UNIVERSITÉ DE NANTES

École Doctorale
« Sciences et Technologies de l'Information et Mathematiques »

Année : 2008

Thèse de Doctorat de l'Université de Nantes

Spécialité : Génie Informatique, Automatique et Traitement du Signal

Présentée et soutenue publiquement par

Guillaume Pinot

le 14 novembre 2008 à l'IRCCyN, Nantes

Coopération homme-machine pour l'ordonnancement sous incertitudes

Jury :

Jean-Charles Billaut	Professeur, Université de Tours	Président
Christian Artigues	Chargé de Recherche, CNRS, Toulouse	Rapporteur
Jacques Carlier	Professeur, UTC, Compiègne	Rapporteur
Stéphane Dauzère-Pérès	Professeur, École des Mines de Saint Étienne	Examinateur
Pierre Castagna	Professeur, Université de Nantes	Directeur de thèse
Jean-Michel Hoc	Directeur de Recherche, CNRS, Nantes	Codirecteur
Nasser Mebarki	Maître de Conférences, Université de Nantes	Coencadrant

Composante de rattachement du directeur de thèse : IUT de Nantes N° ED 503 -016

À Aude.

Remerciements

Tout d'abord, je tiens à remercier mes encadrants sans lesquels je n'aurais pas pu faire cette thèse. Merci à Nasser Mebarki pour son suivi et son attention, à Jean-Michel Hoc pour ses précieux conseils en coopération homme-machine, et à Pierre Castagna pour sa supervision.

Mes remerciements vont aussi aux membres des équipes ACSED et PsyCoTec, et plus particulièrement à Jean-Jacques Loiseau, Julien Cegarra et Olivier Cardin pour leurs conseils et les travaux que nous avons effectués ensembles.

Je remercie Christian Artigues et Jacques Carlier pour avoir accepté d'être les rapporteurs de cette thèse de doctorat. Merci également à Jean-Charles Billaut et Stéphane Dauzère-Pérès, membre du jury.

Merci à Chantal Enguehard pour m'avoir mis en relation avec mes encadrants.

Un grand merci à ma famille et mes proches pour leur soutien et leurs relectures. Merci Andrine, Anne Marie, Antoine, Armelle, Aude, Lucile et Yves.

Finalement, je remercie mes amis ainsi que les doctorants de l'IRCCyN et du LINA, que je ne citerai pas pour ne pas faire de jaloux.

Table des matières

Liste des tableaux

Table des figures

Glossaire

C_i	Date de fin de l'opération O_i.
$C_{\max}$	*Makespan*, ou temps total de l'ordonnancement.
$\underline{C}_i^{\Pi}$	Date de fin au plus tôt de l'opération O_i dans le pire des cas pour l'ordonnancement de groupes Π.
$\bar{C}_i^{\Pi}$	Date de fin au plus tard de l'opération O_i dans le pire des cas pour l'ordonnancement de groupes Π.
d_i	Date de fin souhaitée de l'opération O_i.
$\tilde{d}_i$	Date de fin impérative de l'opération O_i.
$g_{\ell,k}$	Groupe d'opérations permutables sur la machine M_ℓ en k^{e} position.
$g(i)$	Groupe d'opérations permutables contenant l'opération O_i.
$g^+(i)$	Groupe d'opérations permutables situé après le groupe $g(i)$.
$g^-(i)$	Groupe d'opérations permutables situé avant le groupe $g(i)$.
$\#\text{Gps}(\Pi)$	Nombre de groupes dans l'ordonnancement de groupes Π.
$\Gamma^-(i)$	L'ensemble des prédécesseurs de l'opération O_i.
$\Gamma^+(i)$	L'ensemble des successeurs de l'opération O_i.
i	Indice des opérations.
k	Indice de séquence d'un groupe d'opération permutable sur une machine (voir $g_{\ell,k}$).
L_i	Retard algébrique de l'opération O_i, calculé $C_i - d_i$.
$L_{\max}$	Retard algebrique maximum de l'ordonnancement.
ℓ	Indice des machines.
M_ℓ	Machine ℓ.
m_i	Machine utilisée par l'opération O_i.
m	Nombre de machines.
$\text{m}_{\text{s}}^{\Pi}(i)$	Marge libre séquentielle de l'opération O_i pour l'ordonnancement de groupes Π.
$\text{m}_{\text{sp}}^{\Pi}(i)$	Marge propre de l'opération O_i pour l'ordonnancement de groupes Π.

$\mathrm{m}_{\mathrm{sg}}^{\Pi}(i)$	Marge groupe de l'opération O_i pour l'ordonnancement de groupes Π.
n	Nombre d'opérations.
O_i	Opération i.
p_i	Temps d'exécution de l'opération O_i.
ϕ_{Π}	Flexibilité sous forme de taux, relatif au nombre de groupes, pour l'ordonnancement de groupes Π.
r_i	Date de disponibilité de l'opération O_i.
$\#\mathrm{Seq}(\Pi)$	Nombre d'ordonnancements semi-actifs décrits par l'ordonnancement de groupes Π.
t_i	Date de début de l'opération O_i.
T_i	Retard de l'opération O_i, calculé $\max(0, C_i - d_i)$.
$T_{\max}$	Retard maximum de l'ordonnancement.
$\sum T_i$	Somme des retards.
$\sum w_i T_i$	Somme pondérée des retards.
$\underline{\tau}_i^{\Pi}$	Date de début au plus tôt de l'opération O_i dans le pire des cas pour l'ordonnancement de groupe Π.
$\bar{\tau}_i^{\Pi}$	Date de début au plus tard de l'opération O_i dans le pire des cas pour l'ordonnancement de groupe Π.
U_i	Indique si l'opération O_i est en retard : $U_i = 0$ si O_i n'est pas en retard, $U_i = 1$ sinon.
$\sum U_i$	Nombre d'opérations en retard dans l'ordonnancement.
v_ℓ	Nombre de groupes d'opérations permutables sur la machine M_ℓ.

Introduction

Depuis les années cinquante, l'ordonnancement est un domaine actif de recherche. La plupart des travaux de ce domaine repose sur un modèle déterministe, peu adapté à la réalité de l'ordonnancement d'atelier. En effet, les ateliers de production sont soumis à un certain nombre d'incertitudes, qui dégradent les performances prédites par les méthodes déterministes. C'est pourquoi l'ordonnancement sous incertitudes est un domaine en pleine expansion.

D'autre part, l'humain n'est généralement pas pris en compte dans l'élaboration de la méthode d'ordonnancement. Pourtant, il joue un rôle central dans le processus d'ordonnancement, et ses connaissances du terrain sont précieuses. En effet, il peut modifier certaines contraintes du problème pour le rendre plus facile à résoudre, par exemple en négociant les dates de livraison des produits. De plus, il est généralement nécessaire dans l'atelier pour des tâches annexes au processus d'ordonnancement comme la qualité ou la sécurité. Impliquer l'humain dans le processus est alors important pour qu'il ait une bonne connaissance de l'état de l'atelier. C'est pourquoi nous pensons que des systèmes homme-machine efficaces favorisant l'activité de l'humain sont nécessaires au bon fonctionnement des méthodes d'ordonnancement d'atelier.

Notre but est donc de proposer des systèmes homme-machine permettant de réaliser l'ordonnancement dans un atelier de production. Un tel système devra combiner les avantages de l'humain et de la machine pour réaliser un ordonnancement efficace, et ce même en présence d'incertitudes.

Un certain nombre de méthodes d'ordonnancement sous incertitudes est décrite dans la littérature. Parmi celles-ci, une méthode se dégage, l'ordonnancement de groupes. En effet, c'est la seule méthode proactive-réactive à être utilisée dans un cadre industriel. Nous avons donc décidé de baser nos travaux sur cette méthode. Notre analyse de cette méthode dans le cadre d'un système homme-machine nous a permis de constater qu'elle ne favorisait pas l'activité de l'humain. Nous nous attachons donc à améliorer la coopération entre l'humain et la machine pour utiliser pleinement les avantages de l'ordonnancement de groupes et pour améliorer l'activité de l'humain. Notre nouveau système homme-machine utilise l'évaluation de la qualité d'un ordonnancement de groupes dans le meilleur des cas. Nous proposons donc des bornes inférieures, des heuristiques et une méthode exacte adaptées à ce problème. Cette évaluation du meilleur des cas pour un ordonnancement de groupes est une contribution à cette méthode d'ordonnancement.

Ce mémoire se compose de trois parties. La première expose le choix de la méthode d'ordonnancement. La deuxième partie se concentre sur l'aspect coopération homme-machine de l'ordonnancement de groupes. La dernière partie étudie le meilleur des cas dans un ordonnancement de groupes.

La première partie, dont le but est de choisir une méthode d'ordonnancement sous incertitudes, est composée de trois chapitres. Tout d'abord, l'ordonnancement est présenté. Les différentes notations et notions utiles à la compréhension du reste du mémoire y sont décrites. Comme l'ordonnancement déterministe ne permet pas de s'adapter aux incertitudes de l'atelier, le domaine de l'ordonnancement sous incertitudes est présenté. Finalement, la méthode d'ordonnancement sous incertitudes que nous avons retenue, l'ordonnancement de groupes, est exposée. Les différents outils existants dans la littérature pour cette méthode d'ordonnancement sous incertitudes y sont présentés.

Dans le but d'améliorer les systèmes homme-machine pour l'ordonnancement sous incertitudes, la deuxième partie analyse la coopération homme-machine de l'ordonnancement de groupes en trois chapitres. Dans le premier, nous présentons les différents travaux sur les systèmes homme-machine que nous pensons utiles à notre problématique. Le deuxième chapitre analyse les différents systèmes homme-machine de la phase prédictive de l'ordonnancement de groupes. Nous proposons de modifier ce système homme-machine afin de faciliter la coopération. Le dernier chapitre de cette partie traite du système homme-machine pour la phase réactive. Après l'analyse de l'existant, nous proposons un nouveau système favorisant l'activité de l'humain. Le plan d'expérimentations permettant l'évaluation de ce nouveau système est décrit.

La qualité dans le meilleur des cas étant utilisée par ce dernier système, elle est étudiée dans la dernière partie. Le premier chapitre propose des bornes inférieures pour le meilleur des cas, ainsi qu'une relaxation en *one machine problem* de l'ordonnancement de groupes. Le deuxième chapitre présente différentes heuristiques : des règles de priorités ainsi qu'une adaptation du *shifting bottleneck* sont décrites et évaluées par des expérimentations. Le troisième chapitre propose une méthode exacte pour résoudre le meilleur des cas. Cette méthode est utilisable pour tout objectif régulier. Finalement, le dernier chapitre étudie l'utilisation de notre étude du meilleur des cas, aussi bien durant la phase prédictive que durant la phase réactive de l'ordonnancement.

