $O(nlogn)$. Nous avons étendu ce résultat, dans Oulamara et al. [127], au cas d'un flowshop à trois p-batch machines. Nous avons montré que la solution optimale contient au plus 9 batches, nous reviendrons sur ce résultat dans le chapitre 4. Concernant le problème à deux p-batch machines, rappelons en premier que pour le modèle de flowshop sans attente, la présence de tâches ayant des durées opératoires nulles sur l'une des machines modifie la structure de la solution optimale du problème, selon l'interprétation donnée à cette durée nulle. En effet, Tanaev et al. [148] rapportent qu'une durée opératoire nulle sur une machine, dans un flowshop sans attente, peut être interprétée de deux manières différentes. Dans la première interprétation, la notation $p_{i,j} = 0$ signifie que la tâche j n'est pas exécutée sur la machine i, alors que dans la deuxième interprétation, la tâche j est exécutée sur la machine i, mais cette durée est tellement petite qu'elle devient négligeable. Nous avons montré dans [124] que le problème $F2|p-batch(1,2), b=k, no-wait|Cmax$ est NP-difficile au sens ordinaire. Dans la preuve nous avons utilisé des tâches avec des durées opératoires nulles sur la première machine, et ces tâches ne sont pas exécutées sur la second machine. Si toutes les tâches doivent être exécutées sur les deux machines, et après plusieurs tentatives, nous n'avons pas réussi à déterminer la complexité du problème $F2|p-batch(1,2), b=k, no-wait|Cmax$, ainsi ce problème reste jusqu'à aujourd'hui un problème ouvert.

Dans le cadre d'un atelier de production des pneumatiques, nous avons considéré, Oulamara et al. [125], le problème de flowshop à deux machines. La première est une machine classique et la seconde est une p-batch machine. La première machine représente une machine d'assemblage de tous les éléments rentrants dans la composition d'un pneu, et la deuxième machine est une machine de cuisson de pneus assemblés sur la première machine. Les durées opératoires des tâches sur la deuxième machine sont données par des intervalles de type $[a_j, b_j]$ (i.e. un temps a_j de cuisson avec une tolérance qui peut aller jusqu'à b_j). Un graphe de compatibilité est défini entre les tâches. Deux tâches sont dites compatibles, si l'intersection de leur intervalle (durée opératoire sur la second machine) n'est pas vide, ce qui permet de pouvoir les exécuter sur la seconde machine dans le même batch. Ainsi, les tâches appartenant au même batch doivent être toutes compatibles (le graphe de compatibilité est ici un graphe d'intervalle). La durée opératoire d'un batch est égale à la plus petite valeur de l'intersection des intervalles des tâches appartenant à ce batch ($p_B = \max_{j \in B} a_j$), ce problème est noté $F2|p-batch(2), G_p = INT, b < n|Cmax$. Nous avons montré que ce problème

est NP-difficile au sens ordinaire, par une réduction du problème de Partition. Nous avons développé des approches heuristiques pour la résolution de ce problème. Nous reviendrons sur ces approches dans le chapitre 4.

3.3.2 Modèles avec s-batch machines

Kleinau [93] a considéré le cas d'un flowshop avec deux s-batch machines où le temps de changement (setup) peut être avec ou sans anticipation, l'objectif est la minimisation du makespan, soit $F2 \mid s - batch(1,2),\ as\ ou\ ns \mid C_{max}$. Il a montré que ce problème est NP-difficile au sens fort. Pour ce même modèle Chen et al. [33] ont proposé deux heuristiques avec des performances garanties $\frac{3}{2}$ et $\frac{4}{3}$ respectives et de complexité $O(nlogn)$.

Glass et al. [71] ont étudié le problème de minimisation de la durée totale de l'ordonnancement sur deux s-batch machines lorsque le temps de changement est sans anticipation, $F2 \mid s - batch(1,2),\ ns \mid C_{max}$. Ils ont montré qu'il existe une solution optimale, dont les batches sont consistants (i.e. chaque batch est composé des mêmes tâches sur les deux machines). Malgré cette propriété qui réduit considérablement l'ensemble des séquences réalisables, ils ont montré que la minimisation du Cmax est NP-difficile au sens fort, cela en utilisant la réduction du problème de 3-PARTITION au problème $F2 \mid s - batch(1,2),\ as\, ou\, ns \mid C_{max}$. Par ailleurs, ils ont proposé une heuristique, notée *FlowBatch*, qui fournit une solution du problème avec une performance garantie de $\frac{4}{3}$.

Dans un environnement sans attente, nous avons considéré, dans [121], le cas d'un flowshop à deux s-batch machines, et un temps de setup avec et sans anticipation, l'objectif est de minimiser le makespan, soit $F2 \mid s-batch(1,2),\ ns$ ou $as\ no-wait,\ \mid C_{max}$. Nous avons montré que ce problème est NP-difficile au sens fort, en utilisant une réduction du problème de 3-partition. Nous avons modélisé ce problème sous forme d'un programme linéaire en nombre entiers, puis nous avons utilisé une approche de résolution par la relaxation lagrangienne, le résultat des expériences numériques montre que cette méthode de résolution est efficace. En effet, la distance relative qui séparent la solution fournie par l'heuristique et la borne inférieure du problème se situe entre 7% et 9%, avec des temps de résolution très rapide.

3.3.3 Modèle mixte

Les modèles d'atelier contenant à la fois les deux types de batching machines ne sont pas considérés dans la littérature. Dans Oulamara [122], nous avons traité le modèle mixte de flowshop sans attente à deux machines. La première machine est une p-batch machine avec une capacité b, et la seconde machine est une s-batch machine. L'objectif est de minimiser le makespan, soit $F2|p-batch(1), s-batch(2), b, no-wait, as\, ou\, ns|C_{max}$, qu'on notera par la suite par $P(b)$. Nous présentons ici la complexité de ce problème.

Théorème. 1 *Le problème $P(b)$ est NP-Difficile au sens fort.*

Preuve. Nous montrons que le problème $P(b=2)$ est NP-difficile au sens fort par la réduction du problème NMTS (*Numerical Matching with Target Sums*) qui est NP-difficile au sens fort (Garey et Johnson [65]) au problème $P(b=2)$.

Le problème NMTS est défini comme suit : Etant donnés deux ensembles d'entiers A et B, avec $A = \{a_1, \ldots, a_n\}$ et $B = \{b_1, \ldots, b_n\}$, et un ensemble $E = \langle E_1, \ldots, E_n \rangle$ d'entiers positifs. Est-il

possible de partitionner l'ensemble $A \cup B$ en n sous-ensembles disjoints $N_1, \ldots, N_n$, où chaque N_i $(i = 1, \ldots, n)$ contient exactement un élément de A et un élément de B, tel que $a_{N_i} + b_{N_i} = E_i$?

Soit une instance I' du problème NMTS, en premier nous modifions l'instance I' comme suit :
Soit $w = \max\{E'_1, \ldots, E'_n\}$. Un nouvelle instance I de NMTS est construite à partir de I' tel que $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$ et $E = \langle E_1, \ldots, E_n \rangle$ où, $a_i = a'_i, i = 1, \ldots, n$, $b_i = b'_i + 3w, i = 1, \ldots, n$ et $E_i = E'_i + 3w, i = 1, \ldots, n$. Il est clair que l'instance I admet une solution, si et seulement si, l'instance I' en admet une.

A partir d'une instance modifiée I du problème NMTS, nous construisons une instance du problème d'ordonnancement $P(b = 2)$ avec $2n^2 + 2(n + 1)$ tâches. Les tâches sont partitionnées en trois groupes : (i) A-tâches : notée $A_{i,j}$, $(i, j = 1, \ldots, n)$, (ii) B-tâches : notée $B_{i,j}$, $(i, j = 1, \ldots, n)$, (iii) D-tâches : notée $D_{i,j}$, $(i = 1, \ldots, n + 1,\ j = 1, 2)$. Les durées opératoires des tâches sur les deux machines sont données par la table 3.1, avec un temps de setup nul sur la deuxième machine.

		$p_{1,j}$	$p_{2,j}$
A-tâches	$A_{i,j},\ i, j = 1, \ldots, n$	E_i	a_j
B-tâches	$B_{i,j},\ i, j = 1, \ldots, n$	E_i	b_j
D-tâches	$D_{1,1}$	0	$2w$
	$D_{1,2}$	0	$E_1 - 2w$
	$D_{i,1},\ i = 2, \ldots, n$	E_{i-1}	$2w$
	$D_{i,2},\ i = 2, \ldots, n$	E_{i-1}	$E_i - 2w$
	$D_{n+1,1}$	E_n	0
	$D_{n+1,2}$	E_n	0

TAB. 3.1 – Durées opératoires des tâches

Le problème de NMTS admet une solution, si et seulement si, le problème $P(b = 2)$ admet une séquence S, tel que la durée totale de l'ordonnancement, $C_{max}(S) \leq (n + 1) \sum_{i=1}^{n} E_i$.

En premier, supposons que le problème NMTS admet une solution. Soit $N_1, \ldots, N_n$ les n sous-ensembles disjoints, tel que chaque sous-ensemble N_i contient exactement un élément de A et un élément de B, et $a_{N_i} + b_{N_i} = E_i,\ i = 1, \ldots, n$. Alors un ordonnancement S du problème $P(b = 2)$ existe, où la durée totale de l'ordonnancement $C_{max}(S)$ est égale à $(n + 1) \sum_{i=1}^{n} E_i$. Dans cet ordonnancement les tâches sont groupées en deux classes de batches, et chaque batch contient exactement deux tâches. Les classes sont : (i) la classe $C1$, contient les batches de type $F_i = \{D_{i,1}; D_{i,2}\},\ i = 1, \ldots, n$, avec une durée opératoire E_{i-1} sur la première machine, excepté pour le batch $\{D_{1,1}; D_{1,2}\}$ qui a une durée opératoire nulle sur la première machine (ii) la classe $C2$, contient les batches de la forme $F_{l,i} = \{A_{l,j}; B_{l,k}\}$, $l, i = 1, \ldots, n$, $j \in N_i$ et $k \in N_i$, avec une durée opératoire E_l sur la première machine et $a_j + b_k = E_i$ sur la seconde.

Le premier batch $F_1 = \{D_{1,1}; D_{1,2}\}$ de la classe C_1 est séquencé en première position dans S. Notons r_i la position du batch $F_i = \{D_{i,1}; D_{i,2}\}$ dans S, r_i est donnée par :

$$\begin{cases} r_1 &= 1, \\ r_i &= r_{i-1} + 2(n - i + 2), \quad i = 2, \ldots, n + 1. \end{cases}$$

Soit $r_{l,i}$ la position du batch $F_{l,i} = \{A_{i,j}; B_{i,k}\}$, $l, i = 1, \ldots, n$ de la classe C_2 dans S, $r_{l,i}$ est donné par la formule suivante

$$r_{l,i} = \begin{cases} r_i + 2(l-i) + 1, & \text{if } i \leq l \\ r_l + 2(i-l), & \text{if } i > l \end{cases}$$

Dans la séquence S, les batches sont placés sans attente et il n y a pas de temps d'attente sur aucune des deux machines ainsi,

$$C_{max}(S) = \sum_{i=1}^{n} Q(F_i) + \sum_{l=1}^{n}\sum_{i=1}^{n} Q(F_{l,i}) = \sum_{i=1}^{n} E_i + n\sum_{i=1}^{n} E_i = (n+1)\sum_{i=1}^{n} E_i.$$

Pour illustrer la séquence S, soit l'exemple suivant, où $n = 4$, $A = \{a_1, a_2, a_3, a_4\}$, $B = \{b_1, b_2, b_3, b_4\}$ et $E = \langle E_1, E_2, E_3, E_4\rangle$. Supposons que le NMTS admet la solution suivante : $N_1 = \{a_1, b_2 \ / \ a_1 + b_2 = E_1\}$, $N_2 = \{a_2, b_1 \ / \ a_2 + b_1 = E_2\}$, $N_3 = \{a_3, b_4 \ / \ a_3 + b_4 = E_3\}$, $N_4 = \{a_4, b_3 \ / \ a_4 + b_3 = E_4\}$. Alors la séquence S est composé des classes C_1 et C_2 comme suit :

C_1 :	C_2 :			
$F_1 = \binom{0}{2w+E_1-2w}^{1}$	$F_{1,1} = \binom{E_1}{a_1+b_2}^{2}$	$F_{1,2} = \binom{E_1}{a_2+b_1}^{3}$	$F_{1,3} = \binom{E_1}{a_3+b_4}^{5}$	$F_{1,4} = \binom{E_1}{a_4+b_3}^{7}$
$F_2 = \binom{E_1}{2w+E_2-2w}^{9}$	$F_{2,1} = \binom{E_2}{a_1+b_2}^{4}$	$F_{2,2} = \binom{E_2}{a_2+b_1}^{10}$	$F_{2,3} = \binom{E_2}{a_3+b_4}^{11}$	$F_{2,4} = \binom{E_2}{a_4+b_3}^{13}$
$F_3 = \binom{E_2}{2w+E_3-2w}^{15}$	$F_{3,1} = \binom{E_3}{a_1+b_2}^{6}$	$F_{3,2} = \binom{E_3}{a_2+b_1}^{12}$	$F_{3,3} = \binom{E_3}{a_3+b_4}^{16}$	$F_{3,4} = \binom{E_3}{a_4+b_3}^{17}$
$F_4 = \binom{E_3}{2w+E_4-2w}^{19}$	$F_{4,1} = \binom{E_4}{a_1+b_2}^{8}$	$F_{4,2} = \binom{E_4}{a_2+b_1}^{14}$	$F_{4,3} = \binom{E_4}{a_3+b_4}^{18}$	$F_{4,4} = \binom{E_4}{a_4+b_3}^{20}$
$F_5 = \binom{E_4}{0+0}^{21}$				

L'ordonnancement S est défini par le passage des baches sur les deux machines dans l'ordre indiqué par les numéros en exposant de chaque batch. La durée totale de l'ordonnancement de S est égale à $C_{max}(S) = 5\sum_{i=1}^{4} E_i$.

Maintenant supposons qu'il existe un ordonnancement S^* tel que $C_{max}(S^*) \leq (n+1)\sum_{i=1}^{n} E_i$. Soit v le nombre de batches dans S^*, on a $v \geq n^2 + n + 1$, car dans le meilleur des cas, il y a exactement 2 tâches par batch. Notons par $P(B_i)$ la durée opératoire du batch B_i, $i = 1, \ldots, v$, sur la première machine. Pour chaque batch B_i, soit la quantité $P'(B_i)$ définit comme suit :

$$P'(B_i) = \begin{cases} 0 \quad \text{si } B_i \text{ contient exactement une seule tâche} \\ \min\{p_j \ / \ j \in B_i\} \quad \text{sinon} \end{cases}$$

Comme la capacité de la première machine est égale à 2, on a

$$\sum_{i=1}^{v} P(B_i) + \sum_{i=1}^{v} P'(B_i) = \sum_{i=1}^{2n^2+2(n+1)} p_j = 2(n+1)\sum_{i=1}^{n} E_i$$

et $C_{max}(S^*) \leq (n+1)\sum_{i=1}^{n} E_i$, alors,

$$\sum_{i=1}^{v} P(B_i) = \sum_{i=1}^{v} P'(B_i) = (n+1)\sum_{i=1}^{n} E_i$$

Ainsi,

- Chaque batch B_i, $i = 1, \ldots, v$, contient exactement deux tâches, *i.e.* $v = n^2 + n + 1$.
- Les tâches ayant les mêmes indices sont regroupées dans le même batch, i.e. le batch contenant la tâche $A_{i,j}$ contient aussi la tâche $A_{i,k}$ ou $B_{i,l}$ ou $D_{i+1,t}$.
- La première machine est continuellement occupée.

Sur la deuxième machine, la somme des durées opératoires des tâches est donnée par

$$n\sum_{i=1}^{n} a_i + n\sum_{i=1}^{n} b_i + \sum_{i=1}^{n} E_i = (n+1)\sum_{i=1}^{n} E_i = C_{max}(S),$$

Comme il n'y a pas de temps mort sur la deuxième machine, seulement le batch $\{D_{1,1}; D_{1,2}\}$ peut être placé en première position de l'ordonnancement, avec une durée opératoire nulle sur la première machine et une durée opératoire E_1 sur la seconde machine.

Comme c'est mentionnée ci-dessus, les tâches ayant les mêmes indices sont groupées dans le même batch (i.e. la tâche $A_{i,j}$ de l'ensemble $A_i = \{A_{i,1}, \ldots, A_{i,n}\}$ peut être groupée soit avec la tâche $A_{i,k}$, avec $j \neq k$, de l'ensemble A_i soit avec la tâche $B_{i,k}$ de l'ensemble $B_i = \{B_{i,1}, \ldots, B_{i,n}\}$, soit avec la tâche $D_{i+1,k}$, $k = 1, 2$). Cela implique les cas suivants :

1. Si $D_{i+1,1}$ est groupée avec une tâche $A_{i,j}$ dans le batch F_i, alors la durée opératoire de F_i sur la deuxième machine est $Q(F_i) = a_i + 2w \leq 3w < E_k$, avec $k = 1, \ldots n$. Cela crée un temps mort sur la deuxième machine entre F_i et le batch sequencé après F_i dans la séquence S^*, cela contredit le fait que la solution optimale ne contient pas de temps mort.
2. Si $D_{i+1,1}$ est groupée avec une tâche $B_{i,j}$ dans le batch F_i, alors la durée opératoire de F_i sur la deuxième machine est $Q(F_i) = b_i + 2w > E_k$, avec $k = 1, \ldots n$. Cela crée un temps mort sur la première machine entre F_i et le batch sequencé après F_i dans la séquence S^*, cela contredit le fait que la solution optimale ne contient pas de temps mort.

Ainsi $D_{i+1,1}$ peut être groupée seulement avec la tâche $D_{i+1,2}$. En plus, d'après le choix de l'instance du problème NMTS, les tâches de l'ensemble A_i ne peuvent être groupées qu'avec des tâches de l'ensemble B_i, sinon un temps mort apparaîtra, soit sur la première soit sur la deuxième machine. Ainsi les batches de la séquence S^* sont de type $\{A_{i,j}; B_{i,k}\}$ et $\{D_{i,1}; D_{i,2}\}$.

Sans perte de généralité, on suppose que le batch $B = \{A_{i,j}; B_{i,k}\}$ est séquencé à la position r. Soit B' le batch séquencé à la position $r+1$. La durée opératoire du batch B' sur la première machine est égale à une certaine valeur E_l, avec $E_l \in E$. Or, comme B' est séquencé sans attente et qu'il n y a pas de temps mort sur les deux machines, alors $p_2(B) = p_2(A_{i,j}) + p_2(B_{i,k}) = E_l$, pour chaque $E_l \in E$. Ainsi l'ensemble $N_l = \{a_j, b_k \ / \ a_i + b_j = E_l\}$ contient exactement un élément de A et un élément de B. En répétant cette opération pour toutes les valeurs de l ($l = 1, \ldots, n$), on obtient une solution pour le problème NMTS. □

3.4 Contraintes d'écarts temporels

Cette section examine la complexité des problèmes d'ordonnancement de type flowshop en présence de contraintes d'écarts temporels. En premier lieu, nous rappelons les notations utilisées pour ce type de contraintes. Notons par n le nombre de tâches à réaliser sur un flowshop à deux ou plusieurs machines. Chaque tâche doit passer sur chaque machine, et l'ordre de passage des tâches sur les machines est le même. La tâche j a une durée opératoire $p_{i,j}$ sur la machine i, $i = 1, \ldots, n, j = 1, \ldots, m$. La tâche j ne peut commencer sur la machine $i+1$ avant d'être terminée sur la machine i ($i = 1, \ldots, m-1$, $j = 1, \ldots, n$). De plus, il existe entre tout couple d'opérations successives $(k, k+1)$ de la tâche j, des contraintes temporelles représentées par un time lag min $\underline{\theta}_{j,k}$ et un time lag max $\overline{\theta}_{j,k}$ avec $\underline{\theta}_{j,k} \leq \overline{\theta}_{j,k}$. Autrement dit, la tâche j ne peut commencer sur la machine $k+1$ avant $\underline{\theta}_{j,k}$ unités de temps de sa fin sur la machine k, et doit absolument commencer sur la machine $k+1$ après $\overline{\theta}_{j,k}$ unités de temps de sa fin sur la machine k. Nous avons choisi de définir les time-lags entre la fin d'une opération et le début d'une opération suivante, i.e. *stop-start time-lags*, ce choix n'a aucune influence sur le résultat final, à partir de l'instant où on se place dans le cadre d'ordonnancement déterministe avec tâches non interruptibles.

3.4.1 Time-lags minimaux

Cette section est dédiée aux problèmes de flowshop en présence de time-lags min seulement. En premier, notons que les ordonnancements de permutation ne sont plus dominants même pour les problèmes à deux machines et des durées unitaires des tâches dès qu'au moins un $\underline{\theta}_{j,k}$ est strictement positif (Fondrevelle [60] présente un contre exemple). Aussi la recherche d'un ordonnancement de permutation est plus dictée par des raisons pratiques de simplicité de pilotage ou par des contraintes techniques.

Pour le flowshop à deux machines, le critère du makespan est le plus étudié dans la littérature. Le cas le plus simple se présente quand toutes les tâches ont le même time-lag min $\underline{\theta}$. On peut facilement montrer que ce problème est équivalent au cas sans time-lags en décalant toutes les tâches sur la deuxième machine de $\underline{\theta}$ unités de temps, par conséquent l'algorithme de Johnson [88] donne la solution optimale. Considérons maintenant le cas où tous les time-lags min sont différents. Le problème $F2|\underline{\theta}_j|C_{max}$ est étudié par Mitten [105]. Il a montré que si on se restreint aux ordonnancements de permutation, la solution optimale peut être obtenue par l'algorithme de Johnson [88] moyennant des modifications sur les durées opératoires des tâches : pour chaque tâche, il ajoute la valeur de son time-lag min aux durées opératoires sur les deux machines. Par contre, si on s'intéresse au cas général (pas de restriction aux ordonnancements de permutation), la minimisation du makespan est NP-difficile au sens fort, Dell'Amico [51], Lawler et al . [97]. Yu et al. [157] ont montré que ce problème reste NP-difficile au sens fort même si toutes les durées opératoires des tâches sont unitaires, ou si les durées opératoires ne dépendent que des tâches mais pas des machines, et que les time-lags ne peuvent prendre que deux valeurs. Yu [156] a proposé une condition pour laquelle l'ordonnancement de permutation devient dominant, à savoir, il faut que $\forall i, j\ \underline{\theta}_i \leq \underline{\theta}_j + \max\{p_{1,j}; p_{2,j}\}$. Par ailleurs, quand le flowshop est composé de plus de deux machines et en présence de contraintes de time-lags min, la plupart des problèmes de minimisation de critères réguliers sont NP-difficile au sens fort, même si on se restreint aux ordonnancements de permutation. En fait, déjà dans le cas classique (sans les time-lags) hormis le makespan, les autres critères sont déjà difficiles. Pour cette raison, nous nous sommes focalisés

à l'étude de cas particuliers où des durées opératoires sont égales voir unitaires. Nous avons alors établi la polynomialité de plusieurs problèmes d'ordonnancement pour des critères réguliers (durée totale, retard algébrique, nombre de tâches en retard). Nous reviendrons sur ces problèmes dans le chapitre 4.

3.4.2 Time-lags maximaux

Dans cette section, nous considérons les problèmes de flowshop en présence seulement de time-lags max, i.e. tous les time-lags min des tâches sont nulles. Les travaux de la littérature se focalisent sur le problème flowshop à deux machines et le critère Cmax. Remarquons que, si tous les time-lags max sont infinis on revient au cas classique de deux machines, et l'algorithme de Johnson [88] donne la solution optimale en $O(nlogn)$. De même, si tous les time-lags max sont nuls, on retrouve cette fois-ci le cas sans attente, où l'algorithme de Gilmore-Gomory [70] donne la solution optimale du makespan en $O(nlogn)$. Dans le troisième cas particulier, on considère à la fois des tâches avec des time-lags max infinis et des tâches avec time-lags max nuls, c'est-à-dire, pour toute tâche j , $\overline{\theta}_j \in \{0, +\infty\}$, Finke et al. [55] ont montré que la minimisation du makespan est NP-difficile au sens fort. Le cas général où au moins un time-lag max a une valeur finie et strictement positive est considéré par Yang et Chern [155]. Ils ont montré que pour le problème de flowshop de permutation à deux machines, la minimisation du makespan est NP-difficile au sens ordinaire. Yang et Chern [155] ont utilisé dans leur preuve de complexité des tâches de durées nulles sur la deuxième machine, et ne précisent à aucun moment le sens de ces durées opératoires nulles. A la lecture de leur preuve, nous avons constaté que les tâches de durées nulles ne sont pas exécutées sur la deuxième machine, d'où la non prise en compte de time-lags max pour ces tâches. Nous avons alors reconsidéré la preuve de ce problème. Dans Fondrevelle et al. [62] nous avons traité le cas où toutes les tâches ont le même time-lags max, $\forall j$, $\overline{\theta}_j = \overline{\theta} > 0$. Nous avons montré que la minimisation de la durée totale de l'ordonnancement est NP-difficile au sens fort. Nous avons élaboré des résultats de dominance pour les time-lags quelconques et nous avons proposé une heuristique avec garantie de performance, basée sur l'algorithme de Gilmore-Gomory [70]. Nous reviendrons sur les approches de résolution dans le chapitre 4.

3.4.3 Time-lags minimaux et maximaux

La présence à la fois de time-lags min et max rend les problèmes à deux machines plus difficiles. Ainsi, tous les résultats de la NP-completitude en présence de l'un des deux types de time-lags se transposent aux problèmes impliquant les deux types simultanément. Toutefois, dans le cas de time-lags exacts, c'est-à-dire, pour chaque tâche le time-lag min est égale au time-lag max ($\forall k, j$, $\underline{\theta}_{k,j} = \overline{\theta}_{k,j}$), peut être considéré à part. En fait, ce problème peut être vu comme une généralisation du cas classique sans attente, où le temps d'attente ne se réduit pas à zéro, mais à une constante positive. En premier, lieu on peut constater facilement que les ordonnancement de permutation ne sont pas dominants, puis pour la minimisation de la durée totale de l'ordonnancement on retrouve le même résultat de time-lags min, c'est-à-dire le problème est NP-difficile au sens fort, Yu et al. [157]. Néanmoins si nous considérons que les ordonnancements de permutation, on a montré que la minimisation de la durée totale de l'ordonnancement est polynomiale, la solution peut être obtenue par l'algorithme de Gilmore-Gomory [70].

3.5 Conclusion

Ce chapitre est essentiellement consacré à la complexité des problèmes de type flowshop en présence des contraintes de groupement de tâches et des contraintes d'écarts temporels. Dans un premier temps, nous avons justifié la nécessité de l'étude de la complexité des problèmes, puis dans un second temps, et pour chacune des contraintes supplémentaires prises en compte, nous avons présenté une partie de nos contributions à l'étude de complexité de ces problèmes, ainsi qu'une revue de la littérature spécifique aux problèmes d'ordonnancement de type flowshop. Dans le chapitre suivant, nous exposerons quelques problèmes polynomiaux ainsi des méthodes de résolution approchées avec garantie de performance pour les problèmes NP-difficiles.

4

Problèmes polynomiaux et heuristiques avec garantie de performances

4.1 Introduction

Bien que la majorité des problèmes d'ordonnancement que nous avons présenté dans le chapitre 3 sont NP-difficiles, néanmoins, il peuvent contenir des sous-problèmes polynomiaux dont la solution peut être utilisée pour améliorer les méthodes de résolution des problèmes originaux. En ce qui concerne les problèmes difficiles, nous avons, en particulier, orienté nos recherches sur la construction d'algorithmes polynomiaux avec garantie de performances, et récemment à la construction de schémas d'approximation polynomiaux (*Polynomial Time Approximation Scheme* - PTAS). Dans ce chapitre, nous passons en revue les différents sous-problèmes polynomiaux traités, et nous présentons brièvement les différentes heuristiques avec garantie de performances, particulièrement, celles construites sur la base de listes. Nous commençons cette section par des notions sur les algorithmes et les schémas d'approximation.

4.2 Définitions et notations

Définition 5 *(algorithme d'approximation) Notons par π un problème de minimisation. Soit $\epsilon > 0$, et $\delta = 1+\epsilon$. Un algorithme A est appelé un $\delta-$approximation algorithme pour le problème π, si pour toute instance I de π, l'algorithme A donne une solution réalisable de I avec une valeur de la fonction objectif égale à $A(I)$ tel que*

$$A(I) \leq \delta \times Opt(I)$$

où $Opt(I)$ est la solution optimale de l'instance I.

La valeur de $\delta = 1 + \epsilon$ est dite ratio *de pire cas*, elle peut être interprétée comme une mesure de qualité de l'algorithme d'approximation A ou comme une garantie de la performance de l'algorithme A, c'est-à-dire, plus δ est proche de 1 meilleur est l'algorithme A. La classe de

complexité APX contient les problèmes d'optimisation ayant un algorithme d'approximation polynomial avec un ratio de garantie de performance finie.

Définition 6 *(schéma d'approximation) Soit π un problème de minimisation :*
- *Un schéma d'approximation pour le problème π est une famille de $(1+\epsilon)$-approximation algorithmes A_ϵ pour le problème π pour tout $0 < \epsilon < 1$.*
- *Un schéma d'approximation polynomial (Polynomial Time Approximation Scheme - PTAS) pour le problème π est un schéma d'approximation avec une complexité polynomiale dépendante de la taille des entrées.*
- *Un schéma d'approximation entièrement polynomial (Fully Polynomial Time Approximation Scheme - FPTAS) pour le problème π est un schéma d'approximation avec une complexité polynomiale dépendante à la fois de la taille des entrées et de $1/\epsilon$.*

Pour un schéma d'approximation polynomial PTAS, il est toute à fait naturel que sa complexité soit proportionnelle à $|I|^{1/\epsilon}$, où $|I|$ est la taille de l'instance I de π. Bien que cette complexité est exponentielle en $1/\epsilon$ elle reste polynomial en taille des entrées. En revanche, un schéma d'approximation entièrement polynomial FPTAS ne peut pas avoir une complexité qui croit exponentiellement en $1/\epsilon$, par exemple une complexité de $|I|^5/\epsilon^4$ est acceptable pour un FPTAS.

Dans les sections qui suivent, nous développons les problèmes d'ordonnancement en présence respectivement de contraintes de groupement de tâches, de time-lags et de détérioration de tâches.

4.3 Groupement de tâches

Dans cette section, nous présentons les problèmes d'ordonnancement que nous avons étudié en présence de contraintes de groupement de tâches.

4.3.1 Modèle avec p-batch machines

Dans [123] et [127], nous avons considéré la minimisation du makespan dans un flowshop sans attente avec des p-batch machines, la capacité des machines est infinie. Le traitement sans attente oblige un batch à passer immédiatement sur une machine dès qu'il termine sur la précédente machine. Cette caractéristique, impose la consistance des batches, c'est-à-dire, que le contenu de chaque batch est identique sur toutes machines. Dans la suite de cette section, nous présentons le problème avec deux p-batch machines, puis le cas de trois p-batch machines, et nous concluons sur le cas de plusieurs machines.

4.3.1.1 Problème à deux p-batch machines

Notons par j_R et j_L deux tâches tel que : $p_{1,j_R} = \max\{p_{1,j} | 1 \leq j \leq n\}$ et $p_{1,j_L} = \max\{p_{2,j} | 1 \leq j \leq n\}$, c'est-à-dire, les tâches j_R et j_L ont les plus grandes durées opératoires respectivement sur la première et la deuxième machine. On note par S^* la solution optimale contenant un nombre minimum de batches. On note par R et L les batches de S^* contenants respectivement les tâches j_R et j_L. Alors les durées opératoires des batches R et L sont $P_1(R) = p_{1,j_R}$ et $P_2(L) = p_{2,j_L}$, respectivement.

Théorème. 2 *L'ordonnancement S^* contient au plus deux batches.*

Preuve. Soit S* la solution optimale contenant le minimum de batches, il est clair qu'aucun batch ne peut être séquencé avant le batch L, dans le cas contraire, le regroupement de ces batches avec le batch L diminue le nombre de batches et n'augmente pas le makespan, ce qui contredit la minimimalité du nombre de batches dans S^* (voir la figure 4.1). De façon symétrique, et pour la même raison aucun batch ne peut être séquencé après le batch R. Aussi, si S^* contient au moins deux batches, il est clair que le batch L précède le batch R. De plus il n'existe pas d'autre batches qui peuvent être séquencés entre L et R, sinon la séquence qui regroupe toutes les tâches dans un seul batch a une durée totale d'exécution meilleure que celle de S^*. Ainsi la solution optimale est soit composée d'une seul batch (toutes les tâches sont regroupées dans le même batch) soit composée des deux batches L et R. □

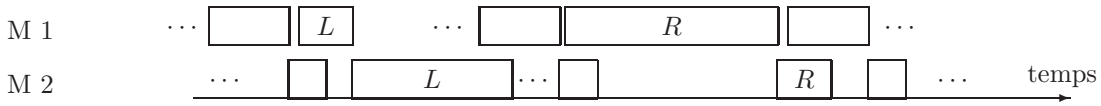

FIG. 4.1 – Un ordonnancement avec plus de deux batches

A partir de ce théorème, nous avons développé un algorithme polynomial qui trouve la solution optimale en $O(nlogn)$. L'algorithme construit deux séquences et compare leur makespan. La première séquence contient un seul batch (toutes les tâches sont incluses dans ce batch) et la deuxième séquence contient deux batches L et R et toutes les autres tâches sont réparties en $O(n)$ sur ces deux batches de façon optimale.

4.3.1.2 Problème à trois p-batch machines

Considérons maintenant le problème de flowshop sans attente avec trois p-batch machines. Soit S^* une séquence optimale avec un nombre minimal de batches. Notons par R, M et L les batches ayant les plus grandes durées opératoires respectivement sur la première, deuxième et troisième machine. Pour tout batch B de S^* on note par a_B, b_B et c_B les durées opératoires de B sur les trois machines. L'objectif ici est de borner le nombre total de batches dans la séquence S^*. Nous commençons en premier par borner le nombre de batches séquencés après le batch R.

Théorème. 3 *Au plus deux batches sont séquencés après R dans la séquence optimale S^*.*

La preuve de ce théorème est basée sur une succession de plusieurs preuves et de résultats intermédiaires. Le lecteur peut trouver le détail de cette preuve dans [127].

Par ailleurs, comme le problème de flowshop sans attente est symétrique, on trouve le même résultat que théorème 3 pour le batch L.

Théorème. 4 *Au plus deux batches sont séquencés avant L dans la séquence optimale S^*.*

D'un autre côté, nous avons montré que, si le batch M n'est pas inclus dans l'un des batches L ou R, alors il est obligatoirement séquencé entre les batches L et R, puis pour la séquence optimale, nous avons montré le résultat suivant,

Théorème. 5 *Au plus trois batches sont séquencés entre le batch L et le batch R.*

En comptabilisant les bornes sur le nombre de batches dans la séquence S^*, on obtient le résultat suivant,

Théorème. 6 *Pour le problème de minimisation du makespan dans un flowshop sans anttente à trois p-batch machines, il existe une solution optimale contenant au plus* 9 *batches.*

Il existe deux structures de la séquence S^* avec au plus 9 batches. Dans la première structure, le batch M est séquencé juste après le batch L suivi de deux batches intermédiaires puis du batch R (i.e. elle a la structure $S^* = (B_1, B_2, L, M, B_3, B_4, R, B_5, B_6)$ voir figure 4.2, alors que dans la deuxième structure, le batch L est suivi de deux batches intermédiaires puis du batch M puis du batch R, une structure de type $S^* = (B_1, B_2, L, B_3, B_4, M, R, B_5, B_6)$.

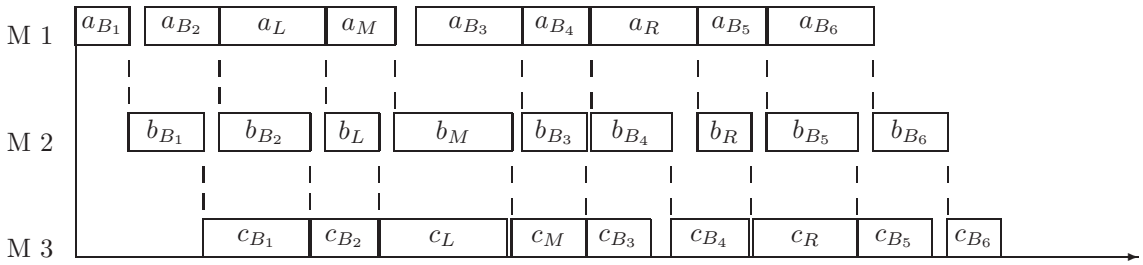

FIG. 4.2 – Un ordonnancement avec au plus 9 batches

Nous avons construit une instance où la séquence optimale contient cinq batches de style $S = (L, M, B_1, B_2, R)$, c'est à dire, exactement trois batches entre L et R, et une autre instance dont la séquence optimale contient sept batches de la forme suivante : $S = (B_1, B_2, L, M, R, B_3, B_4)$, i.e. deux batches avant L et deux autres batches après R. Par ailleurs, nous n'avons pas trouvé d'instances où la solution contient exactement 9 batches, une séquence de style $S^* = (B_1, B_2, L, M, B_3, B_4, R, B_5, B_6$ ou $S^* = (B_1, B_2, L, B_3, B_4, M, R, B_5, B_6)$.

Pour le problème de flowshop avec m p-batch machines, si on fixe le nombre de batch à r, nous avons proposé une méthode d'affectation des tâches aux batches de complexité $O(n^{m(r-2)+1+\lfloor m/r \rfloor})$. Par exemple, cette méthode d'affectation trouve la solution pour le problème à deux machines pour $r = 9$ en $O(n^{22})$. Bien que la complexité de cette méthode d'affectation reste polynomiale, elle est de rang très élevé. Nous avons alors cherché à comparer la meilleure solution respectivement à un, deux et trois batches par rapport à la solution optimale. Nous avons montré qu'une telle solution à, un, deux et trois batches donne un ratio de performance respectivement égale à 3, 2 et $3/2$. Puis, pour le cas général avec m p-batch machines, nous proposons la conjecture suivante,

Conjecture. 1 *Il existe une séquence optimale* S *pour le problème de minimisation du makespan dans un flowshop sans attente avec* m *p-batch machines, où le nombre de batches dans la séquence* S *est indépendant du nombre de tâches et il est une fonction de* m*.*

4.3.2 Modèle p-batch et s-batch machines

4.3.2.1 Modèle classique

Dans [124], nous avons considéré le cas de la minimisation du makespan dans un flowshop à deux machines. La première est une p-batch machine et la seconde est une s-batch machine. Le temps de

changement sur la s-batch machine peut être sans ou avec anticipation ('*as*' ou '*ns*'). Nous avons montré dans le chapitre 3 que, lorsque la capacité de la p-batch machine est finie, la minimisation du makespan est NP-difficile au sens fort. Nous présentons ici le cas où la capacité est infinie. Nous avons montré que la séquence optimale peut être caractérisée par un ensemble de propriétés.

Propriété. 1 *Il existe une séquence optimale où les tâches sont affectées aux batches de la première machine suivant l'ordre croissant de leur durée opératoire sur la première machine.*

La deuxième propriété concerne le nombre de batches sur les deux machines, cette propriété est obtenue à partir de plusieurs résultats intermédiaires, voir [124].

Propriété. 2 *Il existe une séquence optimale telle que le nombre de batches sur les deux machines est le même.*

La propriété 2 nous permet de montrer le résultat principal suivant,

Propriété. 3 *Il existe une séquence optimale dont tous les batches sont consistants.*

A partir des propriétés 1 et 3, nous avons développé un algorithme nommé *Batch-Flow*, basé sur la programmation dynamique, où en utilisant le schéma arrière, les tâches sont ajoutées au début de la séquence partielle construite à chaque étape de l'algorithme. L'algorithme trouve la séquence optimale en $O(n^3)$.

Nous avons adapté le programme dynamique *Batch-Flow* pour le cas de capacité finie de la p-batch machine. Rappelons que la minimisation du makespan dans ce cas est NP-difficile au sens fort. Nous avons montré que l'heuristique obtenu par cette adaptation est un algorithme 2-approximation, et le ratio de performance $\delta = 2$ est atteint. Par ailleurs, nous avons montré que lorsque les durées opératoires des tâches sont constantes sur la première machine, la séquence optimale qui minimise le makespan admet les propriétés suivantes : (i) les batches sur la première machine sont saturés, (ii) les tâches sont affectées aux batches de la deuxième machine dans l'ordre décroissant de leur durée opératoire sur cette machine. A partir de ces deux propriétés, nous avons construit un algorithme polynomial qui trouve la séquence optimale en $O(nlogn)$. De même quand les durées opératoires des tâches sont constantes sur la deuxième machine, nous avons montré qu'il existe une séquence optimale telles que les tâches sont affectées aux baches de la première machine dans l'ordre croissant de leur durée opératoire sur la première machine. Ainsi un programme dynamique est développé pour ce cas particulier, et il trouve la solution en $O(n^2)$.

4.3.2.2 Modèle sans attente

Dans un environnement sans attente, nous avons considéré dans [122] le problème de minimisation du makespan dans un flowshop sans attente à deux machines. La première est une p-batch machine avec une capacité finie, et la seconde est une s-batch machine. Nous avons montré (voir chapitre 3) que ce problème est NP-difficile au sens fort. Toutefois, il existe des cas particuliers qui sont polynomiaux. Nous avons considéré le cas où les durées opératoires sont constantes sur la p-batch machine. Deux cas sont à distinguer, le premier concerne le cas où la capacité de la p-batch machine $b = 2$, et le second cas lorsque cette capacité $b \geq 3$. Pour le premier cas, nous avons montré que tous les batches sur la première machine sont saturés, puis nous avons montré que la minimisation du makespan se réduit à la recherche d'un couplage de cardinalité $n/2 - 1$ et de poids maximum, où n est le nombre de tâches et le poids de chaque arête $e_j = (i, j)$ est donné par $d_j = \min\{p - s, p_{2,i} + p_{)2,j\}}$ (p est la

durée des tâches sur la première machine et s est le temps de changement sur le seconde machine). Nous avons alors élaboré un algorithme qui trouve la séquence optimale en $O(n^3)$ et qui utilise l'algorithme d'Edmonds [54]. Pour le second cas ($b \geq 3$), bien que les durées opératoires des tâches soient constantes, le problème du minimisation du makespan reste NP-difficile. En revanche, quand les durées opératoires des tâches sont constantes sur la seconde machine, les tâches sont affectées aux batches de la première machine dans l'ordre croissant de leur durée opératoire sur la première machine, et nous avons montré que la minimisation du makespan est équivalente à la minimisation de la somme des temps libres de la seconde machine. Nous avons alors élaboré un programme dynamique, basé sur la notion de profil d'un ordonnancement [78], et qui trouve la solution optimale en $O(n^3)$. En se basant sur ce programme dynamique, nous avons développé un algorithme 2-approximation pour le cas général, où le ratio $\delta = 2$ est atteint.

4.3.3 Présence de machines classiques

Dans le cadre d'une collaboration avec GoodYear, nous avons considéré le cas d'un atelier de production de pneumatiques. Cette atelier est assimilé à un atelier de type flowshop hybride composé de deux étages. Le premier étage contient plusieurs machines parallèles classiques et le second étage est composé de plusieurs p-batch machines parallèles. Comme l'atelier de fabrication fonctionne en travail posté, l'objectif est alors de minimiser la durée totale de l'ordonnancement. Dans un premier temps, nous avons considéré le cas simple où chaque étage est composé d'une seule machine, Oulamara et al.[126]. La machine du second étage peut traiter les tâches dans le même batch, si elle sont compatibles. La compatibilité entre les tâches est définie par rapport à leurs durées opératoires. La durée opératoire d'une tâche sur la second machine est donnée par un intervalle $[a_j, b_j]$, et deux tâches sont compatibles si l'intersection de leur intervalle de durée opératoire n'est pas vide. La compatibilité entre les tâches est alors représentée par un graphe d'intervalle. Nous avons montré que le problème de minimisation du makespan est NP-difficile au sens ordinaire. Nous avons alors développé trois heuristiques H_1, H_2 et H_3 pour la minimisation du makespan. Les heuristiques H_2 et H_3 sont basées sur un ordre initial des tâches (respectivement dans l'ordre croissant des $p_{1,j} + a_j$ et des a_j) puis nous avons appliqué le même principe d'insertion que l'algorithme de NEH, [118]. Quand à l'heuristique H_1, elle est basée sur la relaxation du problème de flowshop, où nous ne considérons que la deuxième machine (i.e. le problème $B1|G = INT, b|C_{max}$). Nous avons montré que le problème $B1|G = INT, b|C_{max}$ est polynomial, et nous avons développé un algorithme, noté FCBLPT, correspondant aux étapes suivantes, (i) construire la liste L en rangeant les tâches dans l'ordre croissant de leur point initial de l'intervalle de durée opératoire (i.e. dans l'ordre croissant des a_j, $j = 1, \ldots, n$) (ii) à l'itération i, on construit le batch B_i et on ajoute dans B_i la première tâche j de la liste L, puis on ajoute au batch B_i les $b-1$ tâches compatibles avec j, puis on enlève toutes les tâches ajoutées à B_i de la liste L. On répète l'étape (ii) jusqu'à ce que la liste L devienne vide. A la fin on obtient une liste de batches dont le séquencement est arbitraire, et la valeur de l'objectif est égale à la somme des durées opératoires des batches construits. Puis à partir de la liste des batches obtenue par l'algorithme FCBLPT, nous avons appliqué l'algorithme de Johnson [88]. Nous avons montré que cette heuristique est un 2-approximation algorithme pour le problème de la durée totale, et le ratio $\delta = 2$ est atteint.