Les différentes contributions présentées dans ce mémoire ont donné lieu à des publications en conférences. [PCM07] étudie l'impact de la non-modélisation des temps de transport, présenté à la section 3.4. [PMH08] propose notre nouveau système homme-machine pour la phase réactive décrit au chapitre 6. [PM08a, PM08b] présentent les bornes inférieures et les heuristiques des chapitres 7 et 8. Nous avons également soumis en mai 2008 l'article en revue [PMre] qui présente la méthode exacte du chapitre 9.

Première partie

Choix de la méthode d'ordonnancement

Chapitre 1

L'ordonnancement d'atelier

Ce chapitre présente l'ordonnancement d'atelier. Il regroupe les notions et notations nécessaires à la compréhension de ce mémoire. Il commence par définir l'ordonnancement ainsi que ses différents éléments. Ensuite, la classification des problèmes d'ordonnancement est décrite. Pour finir, une analyse critique de la discipline est réalisée.

1.1 Définitions générales

Cette section introduit les notations utilisées tout au long de ce mémoire ainsi que les définitions principales. Les notations sont celles utilisées dans [ABE05] car le même modèle est utilisé pour notre travail. Les notations non décrites dans cet article ont été choisies pour être les plus proches possible des notations de l'article et des notations classiques de la littérature.

1.1.1 L'ordonnancement

Un problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu de contraintes temporelles et de contraintes portant sur la disponibilité des ressources requises.

Un ordonnancement constitue une solution à ce problème. Il vise à satisfaire un ou plusieurs objectifs. Un ordonnancement est une compilation des dates de début et de fin des différentes opérations couplées à leurs ressources.

On utilise généralement la formule « l'ordonnancement » pour décrire le domaine de l'ordonnancement [1].

En général, la solution d'un problème d'ordonnancement est donnée sous forme d'une séquence d'opérations sur chaque machine. Une telle solution est appelée « séquence », et est une solution du problème de « séquencement ».

1. La distinction entre « un ordonnancement » et « l'ordonnancement » est plus visible en anglais : on parle respectivement de *"schedule"* et *"scheduling."*

L'ordonnancement est un problème très complexe. Les problèmes sont le plus souvent NP-difficiles, comme par exemple, le *flow shop* à 3 machines avec comme critère le $C_{\max}$ (voir [BK07] pour plus d'informations sur les problèmes NP-complet en ordonnancement). Pour résoudre ces problèmes, l'optimisation combinatoire est l'outil le plus commun. Des méthodes d'optimisation exactes (et exponentielles) et des méthodes d'optimisation heuristiques permettent de résoudre ces problèmes.

1.1.2 Les tâches

Une tâche est une entité élémentaire localisée dans le temps par une date de début et de fin, dont la réalisation nécessite un certain temps, et qui consomme certaines ressources. En ordonnancement d'atelier, on parle aussi d'opération.

On notera O_i une opération i. L'indice i sera utilisé pour les opérations exclusivement. Le temps d'exécution de l'opération O_i est noté p_i. La date de début d'une tâche O_i est notée t_i et sa date de fin est notée C_i. Le nombre d'opérations dans un problème donné sera noté n.

Dans la littérature, n est généralement le nombre de travaux ou *job*. Comme notre modèle utilise des contraintes de pécédence généralisées, la notion de *job* est absente du modèle. La constante n peut alors désigner le nombre d'opérations. En utilisant cette notation, n reflète toujours la taille du problème du point de vue des opérations.

1.1.3 Les ressources

La ressource est un moyen technique ou humain destiné à être utilisé pour la réalisation d'une tâche. Les tâches sont en concurrence pour obtenir une ou plusieurs ressources. Ces ressources ne sont disponibles qu'en quantité limitée. Plusieurs types de ressources sont à distinguer.

Une ressource est renouvelable si après avoir été allouée à une ou plusieurs tâches, elle est à nouveau disponible en même quantité (les hommes, les machines, l'équipement en général) ; la quantité de ressources utilisables à chaque instant est limitée.

Dans le cas contraire, elle est consommable (matières premières, budget) : la consommation globale (ou cumul) au cours du temps est limitée.

Une ressource est doublement contrainte lorsque son utilisation instantanée et sa consommation globale sont toutes deux limitées. Par exemple, l'énergie peut être modélisée ainsi : la consommation instantanée est limitée à cause de contraintes techniques et la consommation globale également à cause de son prix.

On distingue par ailleurs les ressources disjonctives (ou non partageables) qui ne peuvent exécuter qu'une tâche à la fois et les ressources cumulatives (ou partageables) qui peuvent être utilisées par plusieurs tâches simultanément.

En ordonnancement d'atelier, le terme machine est généralement utilisé à la place du terme ressource.

Dans le cadre de cette étude, nous ne prendrons en compte que le cas de ressources disjonctives renouvelables. Une machine ℓ sera notée M_ℓ. L'indice ℓ sera utilisé exclusivement pour les machines. Le nombre de machines dans un problème donné sera noté m.

1.1.4 Les contraintes

Les contraintes expriment des restrictions sur les valeurs que peuvent prendre simultanément les variables de décision. Il en existe deux types : les contraintes temporelles et les contraintes de ressources.

1.1.4.1 Les contraintes temporelles

Les contraintes temporelles se répartissent en deux ensembles : les contraintes de temps alloué et les contraintes de cohérence technologique.

Les contraintes de temps alloué sont généralement issues d'impératifs de gestion. Elles sont relatives aux dates limites des tâches (délais de livraisons, disponibilité des approvisionnements). Ainsi, une opération peut posséder une date de début au plus tôt (ou date de disponibilité), noté r_i, avant laquelle l'opération ne peut commencer. Une opération peut également posséder une date de fin impérative (ou date due), noté $\tilde{d}_i$, à laquelle l'opération doit être terminée. Un ordonnancement réalisable doit donc satisfaire les contraintes suivantes :

$$\forall i, r_i \leq t_i, C_i \leq \tilde{d}_i$$

Les contraintes de cohérence technologique, ou contraintes de gammes, décrivent des relations d'ordre entre les différentes tâches. La contrainte de ce type la plus connue est la contrainte de précédence : une opération O_j ne peut s'exécuter qu'après une opération O_i. Une telle contrainte s'exprime donc ainsi :

$$C_i \leq t_j$$

L'ensemble des prédécesseurs de l'opération O_i est noté $\Gamma^-(i)$. Symétriquement, l'ensemble des successeurs de l'opération O_i est noté $\Gamma^+(i)$. Ainsi, la contrainte de précédence décrite au paragraphe précédent s'écrit :

$$\Gamma^+(i) = \{j\}, \quad \Gamma^-(j) = \{i\}, \quad \Gamma^-(i) = \Gamma^+(j) = \varnothing$$

1.1.4.2 Les contraintes de ressources

Les contraintes de ressources traduisent le fait que ces ressources sont disponibles en quantité limitée (contraintes d'utilisation de machine, d'équipe d'ouvriers, *etc.*). On parle généralement de contraintes de partage.

Ainsi, dans notre cas de ressources disjonctives renouvelables, une machine ne peut exécuter qu'une opération à la fois. Soit m_i la machine utilisée par l'opération O_i, on peut alors exprimer cette contrainte ainsi :

$$\forall (i,j), m_i = m_j \Rightarrow C_i \leq t_j \text{ ou } C_j \leq t_i$$

1.1.5 Les objectifs

1.1.5.1 Satisfaction de contraintes

Trouver une solution à un problème d'ordonnancement consiste tout d'abord à trouver une solution satisfaisant toutes les contraintes : une solution réalisable.

Généralement, trouver un ordonnancement réalisable est une tâche aisée : un algorithme constructif arrive à prendre en compte les principales contraintes de l'ordonnancement (notamment les contraintes de ressources, de précédences et les dates de début au plus tôt), et permet donc la construction d'une telle solution rapidement et simplement.

Cependant, certaines contraintes ne peuvent être prises en compte facilement lors de la construction d'un ordonnancement. Par exemple, les dates de fin impératives ($\tilde{d}_i$) ne peuvent être prises en compte simplement, et peuvent même engendrer des problèmes ne possédant pas de solution.

Lors de la résolution d'un problème de satisfaction de contraintes, l'algorithme tente de trouver une solution réalisable. S'il en trouve une, il la propose, et le problème est résolu. Par contre, lorsque le problème ne possède pas de solution réalisable, l'algorithme ne peut rien proposer à l'utilisateur.

Pour pallier ce défaut, on reformule généralement de tels problèmes avec des contraintes molles : au lieu de rendre les solutions réalisables difficiles à trouver, on modélise le problème sous forme de préférences, et on cherche à obtenir une solution réalisable qui maximise nos préférences. On obtient alors un problème d'optimisation.

1.1.5.2 Optimisation d'un objectif

En règle générale, un problème d'ordonnancement est constitué de contraintes et d'un objectif. Cet objectif reflète le type de solution recherché. Le but est alors de trouver l'ordonnancement réalisable de meilleure qualité possible, c'est-à-dire minimisant ou maximisant l'objectif.

La littérature sur l'ordonnancement répertorie un grand nombre d'objectifs différents. Ce sont généralement des fonctions des dates de fin des opérations (C_i). L'objectif le plus étudié est le temps total de l'ordonnancement, appelé *makespan* en anglais, et calculé

$$C_{\max} = \max_i C_i$$

D'autres objectifs courants sont les objectifs relatifs aux dates de livraison d_i : leur but est de minimiser le retard ou l'avance des opérations par rapport à cette date. Par exemple, le retard algébrique maximum, appelé *maximum lateness* en anglais, se calcule ainsi :

$$L_{\max} = \max_i L_i = \max_i (C_i - d_i)$$

Deux caractéristiques principales sont utilisées pour classer les objectifs : la régularité de l'objectif, et le type d'opérateur (somme, maximum, *etc.*) utilisé

par la fonction. Ces caractéristiques permettent de déduire des propriétés sur les ordonnancements recherchés.

Une fonction objectif est dite régulière s'il s'agit d'une fonction monotone des dates de fin des opérations (C_i). Cette caractéristique s'exprime ainsi : on ne peut pas dégrader une fonction objectif monotone en avançant l'exécution d'une tâche.

Il est possible de catégoriser les objectifs en fonction des opérateurs utilisés dans la fonction, ainsi que par le sens d'optimisation de l'objectif (s'il faut minimiser ou maximiser l'objectif). Les deux classes principales sont :

minmax : le but est le minimiser une fonction pouvant s'exprimer sous la forme $\max_i f_i(C_i)$;

minsum : le but est de minimiser une fonction pouvant s'exprimer sous la forme $\sum_i f_i(C_i)$;

avec f_i une fonction quelconque relative à l'opération O_i. Si les fonctions f_i sont croissantes, alors l'objectif minmax ou minsum correspondant est régulier.

Il existe un grand nombre d'objectifs. Le tableau 1.1 décrit les principaux objectifs de la littérature, ainsi que leurs caractéristiques.

Nom	Notation	Formule	Régulier	Type
makespan	$C_{\max}$	$\max_i C_i$	oui	minmax
retard algébrique maximum	$L_{\max}$	$\max_i L_i$	oui	minmax
retard maximum	$T_{\max}$	$\max_i T_i$	oui	minmax
retard total	$\sum T_i$	$\sum_i T_i$	oui	minsum
nombre d'opérations en retard	$\sum U_i$	$\sum_i U_i$	oui	minsum
retard total absolu	$\sum \lvert L_i \rvert$	$\sum_i \lvert L_i \rvert$	non	minsum
objectif régulier quelconque	f	$f(C_1, \ldots, C_n)$	oui	–
objectif régulier quelconque de type minmax	$f_{\max}$	$\max_i f_i(C_i)$	oui	minmax
objectif régulier quelconque de type minsum	$\sum f_i$	$\sum_i f_i(C_i)$	oui	minsum

$$L_i = C_i - d_i, \quad T_i = \max(0, L_i), \quad U_i = \begin{cases} 1 \text{ si } L_i > 0 \\ 0 \text{ sinon} \end{cases}$$

Tab. 1.1: Quelques fonctions objectif pour l'ordonnancement

1.1.5.3 Optimisation multiobjectif

Dans certains cas, un objectif unique n'est pas suffisant : effectuer des compromis vis-à-vis de plusieurs objectifs est nécessaire. Par exemple, on peut vouloir limiter les retards, tout en utilisant au maximum les ressources. Un compromis entre ces deux objectifs est alors nécessaire, et on doit aborder le problème sous forme multiobjectif.

L'utilisation de plusieurs objectifs pose un problème : comment savoir si une solution est meilleure qu'une autre ? Pour remédier à ce problème, on définit une notion de dominance entre deux solutions. Une solution x dominée par une autre solution y signifie que la solution y est meilleure que la solution x pour tout objectif. Cette propriété s'exprime ainsi, pour K objectifs Z_o à minimiser :

x domine $y \Leftrightarrow \forall o \in \{1, \ldots, K\}, Z_o(x) \leq Z_o(y)$ avec au moins une inégalité stricte.

Ainsi, pour un ensemble de solutions donné, on peut séparer les solutions dominées des solutions non dominées. Les solutions non dominées sont les solutions intéressantes du point de vue des objectifs considérés. La figure 1.1 illustre graphiquement, dans un espace de deux objectifs, des solutions dominées et non dominées.

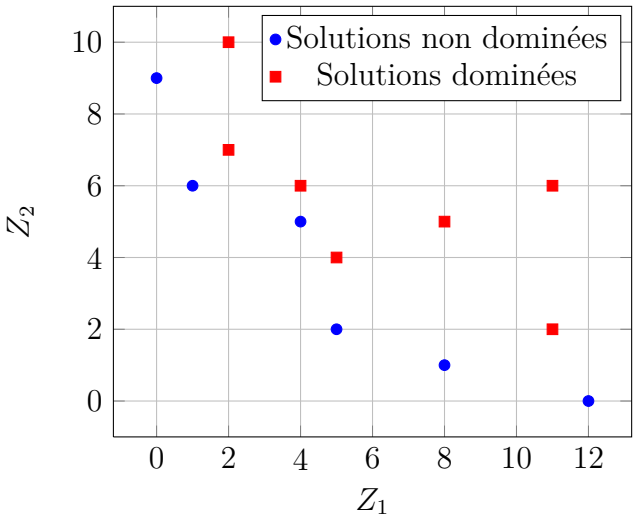


FIG. 1.1: Solutions dominées et solutions non dominées dans un espace de deux objectifs à minimiser

Une solution dominée par aucune solution réalisable est appelée un optimum de Pareto. L'ensemble des solutions dominées par aucune solution réalisable est appelé l'ensemble de Pareto.

Comme la solution à un problème multiobjectif est un ensemble de solutions, et non une solution unique, il faut choisir une solution dans cet ensemble. Pour choisir cette solution, il existe deux approches. Soit l'algorithme propose un ensemble

de solutions non dominées, et le décideur effectue *a posteriori* le compromis en choisissant une solution. Soit le décideur choisit *a priori* le compromis *via* des paramètres donnés à l'algorithme, et l'algorithme propose une solution liée à un problème mono-objectif, l'objectif étant une combinaison des objectifs originaux.

Les algorithmes recherchant un ensemble de solutions non dominées ont comme avantage qu'aucun choix ne doit être effectué *a priori*, et que le décideur peut choisir son compromis en connaissant les différentes solutions réalisables. Cependant, cette approche possède quelques inconvénients. Les algorithmes doivent être conçus de façon multiobjectif, il n'est donc pas possible d'utiliser directement des algorithmes mono-objectif pour résoudre le problème. Un autre inconvénient est que ces algorithmes nécessitent beaucoup plus de puissance de calcul puisqu'il faut générer un ensemble de solutions plutôt qu'une solution unique. On remarquera que des algorithmes exacts proposant un ensemble de solutions non dominées proposent comme solution l'ensemble de Pareto.

Pour transformer un problème multiobjectif en problème mono-objectif, il existe un grand nombre d'approches. Nous en abordons ici quelques-unes.

L'approche la plus simple et la plus utilisée est la somme pondérée des objectifs. Le décideur donne un poids α_o à chaque objectif. La fonction à optimiser devient alors

$$\sum_{o=1}^{K} \alpha_o Z_o$$

Une autre approche est l'optimisation lexicographique. Les objectifs peuvent être optimisés dans un ordre fixé. Tout d'abord, on optimise le premier objectif. Une fois le premier objectif optimisé, on fixe comme contrainte que cet objectif ne doit pas être dégradé, et on optimise le suivant. On continue ainsi jusqu'à ce qu'il n'y ait plus d'objectif. Cette méthode est utile lorsqu'une forte hiérarchie est présente entre les objectifs, le décideur classe alors les objectifs par ordre d'importance.

Finalement, l'approche ϵ-contrainte consiste à fixer pour tous les objectifs une borne supérieure à ne pas dépasser, à l'exception d'un objectif qui est optimisé. Le décideur choisit un objectif à optimiser et donne une borne à respecter pour tous les autres objectifs.

Il existe bien sûr beaucoup d'autres approches pour résoudre les problèmes multiobjectifs, nous avons décrit ici les plus courantes. Le tableau 1.2 énumère les différentes méthodes de résolution, ainsi que leur notation.

1.1.6 Ordonnancements actifs, semi-actifs et sans délai

La plupart des algorithmes de résolution de problème d'ordonnancement donnent en solution des classes spécifiques d'ordonnancement. Ces classes sont la classe des

Notation	Méthode
Z	Optimisation mono-objectif de l'objectif Z.
$\#(Z_1, \ldots, Z_K)$	Énumération de solutions non dominées.
$Lex(Z_1, \ldots, Z_K)$	Optimisation lexicographique des objectifs.
$F_\ell(Z_1, \ldots, Z_K)$	Somme pondérée des objectifs.
$\epsilon(Z_u/Z_1, \ldots, Z_{u-1}, Z_{u+1}, \ldots, Z_K)$	Approche ϵ-contrainte avec Z_u comme objectif à optimiser.

TAB. 1.2: Notations des méthodes de résolutions multiobjectifs

ordonnancements sans délai, la classe des ordonnancements actifs et la classe des ordonnancements semi-actifs.

Un ordonnancement semi-actif est un ordonnancement calé à gauche. Cela correspond donc à un ordonnancement où on exécute la tâche prévue au plus tôt. Nous avons donc une bijection entre séquencement et ordonnancement semi-actif.

Dans un ordonnancement actif, il est impossible d'avancer une tâche sans en reculer une autre. Cela signifie que l'ordonnancement est calé à gauche et sans période de disponibilité suffisamment importante entre deux tâches pour y insérer une autre tâche sans violer de contraintes. Un ordonnancement actif est également un ordonnancement semi-actif.

Un ordonnancement sans délai est un ordonnancement où une machine n'est jamais laissée inutilisée alors qu'elle pourrait l'être. Cela signifie que s'il y a une tâche dans la file d'attente d'une machine, alors cette machine exécute une tâche. Les ordonnancements à règles de priorité génèrent généralement de tels ordonnancements. Un ordonnancement sans délai est également un ordonnancement actif.

Considérons un problème à une machine et à trois opérations représenté dans la figure 1.2a.

L'ordonnancement 1.2b n'est pas un ordonnancement semi-actif : l'opération O_1 commence à $t = 10$, alors quelle pourrait commencer à $t = 9$ sans violer aucune contrainte, et sans changer l'ordre d'exécution des opérations sur la machine. Comme cet ordonnancement n'est pas semi-actif, il n'est ni actif, ni sans délai.

L'ordonnancement 1.2c est un ordonnancement semi-actif car toutes les opérations sont exécutées au plus tôt. Il n'est pas actif car on peut déplacer l'opération O_1 au début de l'ordonnancement sans retarder aucune opération. Comme il n'est pas actif, il n'est pas sans délai.

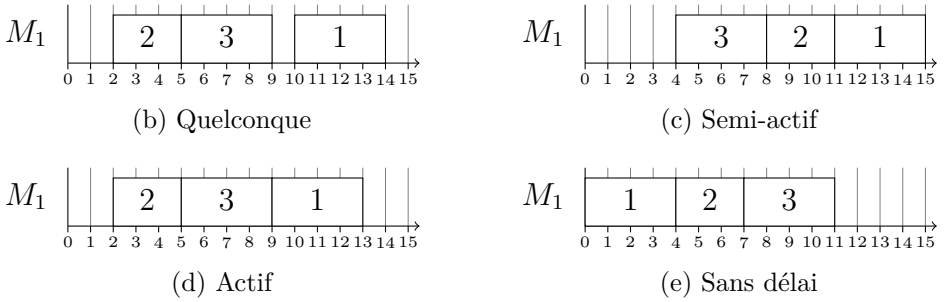
L'ordonnancement 1.2d est un ordonnancement actif car toutes les opérations sont exécutées au plus tôt et on ne peut avancer aucune opération sans en retarder une autre. Il n'est pas sans délai car, à $t = 0$, l'opération O_1 peut être exécutée.