Dans un second temps, dans Bellanger et Oulamara [16], nous avons étudié le problème général du flowshop hybride. La minimisation du makespan est NP-difficile au sens ordinaire. Nous avons orienté notre recherche vers la construction d'heuristiques avec garantie de performances. Nous avons, en premier, déterminé la borne inférieure qui nous est nécessaire pour le calcul de la performance des

heuristiques. Cette borne est déterminée par le principe du plus long chemin d'exécution des tâches. Puis, nous avons développé plusieurs heuristiques, à la fois pour le cas général et pour des cas particuliers, où il n'y a qu'une seule machine respectivement au premier et au second étage. Les trois heuristiques que nous avons développé pour le cas général, sont basées sur des algorithmes de liste. La première heuristique H_{LPT} ordonnance chaque étage séparément. Sur le première étage, les tâches sont placées suivant la règle LPT, puis sur le second étage, la construction des batches se fait par l'algorithme FCBLPT, et leur ordonnancement se fait par la règle LPT. Nous avons montré que cette heuristique donne une solution en $O(nlogn)$ avec une performance garantie de $\frac{8}{3} - \frac{2m}{3}$ et cette borne est atteinte. La deuxième heuristique H_{LBPT} commence par construire la liste des batches en utilisant l'algorithme FCBLPT, puis elle prend les batches un à un, les tâches de chaque batch sont ordonnancées sur le premier étage dans l'ordre LPT, puis sur le second étage, le batch en question est placée sur la première machine disponible. Cette heuristique donne une solution du makespan en $O(nlogn)$ avec une performance $\frac{10}{3} - \frac{4m}{3}$. La troisième heuristique H_j, procède exactement comme l'heuristique H_{LBPT}, sauf que la liste des batches est obtenue par l'algorithme FCBPLT puis la liste est rangée suivant la règle de Johnson [88]. Cette fois-ci, la performance de l'heuristique est $4 - \frac{2}{m}$. Bien que théoriquement l'heuristique H_J est la plus mauvaise de trois heuristiques, sur le plan expérimental, les heuristiques H_J et H_{LBPT} donnent de meilleurs résultats que l'heuristique H_{LPT} avec un léger avantage à l'heuristique H_J.

Finalemet, nous avons traité le cas particulier où les tâches ont des durées opératoires constantes sur le premier étage ($\forall j, p_{1,j} = p$). Ce problème revient à considérer le cas de m p-batch machines en parallèle avec des dates d'arrivée généralisées, par exemple, à l'instant $r_1 = p$ il y a m_1 tâches disponibles (ici la date r_1 correspond à l'instant d'achèvement de m_1 tâches sur le première étage. Rappelons que le premier étage est composé de m_1 machines). Les dates d'arrivée sont de la forme $r_i = \alpha p$, avec α un entier tel que $0 < \alpha < \lceil n/m_1 \rceil$. Puisque le problème $PB(m)|r, G = INT, b|C_{max}$ est NP-difficile (le problème sans dates d'arrivée est déjà NP-difficile), nous nous sommes intéressés à la proposition d'un schéma d'approximation polynomial (PTAS). Le principe du schéma est de simplifier le problème afin d'obtenir une solution en temps polynomial avec un facteur $(1 + \epsilon)$ de la solution optimale ($0 < \epsilon < 1$). Pour garantir ce facteur, les pertes engendrées par la simplification doivent être limitées. La figure 4.3 resume les différentes étapes du schéma d'approximation.

L'idée du schéma d'approximation consiste en premier à arrondir les dates d'arrivée à un multiple d'un nombre M où M est une fonction de la borne inférieure de l'objectif. Ce changement garantit un nombre limité de dates d'arrivée différentes. Avant d'énoncer la valeur de M on définit $r_{max} = \max\{r_j | j = 1, \ldots, n\} = \lceil n/m_1 \rceil p$, la plus grande date d'arrivée, $a_{max} = \max\{a_j | j = 1, \ldots, n\}$ le plus grand point initial des intervalles de durées opératoires des tâches et $P(B)$ la somme des durées opératoires des batches obtenus par l'algorithme FCBLPT. Notons par opt la valeur de l'objectif à l'optimum. Les bornes inférieur et supérieur de opt sont données par le résultat suivant.

Lemme. 1 $\max\{r_{max}, a_{max}, P(B)/m\} \leq opt \leq r_{max} + a_{max} + P(B)/m$

Soit $M = \epsilon \times \max\{r_{max}, a_{max}, P(B)/m\}$, avec $1 < \epsilon < 1$. D'après le lemme 1 la valeur du makespan est inférieure à $3M/\epsilon$. La transformation des dates d'arrivée consiste alors à modifier chaque r_i en un multiple de M, c'est à dire $r_i = M\lfloor r_i/M \rfloor$. On a alors le résultat suivant,

Lemme. 2 *Avec une perte de* $1 + \epsilon$ *on suppose qu'il y a au plus* $1/\epsilon + 1$ *dates d'arrivée différentes dans le problème original.*

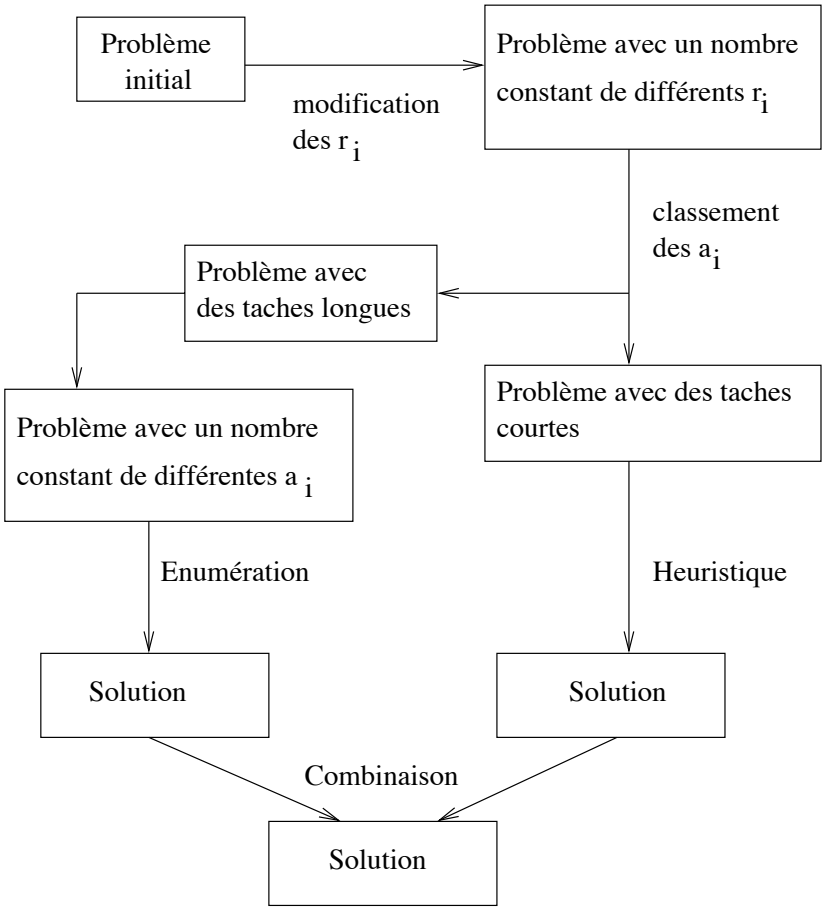

FIG. 4.3 – Schéma général du PTAS

La preuve de ce lemme est donnée en détail dans [16].

La deuxième étape consiste à partager l'axe du temps (l'horizon de l'ordonnancement) en intervalles égaux, sauf pour le dernier, de telle sorte que chaque date d'arrivée modifiée coïncide avec le début d'un intervalle. Ainsi l'horizon du temps $[0, 3M/\epsilon)$ est partitionné en $1/\epsilon + 1$ intervalles disjoints. On note par $\delta_i = [(i-1)M, iM)$ le ième intervalle de temps $i = 1, \ldots, 1/\epsilon$ et $\delta_{1/\epsilon+1} = [M/\epsilon, 3M/\epsilon)$ le dernier intervalle. L'objectif est d'ordonnancer les tâches dans chaque intervalle séparément. Avant de procéder à cet l'ordonnancement, on partitionne d'abord les tâches en deux sous ensembles, le premier sous-ensemble est constitué de tâches courtes et le second sous-ensemble contient les tâches longues. Une tâche est dite *courte*, si $a_j < \epsilon M$, sinon elle est dite *longue*. La suite du schéma d'approximation consiste à séparer l'ordonnancement des tâches courtes de celui des tâches longues. Les tâches longues sont ordonnancées de façon optimale après la modification de leur durée opératoire, et les tâches courtes sont ordonnancées de façon approximatives.

L'ordonnancement des tâches courtes est réalisé par un l'algorithme noté ST. On commence par appliquer l'algorithme FCBLPT pour obtenir une liste de batches. Cette liste est rangée dans l'ordre décroissant des p_{B_i}/k_i, où p_{B_i} est la durée opératoire du batch B_i et k_i est le nombre de tâches contenues dans B_i. A chaque date d'arrivée modifiée (c'est-à-dire à chaque début d'un intervalle de

l'horizon du temps), il y a m_1 tâches possibles à ordonnancer. L'algorithme ST calcule, à chaque début d'un intervalle de l'horizon du temps, le nombre de tâches restantes à ordonnancer. S'il y a suffisamment de place dans l'intervalle, l'algorithme ordonnance le premier batch de la liste, sinon il passe à l'intervalle de temps suivant.

Lemme. 3 *L'algorithme ST est un schéma d'approximation polynomial (PTAS), lorsque toutes les tâches sont courtes, avec une perte d' au plus* $1 + (2 + 1/m)\epsilon^2$ *de l'optimum.*

La preuve de ce lemme est donnée en détail dans [16].

Le traitement des tâches longues nécessite une nouvelle transformation avant leurs ordonnancement. En fait, pour limiter le nombre d'ordonnancements possibles des tâches longues, leurs durées opératoires sont transformées de telle sorte que le nombre de durées opératoires différentes soit indépendant de n. Pour cela, on utilise la technique de transformation d'Afrati [1] qui arrondit les durées opératoires des tâches (seulement le point initial a_j de l'intervalle de durée opératoire) en multipliant a_j de chaque tâche par $(1 + \epsilon)$, puis on fait décroitre cette nouvelle durée opératoire vers la plus proche puissance entière de $(1 + \epsilon)$. Résultat, le nombre de valeurs différentes de points initiaux a_i est bornée.

Lemme. 4 *Le nombre de points initiaux distincts* a_i *des intervalles des durées opératoires des tâches est borné par* $\lfloor log_{1+\epsilon}(1/\epsilon^2) + 1 \rfloor$.

Soit $\overline{a}_1, \ldots, \overline{a}_v$ les v points initiaux distincts des intervalles des durées opératoires des tâches modifiées. Pour l'ordonnancement des tâches longues, on utilise les concepts de *configuration machine* et du *profil d'exécution*, introduits par Hall et Shmoys [77]. Un profil regroupe une configuration de chaque machine, c'est à dire un nombre de batches longs (un batch est dit long s'il contient au moins une tâche longue) possibles à ordonnancer dans chaque intervalle de l'horizon de temps. Comme la durée opératoire d'un batch long est choisie dans la liste $\overline{a}_1, \ldots, \overline{a}_v$, alors le nombre de batches à placer dans chaque intervalle est borné. En conséquence le nombre total de configurations machines Δ est aussi borné. Comme ce nombre est indépendant du nombre de tâches n, alors une énumération de ces ordonnancements est possible.

Enfin nous avons combiné les deux approches pour l'ordonnancement des tâches courtes et des tâches longues dans le même algorithme noté LT-ST. La première étape de l'algorithme consiste à générer tous les profils d'exécution possibles, puis pour chaque profil, l'algorithme ordonnance en premier les tâches longues, puis les tâches courtes dans les espaces restants. Si l'un des deux ordonnancements (tâches courtes ou tâches longues) n'est pas possible ce profil est éliminé et l'algorithme passe au profil suivant, ainsi de suite jusqu'à ce que la liste des profils devienne vide. A la fin, l'algorithme choisit l'ordonnancement ayant la plus petite valeur du makespan.

Théorème. 7 *L'algorithme LT-ST est un schéma d'approximation (PTAS) pour le problème* $BP(m)|r, G = INT, b|C_{max}$.

L'algorithme LT-ST génère une solution du problème $BP|r, G = INT, b|C_{max}$ en $O(nlogn+n\times(m+1)^{\Delta})$ avec $\Delta \leq 2^{1/\epsilon+1} \times v^{3/\epsilon^2}$ (nombre total de profils), avec une perte au plus $1+4\epsilon+(3+1/m_2)\epsilon^2$ par rapport à la solution optimale.

4.4 Ecarts temporels

Dans cette section, nous exhibons les résultats des problèmes d'ordonnancement en présence de time-lags. Comme nous l'avons déjà mentionné dans le chapitre précédent, la plupart des problèmes d'ordonnancement de flowshop sont NP-difficiles. Par exemple, en présence seulement de contraintes de time-lags min, à moins de se restreindre à des cas particuliers comme des durées opératoires égales, voir unitaires, il n'y a que la minimisation du makespan, dans le cas de deux machines, qui est polynomial. Ainsi, cerner la frontière des problèmes difficiles en présence de contraintes d'écarts temporels est un défi intéressant à relever. Nous avons concentré nos efforts à l'étude de certains cas particuliers.

En présence de time-lags min, nous avons mentionné dans le chapitre 3 que les ordonnancements de permutation ne sont pas dominants dans le cas du flowshop à deux machines. Dell'Amico [51] s'est penché sur les conditions à satisfaire pour qu'un ordonnancement de permutation soit dominant. Il présente deux cas pour lesquels un ordonnancement de permutation est dominant : (i) si $\max \underline{\theta}_j | 1 \leq j \leq n\} \leq \min\{p_{1,j} + \underline{\theta}_j \ | 1 \leq j \leq n\}$ et (ii) si $\max\{\underline{\theta}_j | 1 \leq j \leq n\} \leq \min\{p_{2,j} + \underline{\theta}_j | 1 \leq j \leq n\}$. Yu [156] a montré une condition plus générale qui assure la dominance des ordonnancements de permutation, à savoir $\forall i, j \ \underline{\theta}_i \leq \underline{\theta}_j + \max\{p_{1,j}, p_{2,j}\}$. Si cette condition est vérifiée, alors l'algorithme de Mitten [105] résout le problème de minimisation du makespan en temps polynomial.

Nous avons considéré le cas de durées opératoires constantes dans un flowshop de permutation à deux et trois machines. Pour le cas de deux machines, la date de fin d'une tâche de la position $[i]$ est calculé facilement par la formule $C_{2,(i)} = \max_{1 \leq j \leq i}\{\underline{\theta}_j\} + (i+1)p$. De même pour trois machines, la date de fin de la tâche de la position $[i]$ est calculée par la formule $C_{3,(i)} = \max_{1 \leq j \leq k \leq n}\{\underline{\theta}_{1,j} + \underline{\theta}_{2,k}\} + (i+2)p$. Cette dernière formule nous permet de montrer que la plus petite valeur du makespan est obtenue en exécutant les tâches dans l'ordre croissant des time-lags entre les deux premières machines, $\underline{\theta}_{1,j}$. Une autre permutation optimale, éventuellement identique à la première, consiste à ordonnancer les tâches dans l'ordre décroissant des time-lags entre les deux dernières machines $\underline{\theta}_{2,j}$. La valeur optimale du makespan est obtenue par la formule $C_{max} = max_{1j \leq n}\{\underline{\theta}_{1,j} + \underline{\theta}_{2,j}\} + (n+2)p$.

En présence de dates d'arrivée, la minimisation du makespan dans un flowshop à deux machines est équivalent au problème du minimisation du retard algébrique. L'optimum pour le makespan est obtenue en exécutant les tâches dans l'ordre croissant des dates de disponibilité, et l'optimum pour le critère du retard algébrique est obtenu en exécutant les tâches dans l'ordre croissant de leur date de fin souhaitée. Enfin pour le critère de nombre de tâches en retard, nous avons modifié l'algorithme de Moore [107] (optimal pour le problème de minimisation du nombre de tâches en retard sur une machine), nous avons adapté cet algorithme au problème de flowshop à deux machines, l'algorithme modifié donne une solution optimale en $O(n^2)$.

En présence de time-lags max, nous avons montré, dans le chapitre 3, que la minimisation du makespan sur deux machines est NP-difficile au sens fort, même si toutes les tâches ont des time-lags max identiques. Par ailleurs, nous avons mentionné que les ordonnancements de permutation en général, ne sont pas dominants, toutefois il existe des cas particuliers où ces ordonnancements de permutation sont dominants, par exemple, nous avons établi le cas où $\max_{1 \leq j \leq n}\{\overline{\theta}_j\} < \min_{1 \leq j \leq n}\{p_{1,j} + p_{2,j}\}$. En effet, pour ce cas particulier, il y a que les ordonnancements de permutation qui respectent la contrainte de time-lags. Pour le cas général (i.e. time-lags max quelconques) nous avons montré que l'algorithme de Gilmore-Gomory [70] est un algorithme 2-approximation pour le problème de

makespan, et le ratio $\delta = 2$ est atteint asymptotiquement. La présence à la fois de time-lags max et min rend les problèmes encore plus difficiles, néanmoins, pour le cas de deux machines et où pour chaque tâche $\underline{\theta}_j = \overline{\theta}_j$, nous avons montré que l'algorithme de Gilmore-Gomory donne une solution optimale pour le critère du makespan.

4.5 Détérioration de tâches

Dans cette section, nous présentons notre contribution au problème d'ordonnancement flowshop en présence de contraintes de détérioration de tâches. Nous avons considéré la contrainte de détérioration, où la durée d'une tâche sur une machine est une fonction croissante du temps écoulé entre sa date de fin sur la première machine et sa date de début sur la deuxième machine. Pour ce modèle, nous avons mentionné dans le chapitre 2 que le problème de placement des tâches pour une séquence donnée qui optimise un critère régulier, est un problème non évident. Pour les critères makespan et retard algébrique, Finke et Jiang [56] ont proposé une méthode simple pour la résolution du problème restreint, ceci pour toutes les fonctions continues non décroissantes de détérioration de tâches. Ils ont utilisé le modèle inverse du flowshop. Autrement dit, les durées opératoires des tâches sur la deuxième machine sont des constantes positives, tandis que leurs durées opératoires sur la première machine sont des fonctions continues non-décroissantes, qui dépendent du temps écoulé entre la fin d'exécution de la tâche sur la première machine et son début d'exécution sur la seconde machine. A partir de ce modèle, Finke et Jiang [56] ont facilement traité le problème de placement des tâches d'une séquence donnée, ils ont montré que pour le critère du makespan, il existe un placement optimal dans lequel il n'y a aucun temps mort suivi par une opération sur la première machine, A partir de ce résultat, ils ont proposé un algorithme glouton, où le placement optimale est obtenu en plaçant chaque opération le plus tôt possible. Finke et Jiang [56] ont aussi proposé un algorithme de placement pour le critère du retard algébrique, basé sur l'étude du modèle inverse.

Nous avons considéré, dans [58], le problème de placement des tâches pour le critère de la somme des dates de fin des tâches (flot total) pour une séquence donnée de tâches. Contrairement aux critère du makespan et du retard algébrique, le modèle inverse pour le critère du flot total est aussi difficile à traiter pour le critère du flot total que le modèle original. Nous avons développé un algorithme de placement des tâches en utilisant le modèle original. A partir d'un placement sans attente des tâches (voir la figure 4.4), nous définissons la notion de bloc, par un ensemble de tâches tel qu'il n y a pas de temps mort entre ces tâches sur la deuxième machine, les seuls temps morts dans un bloc se trouvent sur la première machine. A partir de ce placement sans attente, une diminution de la valeur du flot total peut être obtenue par la réduction des temps morts sur la deuxième machine (réduction de $I1$ et de $I2$ dans la figure 4.4). Or une réduction d'un temps mort sur la deuxième machine nécessite un déplacement du bloc à gauche du placement actuel (par exemple, la réduction de $I1$ nécessite le déplacement à gauche du Bloc 2 à gauche). Cette dernière opération n'est possible que si les tâches situées après le dernier intervalle de temps mort d'un bloc sur la première machine peuvent être aussi déplacées à gauche (dans la figure 4.4, il est possible de déplacer le bloc 2 à gauche s'il est possible de réduire le temps mort $w2$ du Bloc 1). Nous définissons alors une opération de déplacement à gauche dans un bloc, comme l'opération qui déplace à gauche toutes les tâches d'un bloc placées après le dernier intervalle sur la première machine. Un bloc est dit alors améliorable si l'opération de déplacement dans ce bloc réduit la valeur du flot total du placement. A partir de ces définitions, nous avons fait ressortir les conditions d'optimalité d'un placement, et nous avons développé un

algorithme polynomial qui trouve ce placement, cela pour toute fonction de détérioration des tâches.

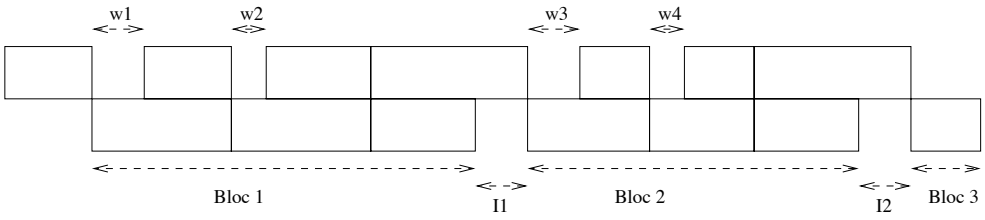

FIG. 4.4 – Un ordonnancement avec plus de deux batches

4.6 Conclusion

Dans ce chapitre, nous avons présenté les problèmes d'ordonnancement en présence de contraintes de groupement de tâches, de time-lags et de détérioration de tâches. Pour chacun des problèmes présenté, nous avons soit mis en évidence des algorithmes polynomiaux pour la résolution, soit pour les problèmes difficiles nous avons axé nos recherches à l'élaboration d'heuristiques avec des performances garanties. Récemment nous avons orienté nos recherches sur la construction de schémas d'approximation polynomiaux et schémas d'approximation complètement polynomiaux. Dans le chapitre suivant nous axons la présentation sur le développement de méthodes exactes de type séparation et évaluation pour la résolution des problèmes d'ordonnancement en présence de contraintes détaillées dans le chapitre 2.

5

Méthodes exactes de résolution

5.1 Introduction

Dans le chapitre 4, nous avons essentiellement présenté des heuristiques avec garantie de performances pour la résolution des problèmes NP-difficiles. Dans ce chapitre, nous mettons l'accent sur la résolution exacte des ces problèmes, cela en utilisant des méthodes de type séparation et évaluation. Nous rappelons que ces méthodes de complexité exponentielle ne sont utilisables que pour des problèmes de taille 'raisonnable'. Le challenge est alors de les utiliser pour des problèmes les plus grand possibles en un temps raisonnable. Nous commençons ce chapitre par une présentation du schéma général des méthodes de type séparation et évaluation, puis nous passons en revue les différents schémas mis-en-oeuvre pour la résolution des problèmes d'ordonnancement en présence de contraintes de groupement des tâches, de contraintes d'écarts temporels et de contraintes de détérioration des tâches.