L'ordonnancement 1.2e est un ordonnancement sans délai car, à chaque instant où une opération peut être exécutée, une opération est exécutée.

Un sous-ensemble d'ordonnancements est dit dominant pour l'optimisation

i	p_i	r_i
1	4	0
2	3	2
3	4	4

(a) Problème



(b) Quelconque (c) Semi-actif

(d) Actif (e) Sans délai

FIG. 1.2: Exemples d'ordonnancements semi-actif, actif et sans délai

d'un objectif donné s'il contient au moins un optimum pour cet objectif. L'ensemble des ordonnancements actifs est dominant pour les critères réguliers alors que l'ensemble des ordonnancements sans délai ne l'est pas. C'est pour cela que les algorithmes d'optimisation cherchent généralement une solution dans l'ensemble des ordonnancement actifs.

Les ordonnancements calculés sont presque toujours des ordonnancements semi-actifs [2] car toute séquence d'opérations peut être représentée par un unique ordonnancement semi-actif.

1.2 Classification des problèmes d'ordonnancement

Comme il existe beaucoup de variantes du problème d'ordonnancement, un effort de classification est fait dans la littérature. Cette classification permet de décrire simplement et efficacement le problème d'ordonnancement. Grâce à cette classification, il est facile d'identifier un problème d'ordonnancement, et ce, sans ambiguïté.

La notation utilisée fut d'abord proposée par [GLLRK79] puis étendue par [Bru98]. Cette notation se décompose en trois champs α, β et γ. Un problème se décrit par ces trois champs de la manière suivante : $\alpha|\beta|\gamma$.

Le champ α décrit la catégorie du problème d'ordonnancement : le nombre de

2. Certains algorithmes incluent des temps morts dans les ordonnancements pour augmenter les marges. De tels algorithmes ne génèrent pas d'ordonnancements semi-actifs. [Gol97] présente un algorithme de ce genre

machines et leur organisation. Le champ β décrit les contraintes supplémentaires. Le champ γ décrit l'objectif à optimiser.

1.2.1 Le champ α

Le champ α se divise en deux sous champs concaténés α_1 et α_2. Le champ α_2, optionnel, indique le nombre de machines dans le problème. Si ce champ est vide, le nombre de machines est une variable du problème.

Le champ α_1 donne une catégorie du problème, en fonction de l'organisation des machines. Deux dimensions différentes en ressortent.

Tout d'abord, les problèmes d'ordonnancements sont classés suivant le parcours des produits sur les différentes machines, c'est-à-dire suivant les caractéristiques des contraintes de précédences entre opérations :

- Les ateliers à cheminement unique (*flow shop*) : Chaque travail est constitué de m opérations et l'ordre de passage sur les différentes machines est le même pour tous les travaux (contraintes de précédences identiques pour tout travail).
- Les ateliers à cheminements multiples (*job shop*) : Le nombre d'opérations n'est pas forcément le même pour tous les travaux et chaque travail a son propre ordre de passage sur les machines (contraintes de précédences propres à chaque travail).
- Les ateliers à cheminements libres (*open shop*) : Le nombre d'opérations n'est pas forcément le même pour tous les travaux et l'ordre de passage sur les machines est totalement libre (pas de contrainte de précédences).

D'autre part, les problèmes d'ordonnancements sont classés vis-à-vis du problème d'affectation des opérations aux machines :

- Dans les problèmes d'ordonnancement, les opérations ne peuvent être exécutées que sur une unique machine.
- Dans les problèmes d'ordonnancement et d'affectation, chaque opération est affectée à un *pool* de machines, les *pools* formant une partition des machines. Le problème est alors, en plus du problème d'ordonnancement classique, d'affecter les opérations aux machines.
- Dans les problèmes d'ordonnancement et d'affectation généralisée, les opérations peuvent être exécutées sur un ensemble de machines quelconques.

Les principales notations du champ α_1 sont énumérées dans le tableau 1.3.

1.2.2 Le champ β

Le champ β permet de décrire des contraintes additionnelles au problème. Il est constitué d'une suite de sous champs séparés par des virgules. Les principaux sous champs sont décrits dans le tableau 1.4.

Valeur	Description
$\varnothing$	Machine unique, la valeur du champ α est alors « 1 ».
P	Machines parallèles identiques.
Q	Machines parallèles proportionnelles.
R	Machines parallèles non reliées.
F	*Flow shop.*
J	*Job shop.*
O	*Open shop.*
FH	*Flow shop* hybride.
JG	*Job shop* généralisé.
OG	*Open shop* généralisé.
XAG	Problème de type X (avec $X \in \{P, Q, R, F, J, O\}$) avec affectation générale.

TAB. 1.3: Principales valeurs du champs α_1

Valeur	Description
d_i	Les opérations ont des dates de fin souhaitées. Généralement, cet indicateur peut être omis car l'existence de telles dates se retrouve dans la fonction objectif utilisée, comme par exemple $L_{\max}$.
$\tilde{d}_i$	Les opérations ont des dates de fin impératives.
$pmtn$	La préemption des opérations est possible.
$prec$	Contraintes de précédences quelconques entre opérations.
r_i	Les opérations ont des dates de début au plus tôt.
S_{sd} (R_{sd})	L'exécution d'une opération est précédée (resp. suivie) d'un temps de montage (resp. démontage) dépendant de la séquence des travaux sur la machine. Ce temps mobilise uniquement la ressource.

TAB. 1.4: Principaux sous champs du champs β

1.2.3 Le champ γ

Le champs γ décrit l'objectif à optimiser. Si le but est de trouver une solution réalisable (pas d'optimisation), on note $-$ pour ce champ. Sinon, la valeur de ce champ correspond aux notations décrites à la section 1.1.5, notamment les tableaux 1.1 et 1.2.

1.2.4 Quelques exemples

Le problème du *job shop*, tel qu'il est abordé dans [ABZ88, BJS94, CP89, CP90, CP94] se note $J||C_{\max}$, ce qui se traduit par « problème de *job shop* à m machines avec comme objectif le *makespan.* ».

Le problème de satisfaction des dates de livraison pour un atelier de type *job shop* se note $J|\tilde{d}_i|-$.

Le *one machine problem*, décrit dans [Car82] est souvent utilisé comme une relaxation du *job shop*, se note $1|r_i|L_{\max}$ ou $1|r_i, d_i|L_{\max}$. C'est donc le problème à une machine, avec des dates de début au plus tôt dont l'objectif est le retard algébrique maximum.

Le *flow shop* à deux machines, résolu optimalement par l'algorithme de Johnson [Joh54], se note $J2||C_{\max}$.

1.3 Ordonnancement théorique et pratique

Comme nous l'avons vu, l'ordonnancement est décrit comme un problème aux données statiques et exactes. En pratique, l'ordonnancement d'atelier est bien différent. L'ordonnancement est une des fonctions de la gestion de production et, à ce titre, est en relation avec les autres fonctions de la gestion de production (planification, suivi de production, gestion d'ateliers, *etc.*). De plus, très souvent, la réalité de l'atelier n'est pas aussi déterministe que dans les problèmes d'ordonnancement présentés. Par exemple des pannes ou des retards peuvent avoir lieu durant l'exécution de l'ordonnancement.

1.3.1 Contexte productique de l'ordonnancement d'atelier

L'ordonnancement d'atelier se fait dans un contexte particulier. Ce contexte est défini par la gestion de la production utilisée par l'entreprise.

En général, on effectue :

- La planification à long terme. Elle se fait sur plusieurs années. Elle exprime la stratégie de l'entreprise.
- La planification à moyen terme, qui s'effectue sur plusieurs mois. Elle prévoit la charge de l'atelier et constitue l'entrée du calcul des besoins.

- La planification à court terme. Elle génère les ordres de fabrications et d'approvisionnements. Cette planification est généralement réalisée par un ordonnancement à capacité infinie.
- L'ordonnancement d'atelier en lui-même. Il utilise la planification à court terme comme entrée et génère l'affectation des tâches aux ressources.

On remarquera que, typiquement, les dates de début au plus tôt des opérations ainsi que les dates de fin souhaitées sont déterminées pendant la planification à court terme. En pratique, il est possible de renégocier ces dates données par la planification. L'ordonnancement se trouve donc au sein d'une organisation complexe gérée par des hommes.

1.3.2 Ordonnancement et déterminisme

Dans ce chapitre, l'ordonnancement est présenté comme un problème totalement déterministe, avec des temps opératoires fixes, et des objectifs très précis à optimiser. En pratique, l'ordonnancement d'atelier n'est pas si déterministe : des livraisons sont en retard, des opérations prennent plus de temps que prévu ou un client a besoin de son produit plus tôt que ce qu'il avait demandé à l'origine.

Pour pallier ces différentes incertitudes, des travaux de recherches tentent de les prendre en compte. Dans le chapitre suivant, nous présentons les différentes familles de méthodes proposées.

Chapitre 2

Les incertitudes en ordonnancement

Ce chapitre présente les incertitudes en ordonnancement d'atelier. L'objectif de ce chapitre est d'introduire au lecteur le domaine de l'ordonnancement sous incertitudes, et non de faire un état de l'art complet. Pour plus d'information sur ce domaine, le lecteur pourra se référer à l'état de l'art [DB00] ainsi qu'au livre [BMS04a].

Dans un premier temps, nous présenterons les différentes définitions utilisées dans la littérature. Ensuite, nous exposerons les différentes méthodes de résolution, ainsi que quelques exemples illustratifs.

2.1 Définitions

2.1.1 Les Phases de l'ordonnancement

Une méthode d'ordonnancement peut se découper en plusieurs parties. En pratique, on distingue deux parties principales : la phase prédictive, qui a lieu avant l'exécution de l'ordonnancement dans l'atelier, et la phase réactive, qui a lieu pendant l'exécution.

La phase prédictive a lieu avant l'exécution de l'ordonnancement, c'est-à-dire que l'exécution de l'ordonnancement est « prédit ». Cette phase n'est pas soumise à des contraintes fortes de temps : il n'est pas nécessaire de répondre à des sollicitations en un temps imparti. Cette phase dure généralement un certain temps, entre quelques heures et quelques semaines suivant le problème et le type d'atelier. Cette phase repose intégralement sur un modèle : elle n'est pas réalisée sur l'atelier, mais hors ligne, le plus souvent en utilisant un ordinateur. L'efficacité de cette phase repose essentiellement sur la modélisation du problème, et les outils disponibles pour le résoudre. Cette phase peut également être appelée « phase proactive » si une méthode proactive ou proactive-réactive est utilisée (voir la section 2.2).

La phase réactive a lieu pendant l'exécution de l'ordonnancement, c'est-à-dire

qu'elle « réagit » aux événements réels qui ont lieu dans l'atelier. Durant cette phase, les décisions à réaliser par le système doivent être prises rapidement, avec généralement un temps de réaction imposé. Ce temps varie en pratique entre quelques secondes et quelques heures suivant le problème. Il n'est donc pas envisageable d'utiliser des algorithmes très coûteux en temps de calcul dans cette phase. Comme le but de cette phase est de réagir en temps réel aux événements de l'atelier, la méthode de résolution possède une connaissance exacte des événements tels qu'ils se sont passés, car ils sont rapportés par le système réel, et non par un modèle comme dans la phase prédictive.

Ces deux phases sont des notions primordiales pour l'ordonnancement sous incertitudes : chaque phase possède des données différentes sur les incertitudes.

2.1.2 Incertitudes

Les incertitudes sont la principale difficulté dans l'utilisation de l'ordonnancement en pratique. Nous prendrons comme définition celle que propose Esswein dans [Ess03] :

> On parle d'incertitudes pour désigner les modifications potentielles des données d'un problème d'ordonnancement qui peuvent intervenir entre le calcul d'un ordonnancement et la fin de sa mise en œuvre réelle dans l'atelier.

Les incertitudes correspondent donc à la différence entre les prévisions effectuées durant la phase prédictive et les données réelles obtenues après la phase réactive.

En pratique, dans le cas de l'ordonnancement d'atelier, les incertitudes généralement étudiées sont :

- Le changement de la durée d'une opération. C'est le cas des opérations en retard, mais également de celles en avance.
- L'insertion ou la suppression d'une opération. Cela arrive fréquemment, généralement sous la forme d'une commande urgente à traiter.
- La panne machine.
- La différence entre le modèle et la réalité. Par exemple, la non-présence des temps de transport entre ressources dans le modèle alors que dans la réalité, ce temps n'est pas négligeable.

Ces incertitudes peuvent avoir une influence importante dans l'ordonnancement d'atelier. Il est donc utile de les prendre en compte. La flexibilité et la robustesse ont pour objectifs de mesurer l'influence des incertitudes et de tenter de les limiter.

2.1.3 Flexibilité et robustesse

Le groupe de travail flexibilité du GOThA (Groupe de recherche en Ordonnancement Théorique et Appliqué) a réalisé une étude de la flexibilité et de la

robustesse en ordonnancement. Les résultats de ces recherches sont exposés dans [BMS04b, Gro02]. La suite de cette section est fortement inspirée de ces articles.

2.1.3.1 Flexibilité

La flexibilité d'un ordonnancement représente les degrés de liberté présents *a priori* dans cet ordonnancement.

Elle peut prendre plusieurs formes :

- La flexibilité temporelle, c'est-à-dire sur les dates de début d'exécution des opérations. Cette flexibilité est souvent implicite en ordonnancement. Elle permet aux opérations de dériver dans le temps, et ainsi de résoudre les différents problèmes de contraintes. Elle est la base de la flexibilité en ordonnancement.
- La flexibilité séquentielle, c'est-à-dire sur l'ordre des séquences d'opérations sur les machines. Elle permet de modifier les séquences de passage des opérations sur les machines. Elle est très intéressante en cas de retard d'une opération : ce retard peut alors ne pas affecter les autres opérations. La flexibilité temporelle est requise pour son fonctionnement.
- La flexibilité sur les affectations, c'est-à-dire sur les affectations des opérations aux machines. Il est ainsi possible de déplacer une opération d'une machine à une autre. La flexibilité séquentielle et la flexibilité temporelle sont nécessaires à son fonctionnement. Cette flexibilité est très pratique en cas de panne de machine.
- La flexibilité sur les modes d'exécution, c'est-à-dire sur la modification du modèle du problème, comme par exemple le changement de gamme ou la modification du nombre de ressources nécessaires pour effectuer une opération.

Un problème donné possède une flexibilité intrinsèque. Un ordonnancement exécuté n'en possède pas. La flexibilité intrinsèque est généralement dégradée de manière à obtenir une certaine performance. Ainsi, un algorithme d'optimisation classique va éliminer totalement cette flexibilité pour donner une solution dont on peut mesurer la (bonne) performance. Un compromis doit donc être trouvé entre performance et flexibilité.

La mesure de la flexibilité dépend du type de flexibilité à mesurer. En effet, les mesures des différentes flexibilités sont indépendantes les unes des autres, et ne possèdent pas de rapports triviaux les unes entre les autres. Il est donc difficile de comparer deux sortes de flexibilité.

De plus, même en ne prenant qu'une forme de flexibilité, il n'existe pas de mesure objective de la flexibilité. Cette mesure va être influencée par différents paramètres :

- Le problème en lui-même. En effet, certaines caractéristiques du problème devraient influencer la mesure de la flexibilité. Par exemple, la flexibilité peut être utile uniquement à certains endroits précis dépendant du problème

comme, par exemple, sur la chaîne critique. Il semblerait alors judicieux de pondérer la flexibilité en fonction du niveau critique. Dans ce cas, on a alors une flexibilité dépendant du problème.
- La manière d'utiliser la flexibilité. Prenons le cas de la flexibilité séquentielle. Une méthode de mesure objective pourrait être le nombre d'ordonnancements semi-actifs. Mais si la méthode d'utilisation de la flexibilité ne peut utiliser qu'un petit sous-ensemble de ces ordonnancements pendant la partie réactive, notre mesure de la flexibilité n'est alors pas valide en pratique. C'est par exemple le cas si une règle de priorité est utilisée, seuls les ordonnancements sans délai seront utilisés. Par exemple, [Alo02] propose différentes mesures de flexibilité : le nombre d'ordonnancements semi-actifs, actifs et sans délai. La mesure est alors sélectionnée en fonction de sa pertinence vis-à-vis du contexte.
- La méthode utilisée pour générer la flexibilité. En effet, la mesure de la flexibilité d'un ordonnancement est souvent le critère à optimiser. Il peut être, dans ce cas, judicieux d'adapter la mesure de flexibilité à la méthode d'optimisation. Par exemple, [La05] utilise comme mesure de flexibilité la « cardinalité de l'ensemble de séquences dominantes », qui est une borne inférieure du nombre de séquences, découlant directement du théorème des pyramides. [Ess03] utilise le nombre de groupes, qui est une mesure propre à l'ordonnancement de groupes. Nous remarquons que ces méthodes utilisent des mesures de flexibilité adaptées à leur méthode.

2.1.3.2 Robustesse

La robustesse est un terme utilisé dans beaucoup de domaines, et souvent avec une définition floue ou inexistante. En règle générale, on peut définir un ordonnancement robuste comme un ordonnancement peu sensible aux incertitudes [BMS04b].

Nous pouvons remarquer un lien certain entre flexibilité et robustesse. En effet, la flexibilité est un outil pour réaliser un ordonnancement robuste : l'utilisation de la flexibilité pendant la réalisation d'un ordonnancement donnera un résultat plus robuste que si la flexibilité n'est pas utilisée (ou n'est pas présente).

De plus, un ordonnancement peut être robuste sans être flexible. Par exemple, [Gol97] est une méthode qui place des temps morts entre certaines opérations. Si ces opérations avaient des temps opératoires augmentant dans la limite de la taille des temps morts insérés, cet ordonnancement absorberait des incertitudes sans effectuer aucune modification par rapport à l'ordonnancement prévu. Cela donnerait donc un ordonnancement robuste et non flexible.

2.2 Méthodes pour l'ordonnancement sous incertitudes

[DB00] propose une classification des méthodes d'ordonnancement sous incertitudes qui est maintenant la classification la plus utilisée dans la littérature. Les phases de l'ordonnancement pendant lesquelles les incertitudes sont prises en compte permettent de classer ces méthodes. Il en ressort trois catégories principales : les méthodes proactives, les méthodes réactives et les méthodes proactives-réactives.

2.2.1 Approche proactive

Une approche proactive focalise son travail sur la phase prédictive. La partie réactive est triviale : l'ordonnancement calculé durant la phase prédictive est suivi strictement. La flexibilité temporelle est souvent utilisée dans de telles approches pour satisfaire les contraintes : si une opération n'est pas réalisable alors qu'elle est prévue dans l'ordonnancement calculé, on attend qu'elle le devienne.

Cette méthode possède un certain nombre d'avantages :

- La contrainte de temps de calcul de l'ordonnancement n'est pas très importante. En effet, il n'est pas nécessaire de donner une réponse rapidement vu que le travail est fait en dehors de l'exécution de l'ordonnancement. Ceci permet de faire de l'optimisation, et donc d'obtenir de meilleures performances.
- Il est possible de prédire le futur. C'est en effet réalisable grâce au temps de calcul disponible. Ces prédictions permettent de calculer la performance de l'ordonnancement calculé, ainsi que de prendre en compte certaines contraintes comme les dates dues impératives qu'il est impossible de vérifier sans ces prédictions.

Elle possède également des inconvénients :

- Tout le travail est effectué sur le modèle. Si ce modèle est faux ou pas assez précis, cette méthode n'est pas utilisable. Cela pose problème dans le cas d'atelier très complexe qu'il est difficile de modéliser fidèlement, ou si la méthode proactive utilise un modèle inadapté à notre problème.
- Les incertitudes ne peuvent pas être prises en compte directement. On peut utiliser une modélisation des incertitudes dans cette méthode, mais les évènements réels ne seront pas pris en compte. Si les incertitudes réelles s'éloignent trop de leur modélisation, la phase réactive sera mauvaise, voire même impossible à réaliser.

Cette méthode est donc adaptée aux environnements prédictifs ou avec peu d'incertitudes. La littérature sur ce sujet est assez abondante.

La méthode la plus connue, notamment en gestion de projets, est la méthode de la chaîne critique développée dans [Gol97]. Elle est aujourd'hui utilisée par plusieurs logiciels de gestion de projet. L'idée directrice est d'insérer, durant la phase statique, des temps morts aux endroits adéquats du projet. Cela a pour effet

d'augmenter les marges de la chaîne critique. Le but est d'éviter les dépassements d'échéances dus à des retards.