5.2 Schéma général

Les procédures de type séparation et évaluation - PSE (*Branch and Bound* en anglais) sont des méthodes génériques de résolution des problèmes d'optimisation combinatoires. Ces méthodes reposent sur une énumération implicite de l'ensemble des solutions réalisables. Ces méthodes sont dotées de moyens qui permettent d'éviter l'examen de sous-ensembles de solutions qui ne peuvent contenir de solution optimale. Parmi ces moyens deux principes sont utilisés : la séparation et l'évaluation. La séparation procède à une décomposition de l'ensemble global des solutions en plusieurs sous-ensembles, telle que tous les sous-ensembles forment un recouvrement ou une partition de l'ensemble intial. Ainsi, en calculant la solution de chaque sous-ensemble puis en choisissant la meilleure, on est assuré d'avoir résolu le problème initial. Si une solution d'un sous-ensemble est difficile à trouver, on procède alors à la séparation de ce sous-ensemble, ainsi de manière récursive, jusqu'à aboutir : (i) soit à un sous-ensemble dont la solution optimale est facile à calculer (ii) soit à un sous-ensemble de solutions non intéressant à explorer. Quant à l'évaluation, l'objectif principal est d'éviter l'énumération systématique de toutes les solutions, par l'établissement de l'optimum d'un sous-ensemble de solutions ou de montrer que cet ensemble ne contient pas de solution intéressante. Dans les deux cas il est inutile d'effectuer

la séparation de l'espace des solutions. Pour montrer qu'un sous-ensemble ne contient pas de solution optimale, la méthode la plus utilisée consiste à calculer une borne inférieure pour les solutions contenues dans cet sous-ensemble (on s'intéresse ici à un problème de minimisation), puis si cette borne inférieure est strictement plus grande (resp. supérieure ou égale) que la meilleure solution réalisable trouvée (resp. strictement meilleure que la meilleure solution connue) jusqu'à présent, alors le sous-ensemble ne contient pas solution optimale et peut donc être éliminé. L'établissement de la borne inférieure passe souvent par une relaxation de certaines contraintes ou par la simplification du problème initial, par exemple en supposant qu'il y a une infinité de machines dans un problème d'ordonnancement. Il est aussi possible d'inclure dans les PSE des propriétés de dominance particulières au problème étudié. Par exemple, si pour un problème donné, une structure particulière est dominante, alors la recherche de la solution optimale peut se restreindre aux sous-ensembles contenant les solutions ayant cette structure, et alors on peut éliminer les autres sous-ensembles. Un dernier point concernant le schéma général des méthodes par séparation et évaluation est la stratégie de séparation. Autrement dit, comment choisir le sous-ensemble à séparer quand on est en face de plusieurs candidats. On distingue en général deux stratégies : (i) les PSE séquentielles, explorent les sous-ensembles dans un ordre établi a priori, par exemple en profondeur d'abord, ou en largeur d'abord. (ii) Les PSE progressives, où à chaque étape le sous-ensemble de plus petite borne inférieure est choisi. Une façon naturelle de représenter la recherche d'une solution consiste à représenter les sous-ensembles de solutions par un arbre, appelée arbre de recherche ou arbre de décision, où chaque noeud est associé à un sous-ensemble de solutions. Dans les sections 5.3, 5.4 et 5.5 nous présentons les différentes méthodes de type séparation et évaluation que nous avons développées pour les différentes contraintes traitées dans ce mémoire.

5.3 Groupement de tâches

Dans le cadre de la thèse de A. Bellanger, nous nous sommes intéressés à la résolution exacte du problème d'ordonnancement de la production dans un atelier de fabrication de pneumatiques. Nous avons considéré le problème de minimisation du makespan dans un flowshop hybride à deux étages. Le premier étage est composé de m_1 machines classiques en parallèle et le second contient m_2 machines à traitement par batches en parallèle.

Dans la littérature, plusieurs études sont consacrés à la résolution exacte de la minimisation du makespan dans un flowshop hybride en présence de machines classiques à chaque étage. Nous citons les travaux de Carlier et Néron [30], Mousli et Pichot [114]. Vignier et al. [150] et Kis et Pesch [92] ont successivement publié un état de l'art sur la modélisation et les différentes méthodes de résolution du flowshop hybride.

En présence de p-batch machines au second étage, nous avons développé dans [15] une méthode de type séparation et évaluation pour la minimisation du makespan. Elle consiste à ordonnancer les tâches séparément sur les deux étages. Autrement dit, cette PSE résout successivement le problème d'ordonnancement des tâches sur premier et sur le second étage. La première étape de la PSE traite le problème de machines parallèles au premier étage, générant à la fin de cette étape des ordonnancements non-dominés des tâches. La deuxième étape de la PSE utilise les dates de fin des tâches, dans les ordonnancements trouvés à la première étape, comme dates de disponibilité pour leur traitement sur le second étage. La décomposition en deux phases est motivée par le fait que à chaque phase on est amené à traiter un problème de machines parallèles possédant des propriétés

intéressantes.

Sur le premier étage, un ordonnancement σ tel que $\sigma \in \Sigma$ (Σ représente l'ensemble des ordonnancements non-dominés), est représenté par une permutation des tâches, obtenue par la numérotation des tâches dans ordre croissant de leur date de début d'exécution sur les machines. Réciproquement, à partir d'une permutation π donnée des tâches, on peut facilement obtenir les instants de début d'exécution des tâches en $O(n)$, cela en plaçant, à chaque fois, la première tâche de la permutation π sur la première machine disponible. Nous pouvons ainsi nous focaliser essentiellement sur la recherche de la meilleure permutation. Par ailleurs, comme toutes les machines sont initialement disponibles à l'instant $t = 0$, et de manière à éviter la représentation de deux ordonnancements identiques juste par une numérotation différente des machines, nous avons imposé un ordre sur le placement des m_1 premières tâches sur les machines, c'est-à-dire, pour deux tâches i et j telle que $i < j$, la tâche i est obligatoirement placée sur une machine de rang inférieur à celle de la tâche j.

Pour la première étape de la PSE, nous avons appliqué le schéma d'exploration en profondeur pour l'arbre de recherche. D'autre part, nous avons utilisé le schéma de séparation d'Ignall et Shrage [86] pour la première étape de la PSE, où à chaque noeud N_i de profondeur i de l'arbre de décision est associée une séquence partielle σ_i composée de i tâches. Le schéma de séparation consiste à séparer le noeud N_i en $n - i$ noeuds-enfants, où à chaque noeud-enfant une tâche est ajoutée à la fin de la séquence partielle σ_i, cette séquence partielle est composée de $i + 1$ tâches ordonnancées. Le dernier niveau de l'arbre de décision correspond aux séquences complètes des tâches. Pour la seconde étape de la PSE, un batch composé de tâches compatibles est ajouté à l'ordonnancement partiel, les comptabilités entre les tâches sont exploitées pour compléter des batches lorsque cela est possible. Nous avons développé trois bornes inférieures, deux bornes pour la première étape de la PSE et une borne pour la seconde étape. Considérons la première étape, soit N_1 le noeud courant et J_1 l'ensemble des tâches non encore ordonnancées. Notons par t_k la plus petite date de disponibilité des machines suite à l'ordonnancement des tâches N/J_1. La première borne LB_1 calcul la date de fin au plus tôt du traitement des tâches de l'ensemble J_1, i.e. $LB_1 = \max_{j \in J_1}\{t_k + p_j + a_j\}$. La deuxième borne concerne l'exécution des tâches de l'ensemble J_1 sur le second étage, i.e. $LB_2 = C_k + C^F_{max}(J_1)/m_2$, où C_k est la date de fin de la plus petite tâche de l'ensemble J_1 sur le premier étage et $C^F_{max}(J_1)$ est la somme des durées opératoires des batches obtenus en appliquant l'algorithme FCBLPT (voir la section 4.3.3 du chapitre 4) sur l'ensemble J_1 de tâches. La troisième borne concerne l'exécution des tâches sur le second étage. En effet, soit N_2 le noeud courant de la seconde étape de la PSE, J_2 les tâches non encore ordonnancées et t^*_k la plus petite date de disponibilité des machines du second étage. Chaque tâche j de l'ensemble J_2 dispose d'une date de fin C^1_j sur le premier étage, cette date représente l'instant au plus tôt à partir duquel cette tâche peut être traitée sur l'une des machines du second étage. Alors, la borne inférieure LB_3 calcule la plus petite date de fin d'exécution des tâches de l'ensemble J_2, i.e. $LB_3 = \max\{t^*_k, \min_{j \in J_2} C^1_j\} + C^F_{max}(J_2)/m_2$, où $C^F_{max}(J_2)$ est la somme des durées opératoires des batches obtenus en appliquant l'algorithme FCBLPT sur l'ensemble J_2 de tâches.

Enfin, nous avons initialisé la borne supérieure de la PSE par la meilleure solution obtenue en appliquant les heuristiques H_J, H_{LPT} et H_{LPBT} (voir la section 4.3.3 du chapitre 4). Nous avons évalué la performance de cette PSE par une série d'expériences. Nous avons généré 4 groupes de problèmes, chaque groupe contient 10 instances. Le nombre de tâches générées dans chaque groupe est initialisé à 12 pour les petites instances et à 100 pour les instances moyennes. Le nombre de machines à chaque étage dans chaque groupe varie entre 2 et 5 machines, et la capacité des p-batch machines

est égale à 2. Nous avons limité le temps de calcul à 10mn. Pour les instances de 12 tâches tous les problèmes sont résolus en quelques secondes, alors que très peu d'instances de 100 tâches sont résolues à l'optimum, mais les solutions obtenues, au bout du temps imparti à l'exécution de la PSE, sont généralement très performantes avec un écart moyen de 1% par rapport à la borne inférieure. Toutefois, le cas où le nombre de machines au premier étage est largement supérieur au nombre de machines au second étage, la PSE tronquée donne de mauvais résultats par rapport à la borne inférieure. Par ailleurs, nous avons constaté que les heuristiques utilisées pour initialiser la solution sont très performantes. Une analyse plus fine des résultats des expériences est présenté dans [15].

5.4 Ecarts temporels

Nous nous intéressons ici à la résolution exacte des problèmes de flowshop de permutation en présence de time-lags min et max. Etant donné un ensemble de n tâches où chaque tâche est soumise à des contraintes de time-lags min et max entre ses opérations sur le machines, le premier problème qui se pose est celui du placement de ces tâches pour une séquence donnée, telle que chaque tâche respecte ses contraintes de time-lags et que l'ordonnancement soit calé à gauche. Nous commençons cette section par la présentation de la méthode de placement des tâches, puis nous présentons les éléments communs aux méthodes exactes développées dans cette section, et enfin nous terminons cette section par une présentation de quelques problèmes particuliers.

Considérons le problème de timing ou de placement des tâches, c'est-à-dire, étant donnée une permutation, le problème consiste à déterminer les débuts d'exécution des tâches telles que toutes les contraintes de times-lags soient satisfaites et la valeur de l'objectif est minimale. En présence seulement de time-lags min, le problème de placement des tâches est évident, or l'ajout de time-lags max peut conduire à des difficultés de placement comme il est signalé dans [53]. Dans [60] nous avons proposé un algorithme de placement optimal pour une séquence donnée des tâches. L'algorithme procède comme suit,

1. Soit L la liste des tâches à placer. La première tâche est placée sur les machines le plus tôt possible tout en respectant les time-lags min entre les opérations.
2. Tant que L n'est pas vide, prendre la première tâche et la placer le plus tôt possible sur toutes les machines, en ne tenant compte que des contraintes de succession des opérations et des time-lags min, puis, vérifier si toutes les contraintes de time-lags max sont respectées entre chaque couple d'opérations successives. Si, pour un couple donné $(k, k+1)$ la contrainte de time-lag max n'est pas respectée, alors déplacer à droite le placement de la tâche sur la machine k, jusqu'à ce que la contrainte de time-lag max soit vérifiée.

Afin de minimiser le nombre de décalages imputables aux contraintes de time-lags max, l'étape 2. de l'algorithme teste les machines dans l'ordre inverse de leur enchaînement, ainsi chaque opération n'est retardée qu'au plus une fois. Cette algorithme donne une valeur optimale pour tout critère régulier en $O(n.m)$. Par ailleurs, cet algorithme peut être aisément modifié en présence de time-lags généralisés où ils sont définis entre tout couple d'opérations quelconques au sein des tâches, la modification de l'algorithme fournit un placement optimal en $O(n^2.m)$. Néanmoins, si on ne se restreint pas aux ordonnancements de permutation le problème de placement des tâches est NP-difficile pour le cas de deux machines et time-lags max uniquement, sachant que la permutation est donnée que sur la première machine.

Comme la recherche d'un ordonnancement optimal qui minimise un critère régulier donné passe obligatoirement par l'application de l'algorithme de placement, alors on ne peut pas dissocier le problème de placement des tâches du problème de leur séquencement. Dans la suite de cette section, un ordonnancement inclut à la fois l'ordre de passage des tâches et leur placement.

Dans les PSE développées, nous avons utilisé le schéma classique de séparation proposé par Ignall et Schrage [86]. La solution est représenté par une permutation de l'ensemble des tâches, où à chaque noeud N_i de profondeur i de l'arbre de décision est associée une séquence partielle σ_i composée de i tâches. Nous avons utilisé deux stratégies pour l'exploration de l'arbre de décision : (i) la première stratégie parcours l'arbre en profondeur, où parmi les noeuds issus de la séparation d'un noeud-père, celui ayant la plus petite borne inférieure est choisi pour le traitement. Si aucun noeud-enfant ne peut conduire à une solution optimale, on effectue un retour arrière jusqu'au niveau où il reste des noeuds à considérer (ii) la deuxième stratégie parcours l'arbre en largeur, elle explore tous les noeuds d'un même niveau avant de passer au niveau suivant. A noter que cette stratégie est moins efficace en pratique, car toute solution réalisable (voir optimale) n'est obtenue qu'à la fin du développement de l'arbre de décision, donc il y a peu de branches de l'arbre qui sont tronqués même si on connaît a priori une très bonne solution grâce à des méthodes approchées . Néanmoins, cette stratégie peut être utilisée dans une méthode de séparation et évaluation tronquée de type *beam search*, où parmi tous les noeuds générés à un niveau i seuls quelques uns de meilleures bornes sont gardés, [50] présente en détail le fonctionnement du beam search. Dans la suite de cette section nous détaillons les PSEs developpées pour les critères du makespan et du plus grand retard.

5.4.1 Minimisation du makespan

Le premier problème que nous considérons ici, concerne la minimisation de la durée totale de l'ordonnancement dans un flowshop de permutation avec m machines et des time-lags min et max entre les opérations de chaque tâche. Nous présentons ici les éléments spécifiques à la méthode de séparation et évaluation, notamment les bornes inférieures, bornes supérieures et règles de dominance.

Soit N_i un noeud de profondeur i de l'arbre de décision contenant i tâches ordonnancées dans la séquence partielle σ_i. Nous avons développé quatre bornes inférieures pour les tâches non encore ordonnancées du noeud N_i. Les deux dernières bornes sont les plus intéressantes, néanmoins, pour la compréhension des deux dernières, nous présentons d'abord les deux premières bornes. La première borne consiste à ordonnancer toutes les tâches restantes le plus tôt possible, cela en relaxant les contraintes de time-lags et de succession entre les opérations. La deuxième borne procède comme la première borne sauf pour la dernière tâche, où son placement respecte les contraintes de succession de ses opérations et de ses time-lags. La meilleure valeur de cette borne (la plus petite) est obtenue en testant toutes les tâches restantes en dernière position. La troisième borne est plus précise que la seconde, plutôt que de se contenter d'additionner les durées des tâches restantes pour définir un profil sur lequel la dernière tâche est calée, on relaxe uniquement les contraintes de time-lags max entre les deux premières machines (sauf pour la tâche en dernière position) et on applique l'algorithme de Mitten [105]. Celui-ci détermine un ordonnancent optimal pour le flowshop de permutation à deux machines avec time-lags min. En calant l'ordonnancement donné par cette règle sur l'ordonnancement partiel associé à σ_i, on obtient une plus grande date de début au plus tôt pour la dernière tâche sur la deuxième machine. La quatrième borne est obtenue en généralisant l'idée de la troisième borne à tous les couples de machines consécutives et en gardant la plus élevée des valeurs obtenues. Il est clair que ces quatre bornes peuvent être classées comme suit, $LB_1(N) \leq LB_2(N) \leq LB_3(N) \leq LB_4(N)$,

mais les bornes les plus sophistiquées demandent plus de temps de calcul à chaque noeud, il faut donc que ce temps perdu soit compensé par les sous-ensembles éliminés plus tôt.

Pour affiner l'arbre de décision, nous avons incorporé une règle de dominance à la racine de l'arbre. Cette règle consiste à comparer chaque couple de tâches (i, j). Si par exemple, pour toutes les machines la date de fin du couple (i, j) est plus petite que (j, i) alors les ordonnancements commençant par (j, i) sont dominés par ceux commençant par (i, j).

Pour l'initialisation de la PSE, nous avons développé cinq heuristiques pour l'initialisation de la PSE. La première heuristique H_1 est une heuristique gloutonne qui ajoute à chaque fois une tâche qui minimise la durée totale des tâches ordonnancées. La second heuristique procède de la même manière que la première, sauf qu'elle minimise les temps morts sur toutes les machines. La troisième heuristique est une adaptation de la méthode CDS [29] développée pour le flowshop classique. Nous avons adapté cette méthode pour satisfaire les contraintes de time-lags. La dernière heuristique utilise le principe de la méthode NEH [118] pour le problème flowshop de permutation à m machines. Nous avons utilisé deux listes initiales pour l'ordre de traitement des tâches, la première liste classe les tâches dans l'ordre croissant de leur durée de traitement totale $\sum_k p_{k,j}$ et la deuxième liste prend les tâches dans l'ordre de leur longueur totale $\sum_k p_{k,j} + \underline{\theta}_{k,j}$.

Pour l'expérimentation de la PSE développée, une considération spécifique est apportée à la génération des données. A travers ces données, nous avons tenté de traduire le plus possible des situations réalistes cela en prenant en compte la charge des machines, les caractéristiques des durées opératoires des tâches, le type de time-lags et leur amplitude. Nous avons généré 16 groupes de problèmes et chaque groupe contient 10 instances. Le nombre de tâches généré dans chaque groupe varie entre 12 et 18 tâches et le nombre de machines est pris entre 5 et 10, les caractéristiques de chaque groupe sont détaillées dans Fondrevelle et al. [62]. A chaque fois la PSE est testée par l'une des bornes LB_2, LB_3 et LB_4. Une analyse rapide des résultats des expériences numériques montre que (i) toutes les instances générées sont résolues en quelques secondes par la PSE (ii) les heuristiques utilisées pour l'initialisation de la PSE sont performantes notamment l'heuristique NEH, cette heuristique donne une déviation moyenne de 2.3% par rapport à l'optimum et ne dépasse pas 5.3% pour les plus mauvais résultats (iii) la PSE utilisant LB_2 est plus rapide en temps de calcul par rapport aux PSE utilisants LB_3 et LB_4, néanmoins, le nombre de noeuds développé est très élevées pour LB_2 que pour LB_4. En conclusion, les expériences numériques nous ont permis d'estimer l'apport de chacun des éléments developpés pour la résolution exacte. Nous avons constaté que la PSE la plus performante est celle qui utilise la stratégie de recherche en profondeur et comme borne inférieure LB_2. Une analyse plus fine des résultats des expériences sont données dans [60].

5.4.2 Minimisation du plus grand retard

Nous avons considéré le problème de minimisation du plus grand retard sur le flowshop à deux machines avec un traitement sans attente des tâches et des temps de montage et de démontage indépendants de la séquence. Ce problème nous été présenté par le Pr. Allahverdi lors de sa visite à Nancy en 2004. Les problèmes d'ordonnancement avec temps de montage et démontage sont étudiés dans la littérature indépendament des time-lags, nous avons cherché ici à utiliser les times-lags pour la modélisation et la résolution de ces problèmes. Allahverdi et al. [7] ont présenté un état de l'art complet sur ces problèmes d'ordonnancement avec des temps de montage et de démontage des tâches.

Remarquons en premier que seulement les ordonnancements de permutation sont réalisables pour tout problème sans attente. Nous avons modélisé les opérations de montage et de démontage par des time-lags exacts. En fait pour chaque tâche j, on note par $S_{1,j}$ et $S_{2,j}$ les temps de montage de la tâche j respectivement sur la première et la deuxième machine, et par $R_{1,j}$ et $R_{2,j}$ les temps de démontage de la tâche j respectivement sur la première et la deuxième machine. Pour transformer ce problème en problème d'ordonnancement avec time-lags, il suffit d'incorporer les temps de montage et de démontage dans les durées opératoires des tâches, puis de définir correctement un time-lag min autorisant un chevauchement partiel (pour garantir la précédence entre les parties correspondant aux durées opératoires) entre les opérations des tâches, et de tenir compte des dates de fin des tâches, ainsi le time-lag min pour la tâche j est égale à $\underline{\theta}_j = -R_{1,j} - S_{2,j}$.

Comme le problème de minimisation de plus grand retard pour le flowshop sans attente est déjà NP-difficile, Roeck [137], alors notre problème l'est également. Toutefois, nous avons proposé deux cas particuliers qui sont polynomiaux à savoir, (i) le cas où $\forall j, R_{2,j} \leq R_{1,j} - p_{2,j}$ et $S_{2,j} \leq S_{1,j} + p_{1,j}$, ici la séquence optimale est donnée par l'ordre croissant des $R_{1,j} - p_{2,j} + d_J$, ce cas représente en fait la situation où la la première machine domine (en terme de temps de charge à réaliser) la seconde (ii) le cas inverse du premier, i.e. $\forall j, R_{2,j} \geq R_{1,j} - p_{2,j}$ et $S_{2,j} \geq S_{1,j} + p_{1,j}$ ici la séquence optimale est obtenue en plaçant les tâches dans l'ordre croissant des $R_{2,j} + d_j$.

Comme le problème de temps de montage et de démontage peut être modélisé avec des time-lags, il est toute à fait naturel d'utiliser le schéma de la PSE présenté dans la section précédente. Nous avons considéré le schéma basé sur la stratégie d'exploration en profondeur. Nous avons proposé une borne inférieure LB spécifique au critère du retard L_{max}. En effet, notons par σ la séquence partielle des tâches ordonnancées et $L_{max}(\sigma)$ le plus grand retard de cette séquence. Soit $C_{2,\sigma(j)}$ la date de fin du démontage de la tâche en position j sur la deuxième machine. Il est évident que si une tâche est ajoutée à la position k ($k \geq i+1$) de la séquence partielle, on a $C_{2,\sigma(k)} \geq C_{2,\sigma(j)} + \sum_{i+1 \leq l \leq k}(S_{2,l} + p_{2,l} + R_{2,l})$, et comme $L_{\sigma(k)} = C_{2,\sigma(k)} - R_{2,\sigma(k)} - d_{\sigma(k)}$ (où $L_{\sigma(k)}$ représente le retard de la tâche de la position k de σ) alors, $L_{max} \geq LB = \max\{L_{max}(\sigma); \max\{C_{2,\sigma(k)} - R_{2,\sigma(k)} - d_{\sigma(k)} | i+1 \leq k \leq n\}\}$. La valeur de $C_{2,\sigma(k)}$ est bornée inférieurement par $\overline{C}_{2,\sigma(k)}$ où $\overline{C}_{2,\sigma(k)}$ est la date de fin de la tâche à la position k quand toutes les tâches restantes sont placées dans l'ordre croissant des $(S_{2,l} + p_{2,l} + R_{2,l})$. Soit maintenant la séquence ϕ des $n-i$ tâches restantes, tel que $R_{2,\phi(1)} + d_{\phi(1)} \leq \ldots \leq R_{2,\phi(n-i)} + d_{\phi(n-i)}$. Comme $\overline{C}_{2,\sigma(k)} \leq \ldots \leq \overline{C}_{2,\sigma(n)}$, alors une borne inférieure pour le plus grand retard est obtenue en soustrayant à chaque $\overline{C}_{2,\sigma(k)}$ la valeur de $R_{2,\phi(k-i)} + d_{\phi(k-i)}$et en gardant la plus petite valeur, i.e. $LB = \max\{L_\sigma; \max\{\overline{C}_{2,\sigma(k)} - R_{2,\phi(k-i)} - d_{\phi(k-i)} | i+1 \leq k \leq n\}\}$. Le dernier point particulier à cette PSE concerne la solution initiale du problème. Pour cela, nous avons développé trois variantes de l'heuristique NEH [118]. La première heuristique $NEH(1)$ considère les tâches dans l'ordre décroissant de leur durée totale en incluant les temps de montage et de démontage. Les deux autres heuristiques $NEH(2)$ et $NEH(3)$ réitèrent le principe de NEH cinq fois (le nombre cinq est obtenu expérimentalement), c'est à dire : à chaque itération du principe de NEH, le résultat trouvé est utilisé pour initialiser l'itération suivante, et à la fin on garde la meilleur solution trouvée par les cinq itérations. L'heuristique $NEH(2)$ considère la même liste initiale (pour la première itération) que l'heuristique $NEH(1)$ alors que l'heuristique $NEH(3)$ considère comme liste initiale l'ordre croissant des dates de fin souhaitées, augmentées du temps de démontage, i.e. $R_{2,j} + d_j$. Il est clair que l'heuristique $NEH(2)$ fournit une meilleure solution que $NEH(1)$ car la solution trouvée par $NEH(1)$ est aussi considérée par $NEH(2)$.