Une autre méthode proactive utilisant, elle, un ensemble d'ordonnancement est l'ordonnancement « au cas où » (*Just In Case scheduling*) développé dans [SBD94]. L'algorithme identifie les retards critiques probables et propose pour chacun d'eux, un ordonnancement alternatif des opérations restantes. Cette méthode est intéressante car elle utilise la flexibilité séquentielle et non plus uniquement la flexibilité temporelle comme c'est généralement le cas dans les approches statiques. Par contre, l'explosion combinatoire du problème rend cette méthode inutilisable pour les problèmes de grande taille.

2.2.2 Approche réactive

Les approche réactives utilisent les événements réels de l'atelier, pour évaluer les incertitudes, les prendre en compte, et y « réagir ». Il existe deux approches différentes :

- les approches dynamiques, qui ne font aucun travail pendant la phase prédictive, et construisent intégralement l'ordonnancement en ligne ;
- les approches prédictives-réactives, qui se basent sur un ordonnancement statique calculé pendant la phase réactive, et effectuent des réordonnancements pendant la phase réactive pour prendre en compte l'état réel de l'atelier.

2.2.2.1 L'approche dynamique

L'approche dynamique, aussi appelée approche totalement réactive, génère l'ordonnancement directement pendant la phase réactive de l'ordonnancement. Voir [BC01] pour un aperçu complet de cette approche.

Son avantage principal est d'utiliser les données réelles, et non des données modélisées et calculées. Ainsi, l'ordonnancement généré par une telle méthode est réalisable dans tous les cas. Une autre conséquence est que les incertitudes peuvent être directement prises en compte par le système.

En revanche, cette méthode possède un gros défaut : elle ne prédit pas l'ordonnancement, mais le construit petit à petit (on parle alors de méthode constructive). Il est donc impossible d'obtenir une prévision de la performance de l'ordonnancement. Certaines contraintes ne peuvent donc être prises en compte. C'est le cas des dates dues impératives qui ne peuvent être respectées qu'avec une prédiction de l'ordonnancement. Cette méthode donne généralement des performances sous-optimales ou, en tout cas, instables (voir [BC01, VAL96]).

La plupart de ces approches utilisent les règles de priorité. L'idée est de classer les tâches à effectuer grâce à une règle pour ensuite choisir, lors de l'exécution, la tâche la mieux classée comme la tâche à exécuter sur la machine courante. Cette technique est très simple à mettre en œuvre et demande peu de temps de calcul. Un grand nombre de règles de priorité existe [Hau89]. La principale difficulté est de trouver la règle à appliquer dans un contexte donné. Par exemple, [PM97]

sélectionne dynamiquement la règle à appliquer en fonction du contexte, alors que [Dim05] génère une règle de priorité adaptée à une instance spécifique.

Pour choisir les règles de priorité les plus adaptées au contexte courant, la simulation est la méthode la plus utilisée. L'idée est de modéliser le problème puis ensuite de simuler la production avec différentes règles. Les informations obtenues lors de ces simulations vont permettre de détecter les règles de priorité les mieux adaptées.

Cette approche est présente dans certains outils de gestion de production. Très efficace dans un environnement peu ou pas prévisible, elle est également intéressante lorsqu'une qualité optimale n'est pas nécessaire, et tire alors ses avantages de sa simplicité.

2.2.2.2 Approche prédictive-réactive

Les approches prédictives-réactives prennent en compte les incertitudes uniquement pendant la phase réactive. Mais, contrairement à l'ordonnancement dynamique, la phase prédictive est utilisée : durant cette phase, un ordonnancement est calculé, sans prendre en compte les incertitudes. Ensuite, pendant la phase réactive, l'ordonnancement prédictif est utilisé jusqu'à ce que les incertitudes le rendent obsolète. À ce moment, un réordonnancement est effectué : l'ordonnancement prédictif d'origine est modifié pour prendre en compte l'état réel de l'atelier. Ces approches prédictives-réactives diffèrent entre elles sur deux axes : le moment où l'ordonnancement prédictif est considéré comme obsolète (ce qui va donner la fréquence des réordonnancements), et le degré de modification effectué par le réordonnancement (d'une petite modification de l'ordonnancement d'origine à un ordonnancement totalement différent). [BC01] décrit un grand nombre d'algorithmes de ce type.

2.2.3 Approche proactive-réactive

Une approche proactive-réactive tente de combiner les méthodes proactives et réactives : elle utilise la phase prédictive et la phase réactive de manière non triviale. Dans un premier temps, un ou plusieurs ordonnancements sont générés. Ensuite, la phase dynamique utilise et adapte ces ordonnancements en temps réel.

Cette méthode permet de combiner les avantages des différentes approches précédentes. En effet, comme dans l'approche proactive, les prédictions et les optimisations sont possibles durant la phase prédictive. De plus, la phase réactive permet de gérer réellement les incertitudes.

Le travail se base toujours sur un modèle, et donc, tout comme pour l'approche statique, la qualité de l'approche prédictive-réactive dépend beaucoup de la qualité du modèle. Mais cette importance est moindre que pour l'approche statique car le résultat calculé est confronté à la réalité durant la phase réactive.

[AP04] et [Alo02] propose une approche proactive-réactive se basant sur un ordre partiel des opérations à exécuter. Le problème étudié est un problème à une

machine. La flexibilité séquentielle, la flexibilité temporelle, la qualité dans le pire des cas et la qualité dans le meilleur des cas sont les objectifs utilisés. La phase prédictive génère un ordre partiel entre les opérations en optimisant ces différents objectifs. Une méthode réactive utilisant cet ordre partiel est également proposée.

[BHLL04] et [La05] propose une méthode proactive-réactive proposant un ensemble de séquences basé sur une règle de dominance pour un problème à une machine. Cette structure spéciale permet de calculer une borne inférieure de la qualité dans le meilleur des cas, la qualité dans le pire des cas, ainsi qu'une borne inférieure du nombre de séquences. Cet outil est utilisé dans [BOB08] pour la réalisation d'une approche coopérative.

Une autre approche originaire du LAAS-CNRS de Toulouse (France) est basée sur les groupes d'opérations permutables, concept créé il y a plus de 25 ans. Elle est notamment développée dans [Tho80, Bil93, Art97, Ess03]. [WBS99] utilise une structure identique dans son approche *Preprocess First Schedule Later* (PFSL). L'idée est de générer un ensemble d'ordonnancements calculé durant la phase prédictive, puis de l'utiliser durant la partie réactive. Pour ce faire, les ordonnancements sont représentés par des groupes d'opérations permutables : au lieu de proposer une opération sur une machine à un moment donné, il est proposé un ensemble d'opérations possibles à un moment sur une machine. Ainsi, un grand nombre d'ordonnancements peut être proposé sans devoir les énumérer. L'avantage de cette méthode est que la qualité de l'ordonnancement dans le pire des cas est calculable en temps polynomial. Un nombre important de types de contraintes est disponible pour ce modèle (comme les temps de préparation).

2.3 Choix de la méthode d'ordonnancement

Pour la suite de notre étude, nous avons besoin d'une méthode d'ordonnancement flexible et puissante permettant la mise en place d'un système homme-machine efficace. La méthode choisie est l'ordonnancement de groupes. Différents avantages de cette méthode nous ont conduits à ce choix.

Tout d'abord, c'est une méthode proactive-réactive. Nous pensons que ce type d'approche est un compromis intéressant permettant de faire à la fois de l'optimisation et de réagir aux événements réels de l'atelier.

L'ordonnancement de groupes est l'une des plus anciennes et des plus étudiées des méthodes proactives-réactives. La littérature sur l'ordonnancement de groupes nous permet donc de commencer sur des bases théoriques fortes et reconnues. De plus, cette méthode est utilisée dans le progiciel ORDO [RBV95]. Cette méthode est donc utilisée dans des cas réels, ce qui valide le modèle.

L'ordonnancement de groupes est une méthode fonctionnant sur de nombreux types différents d'ateliers. Cette généricité est l'une de ces forces. De plus, cette méthode possède un grand nombre d'outils, permettant de l'évaluer, de générer des ordonnancements de groupes, ainsi que de surveiller son exécution. Ces outils nous fournissent une bonne base pour apporter nos contributions à la méthode.

Finalement, l'ordonnancement de groupes possède un grand avantage par rapport aux autres méthodes proactives-réactives : sa compréhension par le décideur. En effet, un humain arrive assez facilement à comprendre la notion de groupe d'opérations permutables, ce qui permet de faciliter la coopération entre l'homme et la machine.

Nos travaux utilisent donc l'ordonnancement de groupes comme méthode d'ordonnancement sous incertitudes. Dans le chapitre suivant, nous allons décrire en détail cette méthode et ses outils.

Chapitre 3

L'ordonnancement de groupes

Ce chapitre présente l'ordonnancement de groupes, la méthode d'ordonnancement utilisée dans cette recherche. Tout d'abord, l'ordonnancement de groupes est défini et ses différentes propriétés sont expliquées. Les principaux algorithmes de l'ordonnancement de groupes sont ensuite formulés, et un algorithme de génération d'ordonnancement de groupes y est amélioré. Après une analyse de la littérature sur la robustesse de l'ordonnancement de groupes, nous proposons une étude sur la robustesse de l'ordonnancement de groupes vis-à-vis de la non-modélisation des temps de transport interopératoires.

3.1 Description de l'ordonnancement de groupes

On trouve dans [ER89] une première présentation des groupes d'opérations permutables. L'objectif de cette méthode est de fournir de la flexibilité séquentielle pendant l'exécution de l'ordonnancement ainsi que de garantir une qualité minimale correspondant au pire des cas. Cette méthode est largement étudiée depuis vingt ans, notamment dans [ER89, BR96, WBS99, ABE05]. Ce dernier article fournit une description théorique de la méthode.

3.1.1 Définitions

Un groupe d'opérations permutables est un ensemble d'opérations à exécuter sur une certaine ressource dans un ordre arbitraire. Il est noté $g_{\ell,k}$. Le groupe contenant l'opération O_i est noté $g(i)$. Dans ce document, l'indice k correspond au numéro de séquence du groupe sur la machine.

Un ordonnancement de groupes est défini par une liste ordonnée de v_ℓ groupes (d'opérations permutables) pour chaque machine M_ℓ : $g_{\ell,1}, \ldots, g_{\ell,v_\ell}$. Les groupes sont exécutés sur leur machine dans l'ordre de la liste. Sur une machine, le groupe après (resp. avant) $g(i)$ est noté $g^+(i)$ (resp. $g^-(i)$).

Un ordonnancement de groupes est réalisable si chaque permutation sur les opérations d'un même groupe donne un ordonnancement réalisable (c'est-à-dire

un ordonnancement qui ne viole pas de contraintes). Ainsi, un ordonnancement de groupes décrit un ensemble d'ordonnancements valides, sans les énumérer.

3.1.2 Évaluation de la qualité d'un ordonnancement de groupes

Les indicateurs de qualité d'un ordonnancement de groupes sont les mêmes que pour un ordonnancement classique. Ainsi, la qualité d'un ordonnancement de groupes correspond à la pire qualité présente dans l'ensemble des ordonnancements semi-actifs, comme défini dans [ABE05].

[Ess03] propose une évaluation complémentaire au pire des cas : le meilleur des cas. Dans ce cas, la qualité de l'ordonnancement de groupes correspond à la meilleure qualité présente dans l'ensemble des ordonnancements semi-actifs. Cette évaluation est complémentaire au pire des cas : avec ces deux évaluations, l'ensemble des qualités possibles de l'exécution d'un ordonnancement de groupes est représenté par un intervalle.

3.1.3 Représentation graphique

La représentation sous forme de diagramme de Gantt d'un ordonnancement de groupes consiste à représenter chaque groupe dans le pire des cas. Le pire des cas d'un groupe correspond à commencer l'exécution du groupe par l'opération possédant la plus grande date de début au plus tôt dans le pire des cas ($\underline{\tau}_i^{\Pi}$, voir la section 3.2.1), puis d'exécuter par la suite toutes les autres opérations (qui seront donc exécutées sans temps d'attente de la machine).

Cette représentation permet de visualiser facilement un ordonnancement de groupes.

3.1.4 Évaluation de la flexibilité

[Ess03] propose de définir la phase proactive de l'ordonnancement de groupes comme un problème d'optimisation biobjectifs avec comme objectifs la qualité de l'ordonnancement de groupes dans le pire des cas et la flexibilité de l'ordonnancement de groupes. Pour résoudre un tel problème, une mesure de la flexibilité est nécessaire.

[ABE05] et [Ess03] présentent deux mesures de la flexibilité d'un ordonnancement de groupes : le nombre d'ordonnancements semi-actifs décrit par l'ordonnancement de groupes, et le nombre total de groupes de l'ordonnancement de groupes.

3.1.4.1 Le nombre de séquences dans un ordonnancement de groupes

Représenter la flexibilité d'un ordonnancement de groupes par le nombre d'ordonnancements semi-actifs (ou de séquences) semble logique. D'ailleurs, les autres

méthodes proactive-réactive d'ordonnancement basées sur des représentations implicites d'un ensemble d'ordonnancements valides utilisent également cette mesure (voir la section 2.2.3).

Le nombre d'ordonnancements semi-actifs dans l'ordonnancement de groupes Π sera noté $\#\text{Seq}(\Pi)$. Calculer le nombre d'ordonnancements semi-actifs contenus dans un ordonnancement de groupes est plutôt simple : il suffit de calculer le nombre de séquences d'opérations possibles pour chaque groupe, et de les multiplier entre eux :

$$\prod_{\ell} \prod_{k=1}^{v_\ell} \|g_{\ell,k}\|!$$

3.1.4.2 Flexibilité et nombre de groupes

Une autre mesure de la flexibilité est le nombre de groupes : moins il y a de groupes dans un ordonnancement de groupes, plus il y a de flexibilité. Le nombre de groupes dans un ordonnancement de groupes Π est noté $\#\text{Gps}(\Pi)$.

Une flexibilité minimale correspond à autant de groupes que d'opérations, et une flexibilité maximale correspond à un groupe par machine [1]. En utilisant cette mesure, la flexibilité est bornée, il est donc possible de l'exprimer sous forme de taux. [Ess03] propose une telle représentation de la flexibilité :

$$\phi_\Pi = \frac{n - \#\text{Gps}(\Pi)}{n - m}$$

Remarquons que cette mesure est la même qu'un autre indicateur de flexibilité : le nombre de décisions à effectuer durant la phase réactive. En effet, pour un groupes composé de k opérations, l'opérateur aura $k-1$ décisions à réaliser pour ordonnancer le groupe. Le nombre total de décisions est donc égal au nombre d'opérations moins le nombre de groupes, ce qui correspond à la partie supérieure de l'expression de ϕ_Π. En ne comptant pas la dernière opération de chaque machine comme une possibilité de décision, ϕ_Π correspond au rapport entre les décisions proposées par l'ordonnancement de groupes par rapport au nombre d'opérations à exécuter donnant la possibilité à une décision. Ainsi, une flexibilité de $\phi_\Pi = 2/3$ signifie que deux opération sur trois sont exécutées suite à une prise de décision, et qu'une opération sur trois est exécutée sans avoir de choix sur l'opération à exécuter.

3.1.4.3 Comparaison des deux mesures de flexibilité

De ces deux indicateurs, $\#\text{Seq}(\Pi)$ semble être le plus intéressant car il représente mieux la flexibilité séquencielle. Par contre, cet indicateur est plus difficile à optimiser comme évoqué dans [ABE05]. De plus, cet indicateur est difficile à

1. Un ordonnancement de groupes avec un groupe par machine n'est pas réalisable pour un problème de type *job shop* car toutes les permutations ne sont pas des ordonnancements valides vis-à-vis des contraintes de précédence.

appréhender par un humain car il n'est pas borné et atteint facilement de très grandes valeurs.

Par contre, comme montré dans la section précédente, la mesure de flexibilité ϕ_Π est un indicateur de la flexibilité adapté et plus simple à optimiser. De plus, grâce à son expression sous forme de taux et à sa relation au nombre de décisions, cet indicateur est bien plus compréhensible par l'humain. Ces avantages rendent, à notre avis, cet indicateur plus pertinent que #Seq(Π). Nous utiliserons donc ϕ_Π par la suite comme indicateur de flexibilité pour l'ordonnancement de groupes. [Ess03] favorise également ce critère.

3.1.5 Exemple

Pour illustrer ces définitions, étudions un exemple.

i	1	2	3	4	5	6	7	8	9
$\Gamma^-(i)$	$\varnothing$	$\{1\}$	$\{2\}$	$\varnothing$	$\{4\}$	$\{5\}$	$\varnothing$	$\{7\}$	$\{8\}$
m_i	M_1	M_2	M_3	M_2	M_3	M_1	M_3	M_1	M_2
p_i	3	3	3	4	3	1	2	2	2

Tab. 3.1: Un problème de *job shop*

Le tableau 3.1 représente un problème de *job shop* composé de trois machines et neufs opérations. La figure 3.1 représente un ordonnancement de groupes réalisable de ce problème. Cet ordonnancement de groupes est constitué de sept groupes : deux groupes de deux opérations et cinq groupes d'une opération. Cet ordonnancement de groupes définit quatre ordonnancements semi-actifs représentés dans la figure 3.2. Sa flexibilité est $\phi_\Pi = 1/3$. On remarquera que ces ordonnancements ont des *makespan* différents : la qualité dans le meilleur des cas est $C_{\max} = 10$ et la qualité dans le pire des cas est $C_{\max} = 17$.

3.1.6 Exécution d'un ordonnancement de groupes

Lors de la phase proactive, un ordonnancement de groupes est généré. Pendant la phase réactive, cet ordonnancement de groupes doit être exécuté. Exécuter un ordonnancement de groupes consiste à choisir une séquence d'opérations parmi celles décrites par l'ordonnancement de groupes.

Chaque machine possède une séquence de groupes à exécuter. Le premier groupe de chaque machine est le groupe en cours d'exécution. L'ensemble des opérations contenues dans ce groupe doit être exécuté. Ces exécutions peuvent être vues comme une succession de décisions : une décision consiste à choisir une opération à exécuter dans le groupe en cours d'exécution. Une fois cette opération choisie, son exécution commence, et elle est enlevée du groupe en cours d'exécution. Lorsque le groupe en cours d'exécution est vide, le groupe suivant devient le

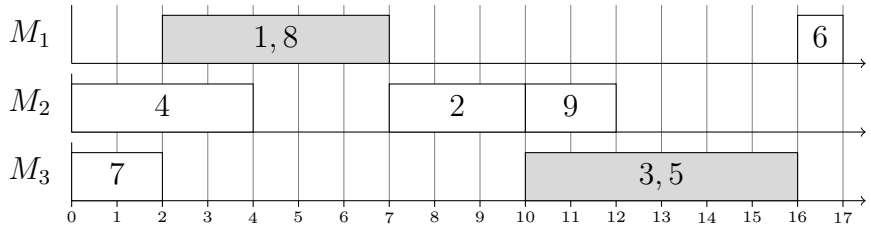


FIG. 3.1: Une séquence de groupes réalisable pour le problème décrit dans le tableau 3.1

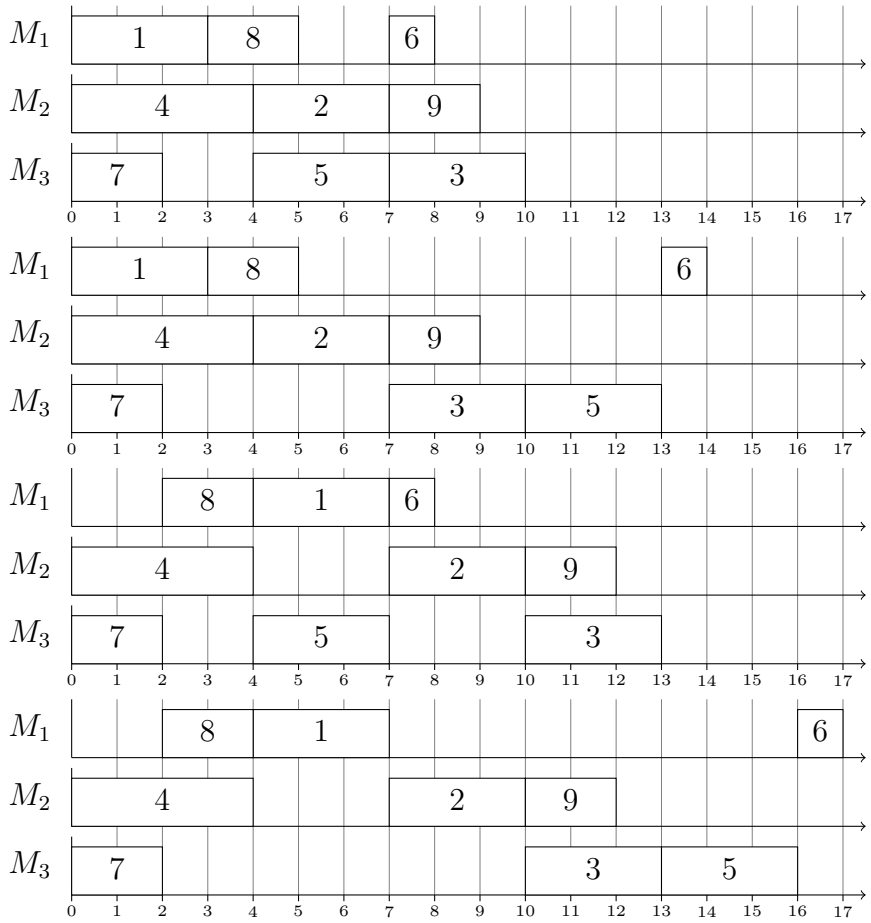


FIG. 3.2: Ordonnancements semi-actifs décrit par l'ordonnancement de groupes représenté dans la figure 3.1

groupe en cours d'exécution, et ainsi de suite jusqu'à ce que toutes les opérations soient exécutées.