Pour les expériences numériques, nous avons généré des instances de 16 tâches et nous avons limité

le temps d'exécution de la PSE à 20mn. Mise à part quelques instances, la plupart sont résolues à l'optimum en moins de 180 secondes. Nous avons aussi constaté que l'heuristique $NEH(2)$ donne des résultats plus performants que les deux autres heuristiques. En effet, l'utilisation itérative du principe de NEH permet d'améliorer la qualité de la solution de manière significative. Le détail de l'analyse des expériences numériques peut être trouvé dans [61].

Pour le problème général à m machines, nous avons considéré, Fondrevelle et al. [63], le problème de flowshop de permutation avec des time-lags exacts et le critère du plus grand retard à minimiser. Les time-lags peuvent être positifs ou négatifs. Nous rappelons que la présence d'au moins un time-lag positive rend les ordonnancements de permutation non dominants. Néanmoins, nous nous intéressons à la recherche d'un ordonnancement de permutation. En premier nous pouvons constater que grâce aux time-lags exacts la valeur du critère pour chaque tâche peut être exprimé sur toute les machines et non pas uniquement sur la dernière machine, pour cela il suffit de définir une date de fin souhaité $d_{k,j}$ sur chaque machine k, où $d_{k,j} = d_j - \sum_{i=k}^{m-1} \theta_{i,j} + p_{i+1,j}$. De plus, suivant les valeurs des time-lags exactes, nous distinguons trois classes de tâches,

1. Classe couvrante : contient les tâches pour lesquelles il existe une machine telle que la période de traitement sur n'importe quelle autre machine, est inclue dans la période de traitement de la machine k (voir figure 5.1).
2. Classe non-couvrante : contient les tâches pour lesquelles les périodes de traitement sur les machines sont disjoints (voir figure 5.2).
3. Classe partiellement couvrante : Contient les tâches qui n'appartiennent pas aux deux classes précédentes (voir figure 5.3).

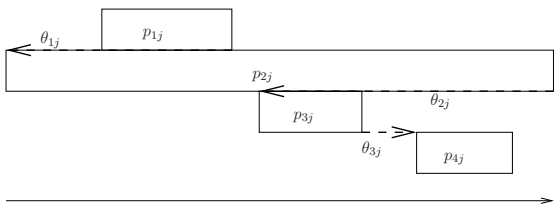

FIG. 5.1 – Tâche couvrante

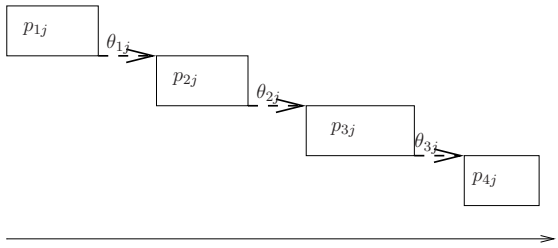

FIG. 5.2 – Tâche non-couvrante

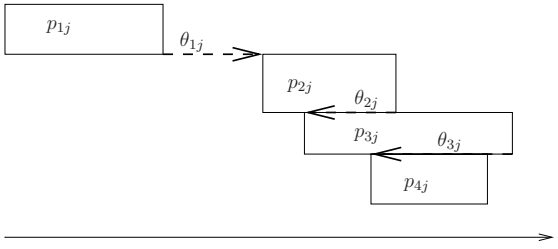

FIG. 5.3 – Tâche partiellement couvrante

A partir de la répartition des tâches en classes, nous avons établi le cas suivant,

Théorème. 8 *Etant donnée une machine k, si toutes les tâches sont de la classe couvrante de la machine k, alors un ordonnancement optimal est obtenu par la règle EDD sur les dates de fin souhaitées $d_{k,j}$ de la machine k.*

Nous avons employé le même schéma de séparation et évaluation précédent pour résoudre le problème à m machines de manière optimale. Nous avons utilisé une stratégie d'exploration en profondeur des noeuds de l'arbre de décision. Nous avons élaboré une borne inférieure LB plus précise que celle qui a été composé pour le problème à deux machines. La borne LB est fournie par la plus grande borne $LB_1, \ldots, LB_m$, où chaque LB_k est une borne du plus grand retard sur la machine k, obtenue en relaxant les contraintes de capacité et de succession sur les autres machines. Ainsi, la borne LB_k est obtenue en ordonnançant les tâches restantes sur la machine k dans l'ordre croissant des $d_{j,k}$. Pour le calcul de la solution initiale, nous avons développé quatre heuristiques. La première H_1 applique la règle EDD sur les $d_{k,j}$ de chaque machine k, puis elle construit les m ordonnancements correspondants et choisit la meilleure solution. Les trois autres heuristiques sont basées sur le schéma de NEH. Plus précisément elles répètent le schéma NEH cinq fois, et les listes initiales sont basées respectivement sur la durée totale de la tâche, la durée totale de la tâche augmentée de time-lag et l'ordre fournie par l'heuristique H_1. Nous avons testé cette approche de résolution sur des instances générées aléatoirement. Nous avons considéré 13 groupes de problèmes comportants chacun 10 instances. Comme toutes les instances générées font partie de la classe partiellement couvrante, nous avons ajouté deux groupes de tâches pour les classes couvrantes et non couvrantes. Nous avons constaté que l'heuristique H_1 est la moins bonne des quatre heuristiques développées, même l'heuristique NEH basée sur la liste initiale fournie par H_1 est légèrement moins bonne que les deux autres heuristiques basées sur le schéma de NEH. Aussi pour évaluer l'efficacité de la PSE nous avons limité le temps d'exécution à 20mn. En comparant les résultats de cette PSE avec celle développée pour le cas de deux machines, nous avons constaté une nette amélioration des temps de calcul, cette efficacité est partiellement due à l'amélioration de la borne inférieure. Une analyse plus détaillée des résultats des expériences peut être trouvée dans [60].

5.5 Détérioration des tâches

Nous avons considéré dans [121] le problème d'ordonnancement de détérioration des tâches sur deux machines et l'objectif de minimiser la durée totale de l'ordonnancement. Les durées opératoires des tâches sur la deuxième machine sont données par les b_j $j = 1, \ldots, n$ et sur la première machine sont données par des fonctions continues croissantes $a_j + F_j(\delta_j)$ où δ_j représente l'intervalle de temps qui sépare le début de l'exécution de la tâche j sur la deuxième machine et la fin de j sur la première machine. Les durées opératoires a_j et b_j sont dites durées opératoires intrinsèques. Nous avons développé une méthode exacte pour la détérioration linéaire des durées opératoires. Cette PSE comporte trois parties, l'initialisation, la séparation et l'évaluation. L'initialisation de la solution de départ se fait entre autre par deux heuristiques H_j et H_{GG} qui sont respectivement l'application de l'algorithme de Johnson [88] et de l'algorithme de Gilmore-Gomory [70] sur les durées intrinsèques. La méthode de séparation développe un ordre de recherche, où les noeuds sont explorés par la stratégie 'meilleur d'abord' et à chaque noeud, une tâche est ajoutée à la fin de l'ordonnancement partiel correspondant. Pour un noeud donnée N de l'arbre de décision, une borne inférieure sur le placement des tâches non encore ordonnancées J peut être calculée en ordonnançant les tâches de J suivant la règle de Johnson [88] appliquée sur les durées intrinsèques des tâches. Bien évidement nous avons développé plusieurs propriétés de dominance utilisées pour éliminer les

ordonnancements partiels qui ne sont pas susceptibles de conduire à une solution optimale. Pour évaluer l'efficacité de la PSE développé, nous avons généré aléatoirement cinq groupes de problèmes avec des détériorations uniformément linéaires. Dans chaque groupe, 20 instances sont générées aléatoirement pour chaque valeur du coefficient de détérioration des tâches qui prend 10 valeurs différentes dans l'intervalle $[0.001, 10]$. En premier, on peut signaler que la PSE peut résoudre à l'optimum des instances contenant jusqu'à 15 tâches en temps raisonnable. Une analyse rapide des résultats nous amène à distinguer entre les instances avec des coefficients de détérioration très faibles ($0.001 \leq f \leq 0.01$) et les problèmes avec un coefficient $f > 0.01$. Pour les petites valeurs du coefficient de détérioration, l'heuristique H_J basée sur l'algorithme de Johnson domine l'heuristique H_{GG}. Toutefois, quand la valeur du coefficient de détérioration augmente l'heuristique H_{GG} prend le dessus sur l'heuristique H_J. Nous avons aussi constaté qu'il est très difficile de construire des heuristiques meilleures pour toute les valeurs du coefficient de détérioration. Une analyse plus fine des résultats des expériences numériques peut être trouvé dans [121].

5.6 Conclusion

Dans ce chapitre, nous avons considéré les approches exactes de type séparation et évaluation pour la résolution des problèmes d'ordonnancement en présence de contraintes de groupement de tâches, de time-lags et de détérioration des tâches. Pou chacune des contraintes étudiées, nous avons présenté une ou plusieurs méthodes exactes de type PSE. Pour chaque problème, nous avons spécifié les outils nécessaires pour le développement de PSE performantes. Notons que pour les problèmes d'ordonnancement avec groupement de tâches la construction de méthodes exactes est difficile du fait qu'il faut traiter à la fois la composition des batches et leur séquencement. Cette difficulté explique probablement le fait que la majorité des méthodes exactes développées dans la littérature concernent les ateliers à une machine. Nous avons exposé dans ce chapitre un exemple d'une PSE développé pour un flowshop hybride à deux étages. La difficulté de ce problème fait en sorte que la PSE développé résout seulement des instances jusqu'à 15 tâches. Il est clair que les performances des PSE diminuent quand la taille des problèmes augmente, cette méthode constitue une première tentative de résolution du problème, néanmoins, d'une manière générale, les PSE développées dans ce chapitre nous ont permis de tester les instances de petites tailles et de positionner les autres approches de résolution par rapport aux méthodes exactes. Il reste bien entendu à améliorer significativement les performances de certaines de ces méthodes de résolution et aussi la qualité des bornes.

6

Conclusion et perspectives de recherche

6.1 Conclusion générale

Dans ce mémoire, nous avons présenté une partie de nos travaux de recherche concernant les problèmes d'ordonnancement d'ateliers de type flowshop. Par soucis d'harmonisation de ce document, nous avons volontairement omis la description des autres travaux de recherche, comme les problèmes d'optimisation sur la maintenance (maintenance des systèmes critiques, ordonnancement de la maintenance), et les problèmes d'optimisation dans la chaîne logistique. L'étude du modèle de flowshop est motivée par les nombreuses applications industrielles décrites dans la littérature, ce modèle présente l'avantage d'être largement répondu dans les systèmes de production où tous les produits suivent la même gamme opératoire et toutes les machines traitent les produits dans le même ordre. Notre attention s'est portée sur l'intégration de contraintes industrielles dans le modèle de flowshop classique. Nous avons considéré trois extensions, à savoir : (i) la possibilité de groupement des tâches (ii) la présence des écarts temporels entre les opérations d'une tâche ou entre les tâches (iii) la prise en compte de la détérioration des durées de traitement des tâches. Chacune de ces extensions généralise le modèle classique, par exemple, la première extension relaxe la contrainte où chaque ressource ne peut traiter qu'une seule tâche à la fois, la second extension généralise les contraintes de précédence classique entre les opérations de la même tâche et la troisième et dernière extension introduit la possibilité que la durée opératoire d'une tâche soit exprimée par une fonction croissante dépendante de l'instant de début de la tâche. Comme le montre l'ensemble des applications mentionné dans le chapitre 2, ces extensions répondent avant tout à une nécessité réelle pour laquelle le modèle classique est loin d'être satisfaisant. Les items suivants récapitulent nos contribution à chacune des trois extensions.

- *Groupement des tâches* : Nous nous sommes intéressés à l'étude du modèle de flowshop en présence de machines à traitement par batches. Notre contribution s'est portée sur trois modèles, à savoir (i) le modèle de flowshop à deux, trois et m p-batch machines, incluant ou non la contrainte de traitement sans attente entre les batches (ii) le modèle de flowshop contenant des machines à traitement par batches différentes, c'est-à-dire, le modèle

contenant à la fois une p-batch machine et une s-batch machine, et le modèle contenant plusieurs p-batch machines au premier étage et plusieurs machines classiques au second étage (iii) le modèle de flowshop en présence de contraintes de compatibilité entre les tâches. Nous avons étudié la complexité de plusieurs de ces modèles. Pour certains, nous avons proposé des algorithmes polynomiaux pour leur résolution, et pour d'autres qui sont NP-difficile, nous avons établi des approches de résolution basées sur des heuristiques avec des garanties de performance. Par ailleurs, pour les problèmes NP-difficiles, nous avons identifié plusieurs cas particuliers polynomiaux.

- *Ecarts temporels* : Les problèmes d'ordonnancement avec des écarts temporels entre les tâches ou entre les opérations d'une même tâche sont largement abordés dans la littérature. Les résultats existants dans la littérature sont essentiellement des résultats de complexité ou de méthodes de résolution approchées. Notre contribution à ce thème porte sur la mise en place de méthodes exactes de résolution (méthodes de type séparation et évaluation). Nous avons proposé un schéma générique de résolution, valable pour tous les problèmes de flowshop de permutation avec des écarts temporels, cela pour tout critère régulier à optimiser. Indépendamment du critère à optimiser, nous avons montré, pour une séquence de traitement donnée, comment obtenir un placement optimale des tâches en temps polynomial. La méthode de placement proposée nous a permis de ne considérer que la permutation des tâches et non pas leur positionnement temporel. Pour chaque critère que nous avons étudié, nous avons développé des bornes inférieures spécifiques et parfois des relations de dominance. Nous avons aussi évalué expérimentalement les approches proposées et également nous avons mis en évidence les limites de ces approches.
- *Détérioration de tâches* : La présence de la contrainte de détérioration des tâches rend rapidement les problèmes d'ordonnancement NP-difficiles, même ceux qui sont polynomiaux dans le cas classique, par exemple la minimisation du makespan dans un flowshop à deux machines et une détérioration linéaire des tâches. Deux modèles de détérioration sont proposés dans la littérature. Par exemple, pour le problème de flowshop, on trouve le premier modèle où la fonction de détérioration d'une tâche sur une machine donnée dépend de la date de début de cette tâche sur la première machine, alors que dans le second modèle la fonction de détérioration d'une tâche sur une machine donnée est exprimée en fonction de son début sur la machine encours et sa fin sur la machine précédente. Nous avons considéré le modèle de flowshop à deux machines pour le critère de la durée totale d'ordonnancement. Nous avons proposé une approche de résolution exacte de type séparation et évaluation, cela en spécifiant les bornes inférieure et supérieure, ainsi que les relations de dominance. Nous avons aussi considéré le critère du flot total. Pour ce dernier, le placement optimal des tâches n'est pas évident, nous avons proposé, pour une séquence donnée de tâches, le placement optimal qui minimise le flot total. Cette méthode est basée sur la notion de blocs et le placement optimal des blocs de tâches.

Comme le montre les travaux présentés dans ce mémoire, ainsi que les thèses encadrées, nos travaux de recherche sont particulièrement orientés vers l'intégration de nouvelles contraintes réelles et pertinentes dans les modèles classiques d'ordonnancement. A chaque nouveau modèle, nous nous attachons particulièrement à la compréhension de ces nouvelles contraintes et leurs impacts sur le modèle classique. Cette compréhension passe par l'utilisation des outils d'optimisation discrète, allant de l 'étude de la complexité jusqu'à la proposition d'approches de résolution.

6.2 Perspectives de recherche

Une gestion efficace de l'utilisation des ressources restera l'une des priorités principales de tout industriel ou fournisseur de service et ce dans la plupart des domaines. Il demeure ainsi une demande permanente en outils et méthodes d'aide à la décision pour une gestion efficace de ces ressources. Les méthodes d'optimisation et plus largement les outils de la recherche opérationnelle ont montré leur efficacité en apportant des réponses positives aux exigences des utilisateurs, néanmoins ces outils nécessitent une amélioration certaine et permanente due aux nouveaux développements et l'amélioration de la technicité des ressources.

Globalement, mes perspectives de recherche (à court et moyen terme) s'inscrivent dans un projet de recherche que j'intitule *'modèles et méthodes pour une gestion optimale des ressources'*. L'ambition principal de ce projet est le développement de nouveaux modèles et de méthodes d'aide à la décision pour une gestion efficace (au sens large) des ressources, et aussi pour répondre à plusieurs situations émergentes principalement dans le fonctionnement de la chaîne logistique tel que le partage des ressources entre plusieurs acteurs.

Au-delà des travaux développés jusqu'à présent sur les systèmes de production et de l'ordonnancement, où plusieurs questions restent sans réponses et laissent part à des travaux futurs (qui peuvent être des sujets de thèses potentiels) ; j'envisage, à travers ce projet, de mener des actions de recherche, non seulement sous l'angle de leur impact scientifique proprement dit, mais aussi sous l'angle de leurs applications industrielles. Le but ici n'est pas de dresser un panorama complet des problèmes que je souhaite aborder, mais d'identifier à travers quelques exemples les verrous scientifiques qui me semblent intéressants à lever, ou de proposer des solutions pour y remédier. Dans la suite, je développerai dans la section 6.2.1 certains points concernant les aspects théoriques liés à la complexité des problèmes d'optimisation et qui constituent une suite logique des travaux menés jusqu'à présent, puis dans la section 6.2.2 j'exposerai les actions de recherche qui me semblent intéressantes à mener dans la gestion des ressources en chaîne logistique.

6.2.1 Peut-on trouver ce qu'on peut affirmer ?

Cette question reste d'actualité en mathématiques discrètes. Pour certains problèmes, il est seulement possible d'affirmer une réponse donnée, sauf si P=NP. La question (à un million de dollars [1]), P=NP ? reste parmi les grands défis scientifiques de ce siècle pour son importance mais aussi pour ses conséquences dans divers domaines comme les mathématiques, l'informatique, la génomique, etc. Cette question occupera certainement, et pour plusieurs années, les spécialistes de la théorie de la complexité.

Parmi les actions de recherche, l'étude de la frontière entre les problèmes d'optimisation difficiles et les problèmes faciles permet d'approfondir un peu plus la complexité de ces problèmes. Cette étude représente un défi majeur pour la construction de nouveaux modèles et méthodes de résolution. Concrètement, en s'intéressant de près à certains problèmes d'optimisation, on constate que la complexité d'un problème change quand la valeur d'un certain paramètre change de valeur, particulièrement, le passage d'un paramètre de 2 à 3 fait basculer un problème dans la classe NP-Complet. Par exemple 2-SAT est polynomial alors que 3-SAT est NP-complet, de même 2-dimensional matching est polynomial tandis que le 3-dimensional matching est NP-complet. Bien que pour certains problèmes la séparation entre les cas faciles et les cas difficiles

[1] Millenium prize problems, http ://www.claymath.org/millennium/index.php

est nette, parfois il n'est pas évident d'appréhender le basculement d'un problème d'une classe de complexité à une autre. L'objectif est d'analyser ces problèmes de manière globale sans se focaliser sur les cas faciles et les cas difficiles séparément et surtout d'identifier les caractéristiques de ces problèmes. Une telle analyse peut contribuer à construire des méthodes avec une performance 'contrôlable' telle que pour les cas polynomiaux ces méthodes donnent une solution optimale et pour les cas difficiles, elles bornent la valeur du pire cas, cela en restant des méthodes polynomiales.

Dans le même ordre d'idée, on sait que la résolution des problèmes NP-difficiles, devienne rapidement intraitable, même pour des tailles raisonnables. Certes, les moyens techniques associés au perfectionnement des méthodes de résolution ont permis de franchir un palier important concernant la taille des problèmes traités, néanmoins il subsiste une limite infranchissable relative à la taille des instances, cela indépendamment des moyens techniques. Ainsi, approcher ces problèmes différemment devient une nécessité pour palier au problème de la taille des instances. Pour certains problèmes NP-difficiles, la difficulté de résolution est souvent due à un seul paramètre, c'est -à-dire la solution optimale peut être obtenue par un calcul exponentiel qui dépend (pour la partie exponentielle du calcul) d'un seul paramètre de ce problème. Il serait donc intéressant d'aborder ces problèmes autrement en étudiant leur complexité paramétrée (*Fixed parameter tractability*) [3]. Le but ici est d'effectuer une analyse plus fine des problèmes NP-difficiles en isolant la partie exponentielle du calcul pour faire émerger des méthodes exactes et efficaces pour certaines familles d'instances des problèmes NP-difficiles. Récemment, plusieurs études sont consacrées à la complexité paramétrée de différents problèmes, notamment en théorie des graphes. L'étude de la complexité paramétrée des problèmes de gestion des ressources est très récentes, quelques rares problèmes sont caractérisés, notamment les problèmes d'ordonnancement où la relation entre les tâches (contraintes de précédences entre les tâches, schéma de communication entre les tâches,...) est décrite par un graphe. L'étude de la complexité paramétrée des problèmes de gestion des ressources représente une piste de recherche très prometteuse dans le sens où elle permet d'identifier et d'isoler les parties qui rendent les problèmes difficiles, puis de proposer des méthodes efficaces de résolution en tenant compte de ces paramètres.

6.2.2 Quelles perspectives pour la gestion des ressources en chaîne logistique ?

L'intérêt de la communauté scientifique pour les problèmes opérationnels dans la chaîne logistique est relativement récent. Depuis environ une dizaine d'années, on assiste à l'apparition de nombreuses études intégrant à la fois les différents problèmes opérationnels (ordonnancement de la production, gestion des flux, livraison, stockage, transport, etc.) dans les mêmes modèles, [79], [98], [80]. Cette tendance s'explique par le fait que pendant longtemps la logistique était interne aux entreprises, réduite aux déplacements des produits à l'intérieur des sites de production ou entre les sites d'une même entreprise, et la fonction transport était juste considérée comme un maillon de la circulation des produits ; mais le changement stratégique dû à l'externalisation de la logistique a induit un besoin en optimisation globale pour le fonctionnement de la chaîne logistique, laissant de côté l'approche classique centrée sur le traitement séparé des problèmes sans tenir compte des interactions existantes entre ces problèmes.

Dans le cadre de la thèse de Zerouk Mouloua (thèse soutenue en Novembre 07), nous avons initié l'étude des problèmes opérationnels au sein de la chaîne logistique. Nous avons considéré

plusieurs problématiques liées à l'ordonnancement et au transport au niveau opérationnel de la chaîne logistique. Nous nous sommes intéressés à l'optimisation des coûts de production et de livraison à travers la coopération entre plusieurs entités de production. Nous avons développé des méthodes génériques d'optimisation des coûts en présence de plusieurs contraintes additionnelles liées aux capacités de transport et de production.

Ce projet de recherche s'inscrit dans la continuité des travaux réalisés jusqu'à maintenant. Les actions de recherche qu'on propose de mener cibleront principalement le niveau opérationnel de la gestion de la chaîne logistique, sans que ces actions soient déconnectées des aspects stratégiques (ou au moins tactiques) de la chaîne logistique. Avant de présenter les pistes de recherche qui nous semblent prometteuses, il est nécessaire d'évoquer ici quelques tendances lourdes dans lesquelles s'inscrit la logistique. Certainement, le but ici n'est pas de dérouler une liste exhaustive des tendances actuelles en logistique, mais à travers quelques exemples, on souhaite décrire les actions de recherche en s'appuyant sur ces exemples et par conséquent on identifie les défis scientifiques liés au fonctionnement de la chaîne logistique. Le point commun à toutes ces actions est l'élaboration de modèles et des méthodes d'aide à la décision pour l'optimisation du fonctionnement de la chaîne logistique.

1. Massification des flux : La massification des flux représente une des tendances actuelles des schémas logistiques. Le but de la massification des flux est d'aboutir aux coûts les plus bas possible tout en garantissant le même taux de service. Elle se met en œuvre par la mutualisation de l'entreposage et un recours massif aux plates-formes d'entreposage de plus en plus importantes et moins nombreuses. Néanmoins, ce développement a des effets négatifs sur le transport, il contribue à l'augmentation des distances par l'allongement des circuits logistiques ainsi que les parcours à vide. Le gain sur l'entreposage est en partie consommé par l'augmentation du coût de transport. Cette nouvelle organisation a débouché sur une gestion opérationnelle complexes de ces plates-formes. Elle nécessite la mise en place de méthodes d'optimisation efficaces pour la conception des tournées optimales de collectes des commandes à l'intérieur de la plate-forme et le remplissage optimale des moyens de transports en accords avec la planification des tournées de véhicules. Certainement ces problèmes sont combinatoires et difficiles à résoudre, on envisage ici comme action de recherche la conception de nouvelles méthodes d'optimisation globales qui prennent à la fois les problèmes externes à la plate-formes (tournées de véhicules, réduction des déplacement à vide, etc.) et les problèmes internes (préparation de la commande, remplissage des camions, tournées dans les allées, etc.).

2. Coordination entre les acteurs : Une autre tendance est la coordination entre les acteurs de la chaîne logistique. Elle prend différentes formes comme la mutualisation des moyens de transport ou de stockage. Le but est le partage et la réduction des coûts et de l'incertitude mais aussi le partage des risques logistiques. Dans certaines situations la coordination prend forme de coopération. L'étude et l'élaboration des modèles et des mécanismes de coordination représentent certainement l'un des thèmes les plus prometteurs en gestion efficace de la chaîne logistique. Jusqu'à présent la coordination se fait par la mutualisation des espaces de stockage ou de transport pour diminuer les coûts. Or, s'agissant du transport par exemple, les coûts baissent lorsque l'augmentation du volume à transporter permet de passer du mode routier à un mode adapté au fort tonnage. L'objectif ici est de modéliser les mécanismes de coopération entre les acteurs par la prise en compte du transport multimodal. Autrement dit, le but est le développement des modèles et de méthodes permettant d'avoir une vision globale et multimodale du transport des produits incluant les caractéristiques de chaque acteur de la

chaîne logistique ainsi que leur objectif, le but est aussi de montrer l'intérêt de la coopération tout en proposant des mécanismes d'incitation à la coopération.

3. La logistique urbaine : L'intensité des flux de marchandises dans les grandes agglomérations, associée aux réglementations de circulation et de déplacement des véhicules dans les grandes villes afin de réduire la congestion tend à redéfinir le mode de fonctionnement de la logistique urbaine. Actuellement les moyens de transport ont des charges moyennes très basses. Plusieurs modes de fonctionnement sont élaborés notamment par la création de centres de distribution urbains (CDU) à la périphérie ou de centres de distribution tampon à l'intérieur des villes notamment pour les fonctions de conditionnement, stockage, groupage/dégroupage, expéditions. Ces centres jouent un rôle primordial dans la majorité des schémas de la logistique urbaine, cela pour la jonction entre l'infrastructure de la logistique classique et le client final dans les grandes agglomérations. Le but de cette action de recherche est le développement de modèles et de méthodes pour la conception et la planification de la logistique urbaine aux niveaux tactique et opérationnel, par la prise en compte à la fois des problèmes d'allocation dynamique des ressources, de tournées de véhicules et la coopération entre les acteurs de la logistique urbaine.

En conclusion, à travers les actions décrites dans les sections 6.2.1 et 6.2.2, il est évident que ce projet de recherche porte à la fois sur les aspects théoriques et pratiques des problèmes d'optimisation et de gestion des ressources. La majorité des problèmes sont combinatoires et difficiles et certainement il n'existe pas de méthodes universelles pour tous ces problèmes, néanmoins une avancée théorique sur la compréhension de ces problèmes ouvrira certainement des perspectives non négligeables pour le traitement des problèmes industriels.

Bibliographie

[1] F. Afrati, E. Bampis, C. Chekuri, D. Krager, C. Kenyon, S. Khana, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko. Approximation schemes for minimizing average weighted completion time with release dates. In Proceedings of the 40th annual IEEE Symposium on Foundations of Computers Science, pages 32–43, 1999.

[2] J.H. Ahmadi, R.H. Ahmadi, S. Dasu, and C.S. Tang. Batching and scheduling jobs on batch and discrete processors. Operations Research, 39 :750–763, 1992.

[3] J.H. Ahmadi, R.H. Ahmadi, H. Matsuo, and D. Tirupati. Component fixture positioning/sequencing for printed circuit board assembly with concurrent operations. Operations Research, 43 :444–457, 1995.

[4] D. Ahr, J. Bekesi, and G. Galambos. An exact algorithm for scheduling identical coupled tasks. Mathematical Methods of Operations Research, 59 :193–203, 2004.

[5] S. Albers and P. Brucker. The complexity of one-machine batching problems. Discrete Applied Mathematics, 47 :87–107, 1993.

[6] B. Alidaee and N.K. Womer. Scheduling with time dependent processing times : review and extensions. Journal of Operational Research Society, 50 :711–720, 1999.

[7] A. Allahverdi, J.N.D. Gupta, and T. Aldowaisan. A review od scheduling involving setup considerations. OMEGA, 27 :219–239, 1999.

[8] A. Allahverdi, C.T. Ng, T.C.E. Cheng, and M.Y. Kovalyov. A survey of scheduling problems with setup times or costs. European Journal of Operational Research, 187 :985–1032, 2008.

[9] A. Bachman and A. Janiak. Minimizing maximum lateness under linear deterioration. European Journal of Operational Research, 126(3), 2000.

[10] K.R. Baker. Introduction to sequencing and scheduling. John Willey and Sons, 1974.

[11] K.R. Baker. Scheduling groups of jobs in two-machine flow shop. Mathematical and Computer Modelling, 13(3) :29–36, 1990.

[12] E. Balas, J.L. Lenstra, and A. Vazacopoulos. The one-machine problem with delayed precedence constraints and its use in jobshop scheduling. Management Science, 41(1) :94–109, 1995.

[13] P. Baptiste. Batching identical jobs. Mathematical Methods of Operations Research, 52 :413–420, 2000.

[14] M. Bartusch, R.H. Mohring, and F.J. Rademacher. Scheduling project networks with resource constraints and time windows. Annals of Operations Research, 16 :201–240, 1988.

[15] A. Bellanger and A. Oulamara. Exact method for hybrid flowshop with batching machines and tasks compatibilities. In Funda, editor, 11th International Workshop on Project Management and Scheduling - PMS2008, Istanbul, Turkey, 2008.

[16] A. Bellanger and A. Oulamara. Scheduling hybrid flowshop with parallel batching machines and compatibilities. Computers & Operations Research, In press, 2008.

[17] K.M.J. De Bontridder. Minimizing total weighted tardiness in a generatized job shop. Journal of Scheduling, 8 :479–496, 2005.

[18] M. Boudhar. Dynamic scheduling on a single batch processing machine with split compatibility graphs. Journal of Mathematical Modeling and Algorithms, 2 :17–35, 2003.

[19] M. Boudhar. Scheduling a batch processing machine with bipartite compatibility graphs. Mathematical Methods of Operations Research, 57 :513–527, 2003.

[20] M. Boudhar. Scheduling on a batch processing machine with split compatibility graphs. Journal of Mathematical Modeling and Algorithms, 4 :391–407, 2005.

[21] M. Boudhar and G. Finke. Scheduling on a batch machine with job compatibility. Belgian Journal of Operations Research, Statistics and Computer Science, 40 :69–80, 2000.

[22] S. Browne and U. Yechiali. Scheduling deteriorating jobs on a single processor. Operations Research, 38 :495–498, 1990.

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

[24] P. Brucker, T. Hilbig, and J. Hurink. A branch and bound algorithm for a single-machine scheduling problem with positive and negative time lags. Discrete Applied Mathematics, 94 :77–99, 1999.

[25] P. Brucker and S. Knust. Complexity results for single-machine problems with positive finish-start time-lags. Computing, 63 :299–316, 1999.

[26] P. Brucker, S. Knust, T.C.E. Cheng, and N.V. Shakhlevich. Complexity results for flowshop and open-shop scheduling problems with transportation delays. Annal of Operations Research, 129 :81–106, 2004.

[27] P. Brucker, S. Knust, and C. Oguz. Scheduling chains with identical jobs and constant delays on a single machine. Mathematics Methods of Operations Research, 63 :63–75, 2006.

[28] J.A Buzacott and J.R. Callahan. The complexity of the soaking pit-rolling mill complex in steel production. INFOR, 9(2), 1971.

[29] H.G. Campbell, R.A. Dudek, and M.L. Smith. A heuristic algorithm for the n-job, m-machine sequencing problem. Management Science, 16 :630–637, 1970.

[30] J. Carlier and E. Néron. An exact method for solving the multiprocessor flowshop. European Journal of Operational Research, 164 :592–608, 2005.

[31] A. Caumond, M. Gourgand, P. Lacomme, and N. Tchernev. Méthaheuristiques pour le problème de jobshop avec time lags : $j_m|l_{i,SJ(i)}|cmax$. pages 939–946. Cinquième Conférence Francophone de Modélisation et Simulation, 2004.

[32] A. Caumond, P. Lacomme, and N. Tchernev.

[33] B. Chen, C.N. Potts, and V.A. Strusevich. Approximation algorithms for two-machine flow shop scheduling with batch setup times. Mathematical Programming, 82 :255–271, 1998.

[34] Z.-L. Chen. Parallel machine scheduling with time-dependent processing times. Discrete Applied Mathematics, 70(1), 1996.

[35] Z.-L. Chen. Erratum : Parallel machine scheduling with time-dependent processing times. Discrete Applied Mathematics, 75(1), 1997.

[36] T.C.E. Cheng, Z.-L. Chen, M.Y. Kovalyov, and B.M.T. Lin. Parallel-machine batching and scheduling to minimize total completion time. IIE Transactions, 28 :953–956, 1996.

[37] T.C.E. Cheng and Q. Ding. The complexity of single machine scheduling with release dates. Information Processing Letters, 65 :75–79, 1998.

[38] T.C.E. Cheng, Q. Ding, and B.M.T. Lin. A concise survey of scheduling with time-dependent processing times. European Journal of Operational Research, 152 :1–13, 2004.

[39] T.C.E. Cheng, J.J.Yuan, and A.F. Yang. Scheduling a batch-processing machine subject to precedence constraints, release dates and identical processing times. Computers & Operations Research, 32(4) :849–859, 2008.

[40] T.C.E. Cheng and M.Y. Kovalyov. Single machine batch scheduling with sequential processing. IIE Transactions, 33(5) :413–420, 2001.

[41] T.C.E. Cheng, A. Toker, and B.M.T. Lin. Makespan minimization in the two-machine flowshop batch scheduling problem. Naval Research Logistics, 47 :128–144, 2000.

[42] T.C.E. Cheng and G. Wang. Batching and scheduling to minimize the makespan in the two-machine flowshop. IIE Transactions, 30 :447–453, 1998.

[43] H.N. Chiu and J.H. Chang. Cost models for lot streaming in a multistage flow shop. OMEGA, 33 :435–450, 2005.

[44] P. Chrétienne, E.G. Coffman, J.K. Lenstra, and Z. Liu. Theory of scheduling and its applications. John Wiley & Sons, Chichester, London, 1995.

[45] P. Chrétienne and C. Picouleau. Scheduling with communication delays : A survey. In P. Chrétienne et al., editor, Scheduling Theory and Its Applications, J. Wiley, 1995.

[46] C. Chu and J.-M. Proth. Single machine scheduling with chain structured precedence constraint and separation time windows. IEEE Transactions on robotics and automation, 12-6(6) :835–844, 1996.

[47] H.R. Lourenço. A polynomial algorithm for special case of the one-machine scheduling problem with time-lags. Technical Report 339, Economics Working Papers Series, Departement of Economics and business, University Pompeu Fabra, 1998.

[48] E.G. Coffman, A. Nozari, and M. Yannakakis. Optimal scheduling of products with two subassemblies on a single machine. Operations Research, 37 :426–436, 1989.

[49] E.G. Coffman, M. Yannakakis, M.J. Magazine, and C. Santos. Batch sizing and job sequencing on single machine. Annals of Operations Research, 26 :135–147, 1990.

[50] F. Della Croce and V. T'kindt. A recovring beam search algorithm for the one-machine dynamic total completion time scheduling problem. Journal of the Operational Research Society, 53(11) :1275–1280, 2002.

[51] M. Dell'Amico. Shop problems with two machines and time lags. Operations Research, 44(5) :777–787, 1996.

[52] M. Demange, D. De Werra, J. Monnot, and V. Paschos. Time slot scheduling of compatible jobs. Journal of Scheduling, 10(2) :111–127, 2007.

[53] F. Deppner. Ordonnancement d'atelier avec contraintes temporelles entre opérations. PhD thesis, Institut National Polytechnique de Lorraine, 2004.

[54] J. Edmonds. Matching and a polyhedron with 0,1 vertices. Journal of research N.B.S.B., 69 :125–130, 1965.

[55] G. Finke, M.-L. Espinouse, and H. Jiang. General flowshop models : job dependent capacities, job overlapping and deterioration. International Transactions in Operational Research, 9 :399–414, 2002.

[56] G. Finke and H. Jiang. A variant of the permutation flow shop model with variable processing times. Discrete Applied Mathematics, 76 :123–140, 1997.

[57] G. Finke, V. Jost, M. Queyranne, and A. Sebo. Batch processing with interval graph compatibilities between tasks. Discrete Applied Mathematics, 156 :556–568, 2008.

[58] G. Finke and A. Oulamara. Total completion time in a two-machine flowshop with deteriorating tasks. Journal of Mathematical Modeling and Algorithms, 6 :563–576, 2007.

[59] L. Finta and Z. Liu. Single machine scheduling subject to precedence delays. Discrete Applied Mathematics, 70 :247–266, 1996.

[60] J. Fondrevelle. Résolution exacte de problèmes d'ordonnancement de type flowshop de permutation en présence de contraintes d'écarts temporels entre opérations. PhD thesis, Institut National Polytechnique de Lorraine, 2005.

[61] J. Fondrevelle, A. Allahverdi, and A. Oulamara. Two-machine, no-wait flowshop scheduling problem to minimize maximum lateness with separate setup and removal times. International Journal of Agile Manufacturing, 8(2), 2005.

[62] J. Fondrevelle, A. Oulamara, and M.-C. Portmann. Permutation flowshop scheduling problems with maximal and minimal time lags. Computers and Operations Research, 33(6), 2006.

[63] J. Fondrevelle, A. Oulamara, M.-C. Portmann, and A. Allahverdi. Permutation flowshops with exact time lags to minimize maximum lateness. International Journal of Production Research, accepted, 2008.

[64] L.R. Foulds and J.M. Wilson. Scheduling operations for the harvesting of renewable resources. Journal of Food Engineering, 70 :281–292, 2005.

[65] M.R. Garey and D.S. Johnson. Computers and Intractability : A Guide to NP-completeness. W.H. Freeman, San Francisco, 1979.

[66] S. Gawiejnowicz. Brief survey of continuous models of scheduling. Foundations of Computer and Decision Science, 21 :81–100, 1996.

[67] S. Gawiejnowicz and L. Pankowska. Scheduling jobs with varying processing times. Information Processing Letters, 54(3), 1995.

[68] A.E. Gerodimos, C.A. Glass, and C.N. Potts. Scheduling the production of two-component jods on a single machine. European Journal of Operational Research, 120 :250–259, 2000.

[69] A. Gharbi and M. Houari. Minimizing makespan on parallel machines to release dates and delivery time. Journal of Scheduling, 5 :329–355, 2002.

[70] P.C. Gilmore and R.E. Gomory. Sequencing a one state variable machine : a solvable case of the traveling salesman problem. Operations Research, 12(5), 1964.

[71] C.A. Glass, C.N. Potts, and V.A. Strusevich. Scheduling batches with sequential job processing for two-machine flow and open shops. INFORMS Journal on Computing, 13 :120–137, 2001.

[72] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling theory : a survey. Annals of Discrete Mathematics, 5 :287–326, 1997.

[73] I.V. Gribkovskaia, C.-Y. Lee, V.A. Strusevich, and D. De Werra. Three is easy, two is hard : open shop sum-batch scheduling problem refined. Operations Research Letters, 34 :459–464, 2006.

[74] J.N.D. Gupta. Comparative evaluation of heuristic algorithms for a single machine scheduling problem with two operations per job and time-lags. Journal of Global Optimization, 9 :239–253, 1996.

[75] J.N.D. Gupta and S.K. Gupta. Single facility scheduling with nonlinear processing times. Computers an Industrial Engineering, 14 :387–393, 1988.

[76] L.A. Hall, A.S. Schulz, D.B. Shmoys, and J. Wein. Scheduling to minimize average completion time : offline and online algorithms. Mathematics of Operations Research, 22 :513–544, 1997.

[77] L.A. Hall and D.B. Shmoys. Approximation schemes for constrained for minimizing average weighted completion time with release dates. In Proceedings of the 30th annual IEEE Symposium on Foundations of Computers Science, pages 134–139, 1989.

[78] N.G. Hall, G. Laporte, E. Selvarajah, and C. Sriskandarajah. Scheduling and lot streaming in flowshops with no-wait in process. Journal of Scheduling, 6 :339–354, 2003.

[79] N.G. Hall and C.N. Potts. Supply chain scheduling : Batching and delivery. Operations Research, 51 :566–584, 2003.

[80] N.G. Hall and C.N. Potts. The coordination of scheduling and batch deliveries. Annals of Operations Research, 135 :41–64, 2005.

[81] R. Heilmann. A branch-and-bound procedure for the multi-mode resource-constrained project scheduling problem with minimum and maximum time lags. European Journal of Operational Research, 144 :348–365, 2003.

[82] D.S. Hochbaum and D. Landy. Scheduling with batching : Minimizing the weighted number of tardy jobs. Operations Research Letters, 16 :79–86, 1994.

[83] A. Hodson, A.P. Muhlemann, and D.H.R Price. A microcomputer based solution to a practical scheduling problem. Journal of the Operational Research Society, 36(10) :903–914, 1985.

[84] J.A. Hoogeveen and S.L. Van der Velde. Scheduling by positional completion times : Analysis of a two stage flow shop with a batching machine. Mathematical Programming, 82 :273–289, 1998.

[85] J. Hurink and J. Keuchel. Local search algorithms for a single-machine scheduling problem with positive and negative time-lags. Discrete Applied Mathematics, 112 :179–197, 2001.

[86] E. Ignall and L. Shrage. Application of the branch-and-bound technique to some flow-shop scheduling problems. Operations Research, 13 :400–412, 1965.

[87] R. Janczewski and M. Kubale. Scheduling unit execution time tasks with symetric time-lags. Journal of Applied Computer Systems, 9-2(2) :45–51, 2001.

[88] S.M. Johnson. Optimal two and three-stage production schedules with setup times included. Naval Research Logistics Quarterly, 1 :61–68, 1954.

[89] J. Kaabi-Harrath. Contribution à l'ordonnancement des activités de maintenance dans les systèmes de production. PhD thesis, Université de Franche-Comte, France, 2004.

[90] A.A. Kalir and S.C. Sarin. Constructing near optima schedules for the flow-shop lot streaming problem with sublot-attached setups. Journal of Combinatorial Optimization, 7 :23–44, 2003.

[91] K. Khurana and P.C. Bagga. Minimizing the makespan in a 2-machine flowshop with time lags and setup conditions. Zeitschrift fur Opertions Research, 28 :163–174, 1984.

[92] T. Kis and E. Pesch. A review of exact solution methods for the non-preemptive multiprocessor flowshop problem. RAIRO - Recherche Operationnelle - Operations Research, 78 :146–161, 2000.

[93] U. Kleinau. Two-machine shop scheduling problems with batch processing. Mathematical and Computer Modeling, 17 :55–66, 1993.

[94] A. Kononov and S. Gawiejnowicz. Np-hard cases in scheduling deteriorating jobs on dedicated machines. Journal of the Operational Research Society, 52(6), 1997.

[95] W. Kubiak and S.L. Van De Velde. Scheduling deteriorating jobs to minimize makespan. Naval Research Logistics, 45 :511–523, 1998.

[96] A.S. Kunnathur and S.K. Gupta. Minimizing the makespan with late start penalities added to processing times in a single facility scheduling problem. European Journal of Operational Research, 47(1), 1990.

[97] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. Sequencing and scheduling theory : algorithms and complexity. In S.C. Graves, P.H. Zipkin, and A.H.G Rinnooy Kan, editors, Handbooks in Operations Research and Management Science, North-Holland : Amesterdam, 1993.

[98] C.L. Li and W.Q. Xiao. Lot streaming with supplier-manufacturer coordination. Naval Research Logistics, 51 :522–542, 2004.

[99] S. Li, G. Li, X. Wang, and Q. Liu. Minimizing makespan on a single batching machine with release times and non-identical job sizes. Operations Research Letters, 33-2(2) :157–164, 2005.

[100] C.J. Liao and W.J. Chen. Single-machine scheduling with periodic maintenance and nonresumable jobs. Operations Research, 30(9) :1335–1347, 2003.

[101] C.K.Y. Lin and K.B. Haley. Scheduling two-phase jobs with arbitrary time lags in a single-server system. IMA Journal of Mathematics Applied in Business and Industry, 5 :143–161, 1994.

[102] C. Low, C. Hsu, and K.I. Huang. Benefits of lot spliting in job-shop scheduling. International Journal of Advanced Manufacturing Technology, 24 :773–780, 2004.

[103] P.L. Maggu, M.L. Singhal, N. Mohammad, and S.K. Yadav. On n-job, 2-machine flow-shop scheduling problem with arbitrary time lags and transportation times of jobs. Journal of the Operations Research Society of Japan, 25(3) :219–227, 1982.

[104] A. Mili, A. Oulamara, and M-C. Portmann. Ordonnancement conjoint de la production et de la maintenance. In FRANCORO V/ ROADEF 2007, editor, Livre articles, pages 207–215, Grenoble, France, 2007.

[105] L.G. Mitten. Sequencing n jobs on two machines with arbitrary time lags. Management Science, 5 :293–298, 1959.

[106] C.L. Monma and C.N. Potts. On the complexity of scheduling with batch set-up times. Operations Research, 37 :798–804, 1989.

[107] J.M. Moore. An n jobs, one machine sequencing algorithm for minimizing the number of late jobs. Management Science, 15 :102–109, 1968.

[108] G. Mosheiov. V-shape policies for scheduling jobsscheduling jobs under simple linear deterioration. Operations Research, 39 :979–991, 1991.

[109] G. Mosheiov. Scheduling jobs under simple linear deterioration. Computers & Operations Research, 21(6), 1994.

[110] G. Mosheiov. Scheduling jobs with step-deterioration : minimizing makespan on a single and multiple-machine. Computers and Industrial Engineering, 28(4), 1995.

[111] G. Mosheiov. Multiple-machine scheduling with linear deterioration. INFOR, 36(4), 1998.

[112] G. Mosheiov. Complexity analysis of job-shop scheduling with deteriorating jobs. Discrete Applied Mathematics, 117 :195–209, 2002.

[113] G. Mosheiov and D. Oron. A note on flow-shop batch scheduling with identical processing time jobs. European Journal of Operational Research, 161 :285–291, 2005.

[114] O. Mousli and Y. Pichot. A branch and bound algorithm forr the hybrid flowshop. International Journal of Production Research, 64 :113–125, 2000.

[115] A. Munier, M. queyranne, and A.S. Schulz. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. In A. Oulamara ans M.-C. Portmann, editor, Project Management and Scheduling, pages 367–382. Sixth International Integer Programming and Combinatorial Optimization Conference, 1998.

[116] A. Munier and F. Sourd. Scheduling chains on a single machine with nonnegative time lags. Mathematical Methods of Operations Research, 57 :111–123, 2003.

[117] D. Naddef and C. Santos. One-pass batching algorithms for the one-machine problem. Discrete Applied Mathematics, 21 :133–146, 1988.

[118] M. Nawaz, E.E. Enscore Jr., and I. Ham. A heuristic algorithm for the m-machine n-job flow-shop sequencing problem. OMEGA, 11(1) :91–95, 1983.

[119] C.T. Ng, T.C.E. Cheng, J.J. Yuan, and Z.H. Liu. On the single machine serial batching scheduling problem to minimize total completion time with precedence constraints, release dates and identical processing times. Operations Research Letters, 31 :323–326, 2003.

[120] A.J. Orman and C.N. Potts. On the complexity of coupled-task scheduling. Discrete Appied Mathematics, 72 :141–154, 1997.

[121] A. Oulamara. Flowshops avec détérioration de tâches et groupement de tâches. PhD thesis, Université Joseph Fourier, 2001.

[122] A. Oulamara. Makespan minimization in a no-wait flow shop problem with two batching machines. Computers & Operations Research, 34(4), 2007.

[123] A. Oulamara and G. Finke. No-wait flowshop problem with two batch processing machines. In Proceedings Second Conference IFAC/IFIP/IEEE on Management and Control of Production and Logistics (MCPL'2000), pages 663–668, Grenoble, France, 2000.

[124] A. Oulamara and G. Finke. Flowshop problems with two batch processing machines. International Journal of Mathematical Algorithms, 2 :169–287, 2001.

[125] A. Oulamara, G. Finke, and A. Kamgaing Kuiteing. Flowshop scheduling problem with a batching machine and task compatibilities. Computers and Operations Research, available online :October, 2007.

[126] A. Oulamara, G. Finke, and A. Kamgaing Kuiten. Flowshop scheduling problem with batching machine and task compatibilities. Computers & Operations Reseatch, 0ctober :Available online, 2007.

[127] A. Oulamara, M.Y. Kovalyov, and G. Finke. Scheduling a no-wait flowshop with unbounded batching machines. IIE Transactions on Scheduling and Logistics, 37(8), 2005.

[128] C.K. Poon and W. Yu. Minimizing total completion time in batch machine scheduling. International Journal of Fondations of Computer Science, 15(4) :593–607, 2004.

[129] C. N. Potts, V. A. Strusevich, and T. Tautenhahn. Scheduling batches with simultaneous job processing for two-machine shop problems. Journal of Scheduling, 4(1) :25–51, 2001.

[130] C.N. Potts and M.Y. Kovalyov. Scheduling with batching : A review. European Journal of Operational Research, 120(4) :228–249, 2000.

[131] C.N. Potts and L.N. Van Wassenhove. Integrating scheduling with batching and lotsizing : A review of algorithms and complexity. Operations Research Society, 42 :395–406, 1992.

[132] Rayward-Smith and D. Rebaine. Open-shop scheduling with delays. Informatique Théorique et Applications, 26 :439–448, 1992.

[133] Rayward-Smith and D. Rebaine. Uet flow shop scheduling with delays. Informatique Théorique et Applications, 30-1(1) :23–30, 1996.

[134] D. Rebaine and V. Strusevich. Two-machine open shop scheduling with special transportation times. Journal of Operational Research Society, 50 :756–764, 1999.

[135] J. Reizebos and G.J.C. Gaalman. Time lag size in multiple operations flow shop scheduling heuristics. European Journal of Operational Research, 105 :72–90, 1998.

[136] J. Reizebos, G.J.C. Gaalman, and J.N.D Gupta. Flow shop scheduling with multiple operations and time lags. Journal of Intelligent Manufacturing, 6 :105–115, 1995.

[137] H. Roeck. Some new results in flow shop scheduling. Zeitschrift fur Operations Reseatch, 28(1), 1984.

[138] B. Roy and M. Dibon. L'ordonnancement par la méthode des potentiels – le programme concord. Automatisme, 2 :1–11, 1966.

[139] B. Roy and B. Sussman. Les problèmes d'ordonnancement avec contraintes disjonctives. SEMA, Note N. 9bis, 1964.

[140] R. Ruiz, J.C. Garcia-Diaz, and C. Maroto. Considering scheduling and preventive maintenance in the flowshop sequencing problem. Computers & Operations Research, 34(11) :3314–3330, 2007.

[141] Y. Sekiguchi. Inter- and intra-goup-of-jobs schedule for minimizing makespan in a two-machine gt shop. In Control Science and Technology for the Progress of Society, pages 2021–2026. Procedings of the Eighth Triennial World Congress of the International Federation of Automatic Control, 1982.

[142] R.D. Shapiro. Scheduling coupled-tasks. Naval Research Logistics Quarterly, 28 :489–497, 1981.

[143] G.-J. Sheen and L.-W. Liao. A branch and bound algorithm for the one-machine scheduling problem with minimum and maximum time lags. European Journal of Operational Research, 181 :102–116, 2007.

[144] Y.N. Sotskov, T. Tautenhahn, and F. werner. On the application of insertion techniques for job shop problem with setup times. RAIRO Recherche Opérationnelle, 33 :209–245, 1999.

[145] F. Sourd. Optimal timing of a sequence of tasks with general completion costs. European Journal of Operational Research, 165 :82–96, 2005.

[146] C. Sriskandarajah and E. Wagneur. Hierarchical control of the two-processeur flow-shop with state dependent processing times. INFOR, 29 :193–205, 1991.

[147] W. Szwarc. The flow shop problem with time lags and separated setup times. Zeitschrift fur Opertions Research, 30 :15–22, 1986.

[148] V.S. Tanaev, Y.N. Sotskov, and V.A. Strusevich. Scheduling Theory. Multi-Stage Systems. Kluwer Academic Publishers, The Netherlands, 1994.

[149] L. Tang and Y. Zhao. Scheduling a single semi-continuous batching machine. OMEGA, 36-6(6) :992–1004, 2008.

[150] A. Vignier, J.-C. Billaut, and C. Proust. Les problèmes d'ordonnancement de type flow shop hybride : Etat de l'art. RAIRO - Recherche Operationnelle - Operations Research, 33(2) :117–182, 1999.

[151] E. Wagneur and C. Sriskandarajah. Optimal control of a class of deds : flow-shops with state-dependent processing times. Discrete Event Dynamic Systems : Theory and Applications, 3 :397–425, 1993.

[152] E. Wagneur and C. Sriskandarajah. The two-machine permutation flow shop with state-dependent processing times. Naval Research Logistics, 40 :697–717, 1993.

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

[154] E.D. Wikum, D.C. Llewellyn, and G.L. Nemhauser. One-machine generalized precedence constrained scheduling problems. Operations Research Letters, 16 :87–99, 1994.

[155] D.L. Yang and M.S. Chern. A two-machine flowshop sequencing problem with limited waiting time constraints. Computers and Indusrial Engineering, 88(1) :63–70, 1995.

[156] W. Yu. The two-machine flowshop problem with delays and the one-machine total tardiness problem. PhD thesis, Technische Universiteit Eindhoven, 1996.

[157] W. Yu, H. Hoogeveen, and J.K. Lenstra. Minimizing makespan in a two-machine flow shop with delays and unit-time operations is np-hard. Journal of Scheduling, 7 :333–348, 2004.

[158] J.J. Yuan, Z.H. Liu, C.T. Ng, and T.C.E. Cheng. The unbounded single machine parallel batch scheduling problem with family jobs and release dates to minimize makespan. Theoretical Computer Science, 320-2,3(2-3) :199–212, 2004.

[159] J.J. Yuan, A.F. Yang, and T.C.E. Cheng. A note on the single machine serial batching scheduling problem to minimize maximum lateness with identical processing times. European Journal of Operational Research, 158(2) :525–528, 2003.

Deuxième partie

Annexes

7

Curriculum Vitae Détaillé

7.1 Curriculum Vitae synthétique

Notice individuelle

Ammar Oulamara, né le 11/07/1973 en Algérie, marié

Statut	Maître de Conférence en Informatique Institut National Polytechnique de Lorraine - INPL Ecole des Mines de Nancy - ENSMN Laboratoire Lorrain de Recherche en Informatique et ses Applications - LORIA UMR 7503
Adresse	Bâtiment A, Laboratoire LORIA, Campus Scientifique - BP 239, 54506 Vandoeuvre-lès-Nancy Cedex et Ecole des mines de nancy, parc de saurupt CS 14234, 54042, Nancy.
Email	Ammar.Oulamara@loria.fr
Tel	+33(0)3 54 95 84 83
Fax	+33(0)3 83 58 43 28

Formation

1998-2001	Doctorat de l'Université Joseph Fourier. Thèse soutenue le 29 juin 2001. Sujet : Flowshops avec détérioration de tâches et groupement de tâches. Laboratoire Leibniz - IMAG, Grenoble. Directeur de thèse : Gerd Finke. Jury : J.C. Billaut, Ph. Chrétienne, M. Dror, L. Dupont, D. Trystram, E. Wagneur.
1996-1997	DEA de Recherche Opérationnelle, Combinatoire et Optimisation, INP - Grenoble

1991-1995	Ingénieur en Informatique - Recherche Opérationnelle Faculté de mathématiques, USTHB, Alger.

Activités professionnelles - synthèse

2007-	Titulaire de la prime d'encadrement doctoral et de recherche - PEDR
2002-	Maître de Conférences en Informatique à l'INPL - Ecole des Mines de Nancy Enseignement en recherche opérationnelle (théorie des graphes, programmation linéaire, programmation dynamique, chaînes de markov, files d'attente, complexité, metaheuristiques, etc.), en optimisation discrète et déterministe (programmation en nombre entiers, flots dans les réseaux, complexité des problèmes, méthodes de résolution exactes et approchées„ etc.), en Informatique (algorithmique et bases de données). Mise en place de cours de génie industriel et de gestion de production pour la filière FI-MGP (Formation d'Ingénieur - Matériaux et Gestion de Production) de l'Ecole des Mines de Nancy.
2001-2002	Ingénieur Expert - Recherche et Développement à la société Temposoft, spécialisée dans le développement d'outils de planification de personnel
2000-2001	Attaché Temporaire d'Enseignement et de Recherche à l'ENSIMAG (Ecole Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble). Enseignement en Recherche Opérationnelle et en Informatique - Bases de Données.

Principales responsabilités

2008- 2009	Président du comité d'organisation de la conférence ROADEF 09 (Nancy, 10-12 février 2009) conférence annuelle de la société française de recherche opérationnelle et d'aide à la décision.
2008-	Membre élu du Conseil Scientifique de l'INPL.
2006 - 2008	Membre élu du Conseil d'Administration de l'INPL.
2004-	Membre titulaire de la Commission de spécialistes, 27e section de l'INPL.
2003-2004	Co-responsable de la conférence PMS 04 (9th International Workshop on Project Management and Scheduling, Nancy, Avril 2004) organisation générale, co-responsable du budget, édition du livre des résumés et édition d'un numéro spécial d' EJOR (European Journal of Operational Research - Elsevier).
2003-2006	Responsable pédagogique du Mastère Génie Industriel - Aide à la décision pour les systèmes de production et de distribution (ancien mastère de Recherche Opérationnelle et Stratégie de Décision).

7.2 Activité d'enseignements et responsabilités pédagogiques

Mes différents enseignements sont dispensés à l'Ecole des Mines de Nancy aux filières FICM (Formation Ingénieur Civil des Mines) et FI-MGP (Formation d'Ingénieur - Matériaux et Gestion de Production) sous forme de cours magistraux et des TDs. Depuis mon recrutement à l'Ecole des Mines de Nancy en septembre 2002, j'effectue environ 220h équivalent TD chaque année de charge d'enseignement et d'encadrement. Dans la section qui suit, je détaille les principaux cours que j'ai assurés ou que j'assure actuellement, puis par la suite je donne une synthèse des activités et responsabilités liées à l'enseignement.

7.2.1 Disciplines enseignées

Recherche Opérationnelle.

- Période : depuis sept. 2002, - Public : 2ème année FICM.

Chargé de travaux dirigés pour un groupe d'une quinzaine d'élèves, ce cours est dispensé en tronc commun de deuxième année de la filière FICM en 15 séances de 3 heures (Cours et travaux dirigés). On y enseigne les concepts et quelques algorithmes sur des graphes, la modélisation et la programmation linéaire, la programmation dynamique, les chaînes de markov et les files d'attente. Dans ce cours j'assure une séance de cours maristral sur la complexité des algorithmes et des problèmes.

Optimisation discrète et déterministe.

- Période : depuis sept. 2002, - Public : 2ème année FICM.

Responsable, chargé de cours et de travaux dirigés. Ce cours est dispensé en deuxième année de l'option ISDP (Ingénierie des Systèmes de Décision et de Production) de la filière FICM en 15 séances de 3 heures (cours et de travaux dirigés). Ayant la responsabilité du cours, je m'occupe de la rédaction des sujets de TD et des sujets d'examen. Ce cours vient en complément du cours de recherche opérationnelle (dispensé en tronc commun), et propose des notions avancées d'optimisation combinatoire, à savoir, la programmation en nombre entiers, flots dans les réseaux et applications, complexité des problèmes, méthodes exactes de type séparation et évaluation, méthodes approchées avec guarantie de performances, métaheuristiques et algorithmes évolutionnistes, programmation par contraintes. Dans e cadre de cours, chaque année 4 projets à réaliser sont proposées aux élèves, à raison de 8 élèves par projets. Les projets portent sur des problèmes d'optimisation complexes proches de la réalités (par exemple le problème de distribution des produits pétroliers des raffineries aux stations, problème de production et de distribution des emballages métalliques, etc.). L'objectif est de réaliser un outils complet d'optimisation et d'aide à la décision (gestion de bases de données, interfaces graphiques et les différents modules d'optimisation). Dans chaque groupe, deux élèves font chefs de projet. En plus de l'objectif pédagogique qui est l'utilisation des méthodes d'optimisation enseignées dans ce cours, cette expérience fait réfléchir les élèves sur les difficultés liées à la gestion du temps et le partage des tâches dans un projet complexe et surtout composé de plusieurs élèves.

Bases de données.

- Période : année, 02-03, - Public : 2ème année FICM.

Chargé de travaux dirigés. Ce cours est dispensé en deuxième année de l'option ISDP de la filière FICM en 7 séances de 3 heures cours et de travaux pratiques. L'objectif de ce cours

est d'appendre aux élèves à construire et interroger une base de données en utilisant l'outil Acces, et une réalisation tout au long de ce cours d'un projet concret comme la gestion d'une agence touristique, gestion des prêts dans une médiathèque.

Analyse des données et Datamining.
- Période : depuis sept. 2005, - Public : 3ème année FICM.
Chargé de travaux dirigés. Ce cours est commun aux options ISDP et IM (Ingénierie mathématique) de la filière FICM et dispensé en troisième année en 15 séances de cours et de travaux dirigés. Ce cours regroupe les différentes techniques d'analyse de données (analyse en composantes principales, analyse des correspondances, analyse discriminante, ...) et des techniques de datamining (classification automatique, discrimination et classification neuronales, segmentation, etc.). En travaux dirigés, je m'occupe aussi de l'assistance à l'utilisation du logiciel SAS et SPAD pour la réalisation des TDs.

Optimisation et gestion de production.
- Période : depuis sept. 2004, - Public : 2ème année FI-MGP.
Responsable, chargé de cours et de travaux dirigés. Ce cours est dispensé en deuxième année de la filière FI-MGP (Formation d'Ingénieur - Matériaux et Gestion de Production) en 7 séances de 3 heures cours et de travaux dirigés. Ce cours reprend les outils de modélisation et d'optimisation pour la gestion de production, par exemple la modélisation en programmation linéaire de la gestion des stocks et de la commande, l'ordonnancement de projets, affectation et de transport.

Algorithms and data bases.
- Période : année, 03-05, - Public : Master of science in Industrial Engineering and International Management.
Chargé de travaux dirigés. Ce cours est dispensé en anglais aux élèves de Master of science in Industrial Engineering and International Management, 14 séances de 3 heures cours et de travaux pratiques. L'objectif de ce cours est d'appendre aux élèves à programmer les différents algorithmes d'optimisation et de construire et interroger une base de données en utilisant les outils Visual basic et Acces.

Discret optimisation.
- Période : année, 03-05, - Public : Master of science in Industrial Engineering and International Management.
Responsable, chargé de cours et de travaux dirigés. Ce cours est dispensé en anglais aux élèves de Master of science in Industrial Engineering and International Management, 14 séances de 3 heures cours et de travaux pratiques. Ce cours reprend une partie du cours de recherche opérationnelle, comme les concepts et quelques algorithmes sur des graphes, la modélisation et la programmation linéaire, la programmation dynamique et une partie du cours optimisation discrète et déterministe, à savoir, les flots dans les réseaux et applications, la complexité des problèmes, les méthodes exactes de type séparation et évaluation et les métaheuristiques.

7.2.2 Activités et responsabilités liées à l'enseignement

Durant la période 03-08, j'ai pris en charge des responsabilités liées à l'enseignement. Ces responsabilités sont décrites dans les points suivants :

– Responsable pédagogique du Mastère Génie Industriel - Aide à la décision pour les systèmes de production et de distribution (ancien mastère de Recherche Opérationnelle et Stratégie de Décision) entre 03 et 05. Responsable du cours 'Optimisation discrète et déterministe' et du cours 'Programmation et bases de donnés'. Je m'occupais aussi du suivi des élèves pour la recherche de stages.

– Encadrement d'élèves ingénieurs (en moyenne 4 élèves chaque année) des filières FICM et FI-MGP dans le cadre de stages de fin détude. Je m'occupe du suivi du stage par email et par une visite sur le lieu de stage.

– Participation aux jurys de soutenance des stages de fin d'études (en moyenne 9 soutenances par année).

– Organisation de la "semaine d'option" de la FICM, une semaine où les élèves suivent des cours essentiellement assurés par des intervenants extérieurs et des industriels. Je m'occupe de la recherche des intervenants industriels et d'organiser leurs interventions (suivi administratif et pédagogique).

7.3 Encadrements de travaux de recherche

L'encadrement des étudiants est une activité importante, elle permet de transmettre un savoir en dehors du cadre pédagogique des cours, par exemple par l'initiation à la recherche au niveau master ou par l'approfondissement et la recherche de nouveaux résultats au niveau de doctorat. Depuis mon recrutement à l'école des mines, j'ai eu l'occasion d' encadrer plusieurs étudiants en doctorat et en master.

7.3.1 Thèses de doctorat

2002-2005 Julien FONDREVELLE, *'Méthodes exactes et approchées pour le problème de flowshop avec des contraintes d'écarts minimaux et maximaux'*, thèse BDI cofinancée par le CNRS et la région Lorraine, soutenue en novembre 2005, encadrement à 50% (encadrement 50% par Marie-Claude Portmann, directeur de thèse). Julien Fondrevelle a obtenu le prix de thèse INPL, et actuellement il est Maître de Conférences à l'INSA de Lyon.

Résumé. Nous nous sommes intéressés à l'étude et la résolution de problèmes d'ordonnancement de type flowshop de permutation, en présence de contraintes d'écarts temporels (ou time lags), définies entre les couples d'opérations consécutives au sein des tâches. De telles contraintes généralisent les contraintes de précédence classiques et peuvent modéliser de nombreuses situations réelles. De nouveaux résultats de complexité sont démontrés et viennent compléter des résultats classiques tirés de la littérature. Nous avons présenté un état de l'art sur les travaux concernant les problèmes d'ordonnancement avec time lags, qui met en évidence le manque d'attention reçu par ces problèmes. Nous avons développé ensuite un schéma générique de résolution exacte reposant sur une Procédure par séparation et Evaluation et nous l'avons utilisé pour résoudre plusieurs problèmes de flowshop de permutation en présence de time lags. L'efficacité de cette approche de résolution est évaluée grâce à des séries d'expériences numériques. Enfin, des extensions permettant de prendre en compte des contraintes

supplémentaires sont proposées.

2004-2007 Zerouk MOULOUA, *'Ordonnancements coopératifs pour les chaînes logistiques'*, thèse MNERT soutenue en novembre 2007, encadrement à 30% (encadrement 70% par Marie-Claude Portmann, directeur de thèse). Actuellement Zerouk Mouloua travaille en tant que consultant en chaîne logistique au cabinet de conseil MOBIUS à Paris.

Résumé. Nous avons développé de nouveaux outils et méthodes d'aide à la décision pour l'ordonnancement de chaînes logistiques. Nous avons proposé des méthodes qui privilégient la coopération entre les différents acteurs de la chaîne logistique notamment en ce qui concerne la négociation avec les fournisseurs sur les dates d'arrivée des composants, et avec les clients sur les dates de livraisons des produits finis. Nous avons considéré une chaîne logistique qui consiste en un réseau d'entreprises avec des centres de décisions indépendants. Les produits finis ou semi finis des entreprises d'assemblage sont fabriqués en utilisant des composants ou produits semi finis par les autres entreprises du réseau ou par les fournisseurs externes. Au niveau opérationnel, chaque entreprise construit son ordonnancement par rapport à ses propres centres de production. Comme la production de produits finis dépend des composants, des négociations sont entamées entre les entreprises concernant les dates d'arrivées des composants (les fenêtres de temps). Une solution globale est obtenue par une approche itérative par décomposition incluant des négociations bilatérales entre les centres de production et de décisions pour définir l'ordonnancement juste à temps minimisant la somme des pénalités (retards et avances par rapport aux dates fixées). Pour résoudre l'ordonnancement juste à temps local à chaque centre de production nous avons proposé une méthode approchée basée sur les algorithmes génétiques. Chaque solution est évaluée grâce à un algorithme pseudo-polynomial basé sur le *PERT coût.* Un contrôle semi décentralisé est envisagé pour assurer la convergence des négociations. Par ailleurs, nous avons étudié un ensemble de problèmes concernant l'optimisation des transports dans les chaînes logistiques.

2006-2009 Adrien BELLANGER, *'Ordonnancement des flux de production dans l'industrie de pneumatique'*, thèse financée par une bourse Formation Recherche du Ministère de la Culture, de l'Enseignement Supérieur et de la Recherche du Grand-Duché de Luxembourg depuis Janvier 2007. Adrien Bellanger soutiendra la thèse en décembre 2009, encadrement à 70% (encadrement 30% par Marie-Claude Portmann, directeur de thèse)

Résumé. Dans cette thèse nous nous intéressons à l'étude et la résolution des problèmes d'ordonnancement de type flowshop et flowshop hybride dans un environnement de fabrication des pneumatiques. Ce sujet de thèse est défini en étroite collaboration avec l'industriel GoodYear au Luxembourg. Le processus de fabrication d'un pneu est composé de plusieurs étapes. Les plus importantes sont l'assemblage et la cuisson des pneus. La première étape consiste à assembler tous les éléments entrant dans la composition d'un pneu sur des machines de construction pour obtenir un pneu avant vulcanisation dit aussi pneu vert. La deuxième étape est la cuisson des pneus verts dans des presses de vulcanisation à des températures très élevées, et des durées de cuisson variant selon les gommes et les dimensions des pneus. L'atelier de production est structuré sous forme d'un flowshop hybride en deux niveaux, le premier niveau est composé de plusieurs machines d'assemblage et le second est composé de plusieurs machines de cuisson. La particularité de cet atelier se situe au deuxième niveau où les machines

de cuisson sont des batching machines, où chaque machine est capable de traiter deux pneus à la fois, à condition que ces deux pneus soient compatibles. Dans cette thèse la compatibilité entre les tâches est représenté sous forme d'un graphe de compatibilité. Nous cherchons à optimiser le problème d'ordonnancement de la production en minimisant les critères comme la durée totale de l'ordonnancement et le flot total. Nous avons obtenu plusieurs résultats théoriques ainsi que des méthodes de résolution exacte de type séparation et évaluation et des méthodes approchées de type heuristiques.

7.3.2 Masters et ex. DEA

2006. Encadrement à 50% d'Aymen Mili (Marie-Claude Portmann 50%) entre février et septembre 2006, du Master Conception industrialisation pour le développement durable, ENSAM de Metz, *'Ordonnancement conjoint de la production et de la maintenance'.* Ce travail a donné lieu à une publication dans le livre des articles de la conférence ROADEF 07. Actuellement Aymen Mili est en thèse CIFRE chez ST microelectronics, à Grenoble.

Ecadrement à 100% d'Adrien Bellanger entre mars et septembre, 2006 du Master Informatique, Paris 6, *'Flowshop hybride avec machines à traitement par batches et contraintes de compatibilités entre les tâches'.* Ce travail a donné lieu à une publication dans le livre des articles de la conférence ROADEF 07. Adrien Bellanger a obtenu une bourse Formation Recherche du Ministère de la Culture, de l'Enseignement Supérieur et de la Recherche du Grand-Duché de Luxembourg et il soutiendra sa thèse en décembre 09.

2005. Encadrement à 100% d'Aimé Kamgaing Kuiteing entre mars et septembre 2005, du Master Informatique, Paris 6, *'Flowshop avec machine à traitement par batches et compatibilité entre les tâches'.* Ce travail a donné lieu à une publication dans les actes de la conférence Incom'06 et un papier dans le journal Computers& Operations Research. Aimé Kamgaing Kuiteing est en thèse de doctorat à l'école Polytechnique de Montréal au Canada.

2004. Encadrement à 50% de Zerouk Mouloua (Wahiba Ramdane Cherif, 35%) entre mars et juin 2004, du DEA Informatique de l'université Henri Poincaré Nancy, *'Réalisation d'une architecture générique pour l'ordonnancement en ligne : application au problème de tournées de personnel'.* Ce travail, dont le sujet a été fourni par BT Exact, filière de British Telecom, a donné lieu à une publication dans le livre des résumés de la conférence ROADEF 06.

2003. Encadrement à 100% de Faouzi Amier entre mars et juin 2003, du DEA Recherche Opérationnelle Combinatoire et Optimisation, INP-Grenoble, *'Minimisation de la durée totale de l'ordonnancement d'un atelier de type flowshop en présence de deux max-batch machines'.*

Encadrement à 100% de Salem Maloum entre février et septembre 2003, du DEA Génie des systèmes industriels, de l'INPL de Nancy, *'Atelier flowshop avec machines à traitement par batches'.*

7.4 Diverses activités liées à la recherche et à l'administration

7.4.1 Participation à des projets de recherche

– Participation au projet européen V-CHAIN, piloté par DMR Consulting, un contrat UE Growth (2001-2004). Le projet concerne la définition et la gestion de chaînes logistiques liées à la production manufacturière dans le cadre de l'entreprise virtuelle, c'est à dire d'un réseau de fournisseurs et de donneurs d'ordre partageant les risques et les profits de la production. Les domaines d'application concernent l'industrie automobile (Ford, Espagne) et la fabrication de motocycles (Aprilia, Italie). En collaboration avec l'université d'Udine (Italie), j'ai réalisé des développements pour la planification et l'ordonnancement de l'assemblage de motos pour le pilote Italien. Les partenaires du projet étaient Ford (Espagne), Aprilia (Italie), DMR Consulting (Espagne), Université de Valencia, Université de Udine.

– Porteur d'une soumission pour un projet PHC - PROCORE, 2005. titre *'Ordonnancement par fournées dans les chaînes logistiques'* avec Chi To Ng du Département de Logistique de l'université Polytechnique de Hong Kong. (projet non obtenu).

– Porteur d'une soumission pour un projet PHC- PICASSO - PROCOPE, 2006. Titre *'Techniques d'optimisation dans les industries de pneumatique et de céramique'* avec Ruben Ruiz Garcia de Grupo de investigacion operativa - GIO, Université Polytechnique de Valencia, Espagne pour la partie PICASSO du projet et avec Thomas Stuetzle de FG Intellektik department d'informatique Darmstadt, Allemagne, pour la partie PROCOPE du projet. (projet non obtenu).

– Participation en 2005 à une soumission (élaboration des documents pour la partie Nancy) pour le 6th Framework Programme des projets Européens, titre *'A Flexible Production Scheduling System to increase competitiveness in the European Manufacturing Industry - FPSS-Manufacturing'* avec des partenaires en Espagne, Italie, l'Allemagne et Turquie. (projet non obtenu).

– Participation en 2006 au projet de recherche GDR RO, ORDO-SC, sur l'ordonnancement en chaîne logistique, les partenaires étaient : le Laboratoire d'Informatique de Tours et le laboratoire LIP6 de Paris.

– Participation en 2007 à une soumission (coordinateur pour l'INPL) pour un projet Européen FP7-ICT-2007-1, titre *'End-User experience agent-based monitoring and management for scalable future Networks*' avec des partenaires en Espagne, Turquie, UK, Allemagne, Israel. (projet non obtenu).

– Porteur d'un projet PICS (Projet International de Coopération Scientifique) entre le CNRS et La Fondation Biélorusse pour la Recherche Fondamentale, projet soumis en mai 2008, titre *'Optimal planning decisions in logistics and supply chain management'* le partenaire est le laboratory of mathematical cybernetics, Biélorusse, représenté par le professeur M.Y. Kovalyov.

7.4.2 Jury de thèses

- Novembre 05 : Membre du jury de thèse de Julien Fondrevelle (co-directeur de thèse), *'Méthodes exactes et approchées pour le problème de flowshop avec des contraintes d'écarts minimaux et maximaux'*, INPL, Nancy.
- Novembre 07 : Membre du jury de thèse de Zerouk Mouloua (co-directeur de thèse), *'Ordonnancements coopératifs pour les chaînes logistiques'*, INPL, Nancy.
- Octobre 08 : Membre du jury de thèse de Sylvain Fournier (examinateur), *'Outils pour des problèmes industriels de tournées de véhicules avec transbordement'*, INPG, Grenoble.

7.4.3 Séjours et invitations à l'étranger

- Séjour d'une semaine en juin 2005 à Minsk suite à la conférence ECCO XVIII, pour travailler avec les Pr. M. Kovalyov, et M. Shafransky, Biélorussie.
- Invité par le Pr. W. Kubiak pour un séjour de recherche d'un mois en Août 2005 à St johns, Newfoundland, Canada.
- Invité par le Pr. M. Boudhar trois fois deux semaines en avril 2006, 2007 et 2008 à la faculté de mathématiques, Université USTHB, Alger, Algérie, (à chaque séjour, je donne un cours sur l'optimisation dans les systèmes de production, je participe au jury de magister,)

7.4.4 Participation à des groupes de travail

- Membre du groupe de travail Gotha - groupe de recherche en ordonnancement théorique et appliquée.
- Membre du groupe de travail Bermudes - groupe de travail du GDR MACS et GDR RO.

7.4.5 Administration de la recherche

- Co-organisateur des journées du séminaire ordonnancement du LORIA, regroupant les équipes, ALGORILLE, TRIO, ORCHIDS.
- Editeur invité d'un numéro spécial du journal European journal of Operational Research - EJOR sur le *Project Management and Scheduling*.
- Co-organisation d'une journée conjointe Gotha-Bermudes à Nancy en 2004.
- Co-responsable du comité d'organisation de la Conférence internationale PMS 04, Nancy 26-28 avril 2004 (préparation du budget, recherche de sponsors, création du site web, collecte et répartition des soumissions, préparation des actes de la conférence).
- Président du comité d'organisation de ROADEF 09, Nancy 10-12 février 2009, (préparation du budget, recherche de sponsors, création du site web, collecte et répartition des soumissions, préparation des actes de la conférence).

7.4.6 Autres responsabilités

- Mai. 07 - Mars. 08 : Membre élu du conseil d'administration de l'INPL (environ une réunion par mois. Préparation des réunion, lecture de documents)
- Depuis Avril 08 : Membre élu du Conseil scientifique de l'INPL (environ une réunion par mois, Préparation des réunion, lecture de documents).

7.4.7 Arbitrages

– Journaux internationaux (en moyen 5 articles par année) : European Journal of Operational Research (EJOR), International Journal of Production Research (IJPR), Journal of the Operational Research Society (JORS), Journal of Scheduling (JOS), OMEGA, International Journal of Management Sciences , 4OR A Quarterly Journal of Operations Research, Computers & Operations Research (COR), IIE Transactions. International Journal of Production Economics (IJPE).

7.5 Publications

7.5.1 Édition

– A. Oulamara, M.-C. Portmann. Guest Editors of issue on Project Management and Scheduling. European journal of Operational Research, Vol. 189, Issue 3, september 2008.

– A. Oulamara, M.-C. Portmann : Editeur du proceeding de la conférence : Ninth international workshop on project management and scheduling, April 26 - 28, 2004, Nancy.

7.5.2 Chapitre de livre

– M.-C. Portmann, A. Oulamara. Optimisation discrète. Chapitre pour Techniques de l'Ingénieur, 2006.

7.5.3 Revues Internationales

– A. Bellanger and A. Oulamara. *Scheduling hybrid flowshop with parallel batching machines and compatibilities.* Computers & Operations Research, 36, 1982–1992, 2009.

– J. Fondrevelle, A. Oulamara, M.-C. Portmann, A. Allahverdi. *Permutation flowshops with exact time lags to minimize maximum lateness.* International Journal of Production Research, article in Press, 2008.

– J. Fondrevelle, A. Oulamara, M-C. Portmann. *Permutation flowshop scheduling problems with time lags to minimize the weighted sum of machine completion times.* International Journal of Production Economics. Volume 112, Issue 1, Pages 168-176, 2008.

– A. Oulamara, G. Finke, A. Kamgaing Kuiteing. *Flowshop Scheduling Problem with a Batching Machine and Task Compatibilities.* Computers and Operations Research, available online October. 2007.

– G. Finke and A. Oulamara. *Total completion time in a two-machine flowshop with deteriorating tasks.* Journal of Mathematical Modeling and Algorithms, Volume 6, Number 4, pp. 563-576, december 2007.

– A. Oulamara, *Makespan minimization in no-wait flow shop problem with batching machines*. Computers and Operations Research, 34, 1033-1050, 2007.

– J. Fondrevelle, A. Oulamara and M-C. Portmann. *Permutation flowshop scheduling problems with maximal and minimal time lags*. Computers and Operations Research. 33(6), p. 1540-1556, 2006.

– M.Y. Kovalyov, M.-C. Portmann, A. Oulamara. *Optimal testing and repairing a failed series system*. Journal of Combinatorial Optimization, 12 : 279-295, 2006.

– J. Fondrevelle, Allahverdi and A. Oulamara. *Two-machine no-wait flowshop scheduling problem to minimize maximum lateness with separate setup and removal times*. International Journal of Agile Manufacturing, Vol. 8, pp. 165-174, 2005.

– A. Soukhal, A. Oulamara, P. Martineau. *Flowshop scheduling problem with transportation constraints*. European Journal of Operations Research, 161 (1), 32-41, 2005.

– A. Oulamara, M.Y. Kovalyov, G. Finke. *Scheduling a no-wait flowshop with unbounded batching machines*. IIE Transactions on Scheduling and logistics, 37(8), 685-696, 2005.

– A. Oulamara, G.Finke. *Flowshop Problems with two batch processing machines*. International Journal of Mathematical Algorithms, Vol. 2, pp.269-287, 2001.

7.5.4 Conférences avec actes

– A. Bellanger, A. Oulamara. Exact method for hybrid flowshop with batching machines and tasks compatibilities. 11th International Workshop on Project Management and Scheduling - PMS2008, Istanbul, Turkey, April 2008.

– A. Mili, A. Oulamara. M-C. Portmann. Ordonnancement conjoint de la production et de la maintenance. Livre articles. FRANCORO V/ ROADEF 2007. Pp.207-215. ISBN. 978-2-0.

– A. Bellanger, A. Oulamara. Flowshop hybride avec machines à traitement par batches et compatibilité entre les tâches. Livre articles. FRANCORO V/ ROADEF 2007. Pp. 17-35. ISBN. 978-2-7061-1397-0.

– Z. Mouloua and A. Oulamara. A dynamic programming approach for minimizing the transportation costs in a supply chain, The 4th International federation of Automatic Control Conference on Management and Control of Production and Logistics, 753-758, Romania, 2007.

– A. Kamgaing Kuiteing, A. Oulamara, G. Finke. Flowshop scheduling problem with batching machines and task compatibilities. 12th IFAC/IFIP/IFORS/IEEE/IMS Symposium on Information Control Problems in Manufacturing INCOM 06, Saint Etienne, France, June, 2006.

– Z. Mouloua and A. Oulamara. Cooperation in supply chain scheduling : minimizing the in-

ventory holding cost. International conference on Information systems, Logistics and Supply chain. (1) 564-573, Lyon, 2006.

– J. Fondrevelle, A. Oulamara. and M.-C. Portmann. Minimizing the weighted sum of machine completion times in flowshop with time lags, IESM 2005, International Conference on Industrial Engineering and Systems Management, Marrakech, du 16 au 19 mai 2005.

– J. Fondrevelle, A. Oulamara and M.-C. Portmann. Scheduling unit time operation permutation flowshop with minimal time lag, PMS 04, Ninth International Workshop on Project Management and Scheduling, Nancy, p.178-181, avril 2004.

– A. Oulamara. No-wait flowshop problem with two mixed batching machines. Conference of Management and Control of Production and Logistics, MCPL2004, 3-5 Novembre 2004.

– A. Oulamara, G. Finke. No-wait flowshop problem with two batch processing machines. In Proceedings Second Conference IFAC/IFIP/IEEE on Management and Control of Production and Logistics (MCPL'2000), Grenoble, France, (2000), vol. 2, 663-668.

7.5.5 Conférences sans actes

– G. Finke , M. Y. Kovalyov , A. Oulamara. Batch scheduling in a no-wait flowshop, in : ECCO XVIII, Minsk, Belarus, p. 17. May 2005.

– A. Oulamara. Makespan minimization in no-wait flowshop with two batching machines, in : ECCO XVIII, Minsk, Belarus, p. 49. May 2005.

– J. Fondrevelle , A. Oulamara , M.-C. Portmann. Approche de résolution pour les problèmes de flowshop avec time-lags minimaux et maximaux, in : 6ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision ROADEF'05, Tours, Février 2005.

– G. Thisselin, A. Oulamara , W. Ramdane-Cherif , M.-C. Portmann. Méthodes d optimisation combinatoire pour un probleme d optimisation de maintenance corrective, in : ROADEF 2005, Février 2005.

– J. Fondrevelle , A. Oulamara , M.-C. Portmann. Minimizing makespan in flowshop with time lags, in : Models and Algorithms for Planning and Scheduling Problems - MAPSP'2005, Siena, Italy, p. 138-139, June 2005.

– J. Fondrevelle , A. Oulamara , M.-C. Portmann. Minimizing the weighted sum of machine completion times in flowshop with time lags : complexity results and a solution approach, in : International Conference on Industrial Engineering and Systems Management - IESM'05, Marrakech, Maroc, Actes sur CD-Rom, May 2005.

– Z. Mouloua , A. Oulamara , W. Ramdane-Cherif. Approche par décomposition pour le problème de planification de tournées de personnel, in : 6ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision - ROADEF'2005, Tours, France, February

2005.

– J. Fondrevelle, A. Oulamara, M.-C. Portmann. Minimizing the weighted sum of machine completion times in flowshop with time lags. International Conference on Industriel Engineering and Systems Management, 2005.

– A. Oulamara, A. Soukhal, P. Martineau. Some results for particular cases of two machines flowshop with transportation constraints, International Conference on Industrial Engineering and Production Management, Quebec city, Canada, August 2001.

– A. Oulamara, A. Soukhal. Flowshop scheduling problems with transportation and capacities constraints, International Conference on Systems, Man, and Cybernetics, Arizona, US, October 2001.

7.5.6 Rapport de Recherche

– J. Fondrevelle, A. Allahverdi A. Oulamara, M.-C. Portmann. Permutation flowshop with exact time lags to minimize maximum lateness. Rapport de Recherche du Loria, 2005

– J. Fondrevelle, A. Allahverdi, A. Oulamara. Two-machine no-wait flowshop scheduling problem to minimize maximum lateness with separate setup and removal times, Rapport de Recherche du Loria, 2004.

8

Sélection de publications

8.1 Computers and Operations Research (2006)

[**1**] J. Fondrevelle, A. Oulamara, M-C.Portmann. *Permutation flowshop scheduling problems with maximal and minimal time lags.* Computers and Operations Research 33, 1540 - 1556, 2006.

Abstract. In this paper, we study permutation flowshop problems with minimal and/or maximal time lags, where the time lags are defined between couples of successive operations of jobs. Such constraints may be used to model various industrial situations, for instance the production of perishable products. We present theoretical results concerning two-machine cases, we prove that the two-machine permutation flowshop with constant maximal time lags is strongly NP-hard. We develop an optimal branch and bound procedure to solve the m-machine permutation flowshop problem with minimal and maximal time lags. We test several lower bounds and heuristics providing upper bounds on different classes of benchmarks, and we carry out a performance analysis.

8.2 International Journal of Production Economics (2008)

[**2**] J. Fondrevelle, A. Oulamara, M.-C. Portmann. *Permutation flowshop scheduling problems with time lags to minimize the weighted sum of machine completion times.* International Journal of Production Economics, 112, 168–176, 2008.

Abstract. In this article, we consider flowshop scheduling problems with minimal and maximal time lag constraints. Such constraints extend precedence constraints between operations in the jobs and may be used to model various industrial applications. The objective is to minimize a non-classical criterion based on the weighted sum of machine completion times. We show that it generalizes makespan and we derive several complexity results for two- and three-machine problems. An exact algorithm based on a branch-and-bound procedure is developed to solve the permutation flowshop problem with m machines.

8.3 Computers and Operations Research (2008)

[**3**] A. Bellanger and A. Oulamara. *Scheduling hybrid flowshop with parallel batching machines and compatibilities.* Computers and Operations Research, Article in press, doi :10.1016/j.cor.2008.06.011. 2008.

Abstract. This paper considers a two-stage hybrid flowshop problem in which the first stage contains several identical discrete machines, and the second stage contains several identical batching machines. Each discrete machine can process no more than one task at time, and each batching machine can process several tasks simultaneously in a batch with the additional feature that the tasks of the same batch have to be compatible. A compatibility relation is defined between each pair of tasks, so that an undirected compatibility graph is obtained which turns out to be an interval graph. The batch processing time is equal to the maximal processing time of the tasks in this batch, and all tasks of the same batch start and finish together. The goal is to make batching and sequencing decisions in order to minimize the makespan. Since the problem is NP-hard, we develop several heuristics along with their worst cases analysis. We also consider the case in which tasks have the same processing time on the first stage, for which a polynomial time approximation scheme (PTAS) algorithm is presented.

8.4 IIE Transactions (2005)

[**4**] A. Oulamara, M.Y. Kovalyov, G.Finke. *Scheduling a no-wait flow shop containing unbounded batching machines.* IIE Transactions, Volume 37, Issue 8, pages 685 - 696, August 2005.

Abstract. We study the problem of scheduling n jobs in a no-wait flow shop consisting of m batching machines. Each job has to be processed by all the machines. All jobs visit the machines in the same order. A job completed on an upstream machine should be immediately transferred to the downstream machine. Batching machines can process several jobs simultaneously in a batch so that all jobs of the same batch start and complete together. The processing time of a batch is equal to the maximum processing time of the jobs in this batch. We assume that the capacity of any batch is unbounded. The problem is to find an optimal batch schedule such that the maximum job completion time, that is the makespan, is minimized. For $m = 2$, we prove that there exists an optimal schedule with at most two batches and construct such a schedule in $O(nlogn)$ time. For $m = 3$, we prove that the number of batches can be limited to nine and give an example where all optimal schedules have seven batches. Furthermore, we prove that the best schedules with at most one, two and three batches are $3-$, $2-$ and $3/2-$ approximate solutions, respectively. The first two bounds are tight for corresponding schedules. Finally, we suggest an assignment method that solves the problem with m machines and at most r batches in $O(n^{m(r-2)+1+[m/r]})$ time, if m and r are fixed. The method can be generalized to minimize an arbitrary maximum cost or total cost objective function.

8.5 Journal of Combinatorial Optimization (2006)

[5] M.Y. Kovalyov, M-C. Portmann and A. Oulamara. *Optimal testing and repairing a failed series system*. Journal of Combinatorial Optimization, Volume 12, Number 3, 279-295, novembre 2006.

Abstract. We consider a series repairable system that includes n components and assume that it has just failed because exactly one of its components has failed. The failed component is unknown. Probability of each component to be responsible for the failure is given. Each component can be tested and repaired at given costs. Both testing and repairing operations are assumed to be perfect, that is, the result of testing a component is a true information that this component is failed or active (not failed), and the result of repairing is that the component becomes active. The problem is to find a sequence of testing and repairing operations over the components such that the system is always repaired and the total expected cost of testing and repairing the components is minimized. We show that this problem is equivalent to minimizing a quadratic pseudo-boolean function. Polynomially solvable special cases of the latter problem are identified and a fully polynomial time approximation scheme (FPTAS) is derived for the general case. Computer experiments are provided to demonstrate high efficiency of the proposed FPTAS. In particular, it is able to find a solution with relative error $\epsilon = 0.1$ for problems with more than 4000 components within 5 minutes on a standard PC with 1.2 Mhz processor.