Pour illustrer cette notion, nous exécutons l'ordonnancement de groupes décrit sur la figure 3.1. Sur la machine M_2, aucune décision n'est nécessaire : chaque groupe est constitué d'une unique opération. Pour la machine M_1, au début de l'exécution de l'ordonnancement, le premier groupe, en cours d'exécution, possède deux opérations. Il est alors nécessaire de choisir quelle opération sera exécutée en premier. Supposons que l'opération O_1 soit sélectionnée. Par la suite, sur cette machine, il n'y a plus de décision : chaque groupe est composé d'une opération. Sur la machine M_3, après exécution de l'opération O_7, le groupe en cours d'exécution est composé de deux opérations. Il faut donc effectuer un choix. Supposons que l'opération O_5 soit exécutée en premier. Il ne reste alors plus de décision à prendre. L'ordonnancement de groupes est alors exécuté. On remarquera que l'ordonnancement résultat est le premier ordonnancement décrit par la figure 3.2.

3.1.7 Propriétés

L'ordonnancement de groupes possède une propriété intéressante : le calcul de la qualité d'un ordonnancement de groupes dans le pire des cas se fait en temps polynomial (voir [ABE05] pour la description de l'algorithme, ainsi que la section 3.2.1). Ainsi, il est possible de calculer la qualité dans le pire des cas, même pour les ordonnancements de groupes de très grande taille. Par conséquent, cette méthode peut être utilisée pour calculer la qualité dans le pire des cas en temps réel pendant l'exécution de l'ordonnancement. Grâce à cette propriété, il est possible de surveiller la qualité d'un ordonnancement de groupes dans un système d'aide à la décision de manière dynamique.

L'ordonnancement de groupes permet donc de décrire un ensemble d'ordonnancements de manière implicite (c'est-à-dire sans énumérer les ordonnancements) tout en garantissant une certaine qualité. Lors de l'exécution d'un tel ordonnancement, il est ainsi possible de choisir la séquence d'opérations adaptée à l'état réel de l'atelier.

3.1.8 Contraintes prises en compte

L'ordonnancement de groupes peut prendre en compte un grand nombre de contraintes. Ces contraintes furent ajoutées successivement.

[Tho80] utilise l'ordonnancement de groupes pour un *job shop* classique avec comme contrainte le respect des dates de livraisons ($J|r_i, \tilde{d}_i|-$).

Par la suite, [LG89] étend le modèle utilisé. En plus du *job shop*, il inclut les gammes non linéaires, les ensembles de ressources disjonctives et les ressources cumulatives.

[Bil93] augmente le modèle de l'ordonnancement de groupes aux opérations utilisant plusieurs ressources simultanément. Il propose également la prise en compte

des temps de préparation dépendant de la séquence dans le cas mono ressource $(J|r_i, \tilde{d}_i, S_{sd}|-)$.

[Art97] complète le modèle par la prise en compte des temps de préparation pour le cas multiressources. Le problème d'insertion d'une opération dans un ordonnancement de groupes y est également étudié.

Cet ensemble de contraintes est utilisé par le progiciel d'ordonnancement ORDO.

Ce modèle étendu est une grande force pour l'ordonnancement de groupes. Grâce à cela, l'ordonnancement de groupes est une méthode générique, pouvant être utilisé dans un grand nombre d'entreprises différentes.

Par la suite, nous n'utiliserons qu'un sous-ensemble du modèle le plus complet. Nous nous positionnons dans le cas d'un *job shop* classique avec contraintes de précédences entre opérations quelconques en optimisant un critère régulier $(J|r_i, prec|f_{\max})$. Ce modèle est également utilisé dans [ABE05].

3.2 Les principaux algorithmes

Dans cette section, nous présentons les principaux algorithmes relatifs à l'ordonnancement de groupes : Les algorithmes permettant d'évaluer l'ordonnancement de groupes pendant les phases proactive et réactive, ainsi que les algorithmes générant des ordonnancements de groupes. Finalement, nous proposons des améliorations à EBJG, un algorithme générant des ordonnancements de groupes.

3.2.1 Calcul des dates de fin au plus tôt dans le pire des cas

Cet algorithme est décrit dans [ABE05]. Il calcule les dates de fin des opérations au plus tôt dans le pire des cas, c'est-à-dire les dates de fin maximales des opérations présentes dans l'ensemble des ordonnancements semi-actifs décrits par un ordonnancement de groupes. Ces dates sont généralement utilisées pour évaluer un ordonnancement de groupes pour les critères réguliers de type minmax. Elles permettent également d'obtenir une borne supérieure pour les critères réguliers de type minsum. Ces évaluations s'effectuent en temps polynomial.

Le problème est modélisé sous la forme d'un programme dynamique. Il est décomposé en deux parties : le calcul des dates de début au plus tôt dans le pire des cas ($\underline{\tau}_i^{\Pi}$) et le calcul des dates de fin dans le pire des cas ($\underline{C}_i^{\Pi}$).

Les dates de début au plus tôt dans le pire des cas correspondent à la date à laquelle l'opération O_i peut être exécutée dans le pire des cas. Cela correspond à la date à laquelle l'opération O_i peut commencer si elle est exécutée en première position de son groupe, sous l'hypothèse que les opérations précédant O_i finissent le plus tard possible. Les opérations précédant O_i sont les opérations du groupe avant le groupe de O_i ($g^-(i)$) et les prédécesseurs de l'opération O_i ($\Gamma^-(i)$). La valeur $\underline{\tau}_i^{\Pi}$ correspond donc au maximum entre le maximum des dates de fin dans

le pire des cas des opérations précédants O_i et la date de début au plus tôt de l'opération O_i :

$$\underline{\tau}_i^{\Pi} = \max\left(r_i, \max_{j \in \Gamma^-(i) \cup g^-(i)} \underline{C}_i^{\Pi}\right)$$

Pour la date de fin dans le pire des cas d'une opération O_i, nous avons deux possibilités :

- O_i est exécuté en première position dans le groupe ;
- l'opération du groupe $g(i)$ différente de O_i possédant la plus grande date de début au plus tôt dans le pire des cas est exécutée en premier, et O_i est exécutée en dernier.

Ceci s'exprime de la manière suivante :

$$\underline{C}_i^{\Pi} = \max\left(\underline{\tau}_i^{\Pi} + p_i, \max_{j \in g(i), j \neq i} \underline{\tau}_j^{\Pi} + \sum_{j \in g(i)} p_j\right)$$

Cette formule récursive se résout en temps polynomial en utilisant les techniques classiques de programmation dynamique.

Pour calculer la qualité dans le pire des cas pour les objectifs réguliers de type minmax, il suffit d'utiliser directement $\underline{C}_i^{\Pi}$. Par exemple, le *makespan* de l'ordonnancement de groupes Π se calcule

$$\max_i \underline{C}_i^{\Pi}$$

3.2.2 Calcul des dates de début au plus tard dans le pire des cas

Le calcul de la date de début au plus tard de l'opération O_i dans le pire des cas correspond à la date de début de l'opération O_i maximale garantissant qu'il n'existe aucun ordonnancement décrit par la séquence de groupes possédant une opération en retard. Un algorithme calculant cette valeur est présenté dans [Tho80]. Pour rendre cohérent la description de cet algorithme avec le reste, nous le reformulons en utilisant la formulation de l'algorithme des dates de fin au plus tôt dans le pire des cas, décrit à la section précédente.

Ce problème est en tout point similaire au calcul des date de fin au plus tôt dans le pire des cas, sauf que le problème est pris dans l'autre sens : on part des dates de livraison, et on parcourt l'ordonnancement de groupes de la fin jusqu'au début. Ainsi, on calcule les dates de fin au plus tard dans le pire des cas ($\bar{C}_i^{\Pi}$, équivalent de $\underline{\tau}_i^{\Pi}$), et les dates de début au plus tard dans le pire des cas ($\bar{\tau}_i^{\Pi}$, équivalent de $\underline{C}_i^{\Pi}$).

La date de fin au plus tard de l'opération O_i dans le pire des cas ($\bar{C}_i^{\Pi}$) est le minimum entre sa date de livraison d_i et les dates de début au plus tard dans le pire des cas de ses successeurs (les opérations du groupe suivant et les successeurs

de l'opération O_i) :

$$\bar{C}_i^{\Pi} = \min\left(d_i, \min_{j\in\Gamma^+(i)\cup g^+(i)} \bar{\tau}_i^{\Pi}\right)$$

En utilisant le même raisonnement que pour $\underline{C}_i^{\Pi}$, la date de début au plus tard de l'opération O_i dans le pire des cas s'exprime ainsi :

$$\bar{\tau}_i^{\Pi} = \min\left(\bar{C}_i^{\Pi} - p_i, \min_{j\in g(i), j\neq i} \bar{C}_j^{\Pi} - \sum_{j\in g(i)} p_j\right)$$

3.2.3 Marge libre séquentielle

La marge libre séquentielle est une adaptation pour l'ordonnancement de groupes de la marge libre. Elle est présentée dans [Tho80]. Cette marge représente le retard maximum de l'opération garantissant qu'aucune opération ne sera en retard. Elle s'exprime ainsi :

$$\begin{aligned}
\mathrm{m}_{\mathrm{s}}^{\Pi}(i) &= \bar{\tau}_i^{\Pi} - \underline{\tau}_i^{\Pi} \\
&= \min\left(\bar{C}_i^{\Pi} - p_i - \underline{\tau}_i^{\Pi}, \min_{j\in g(i), j\neq i} \bar{C}_j^{\Pi} - \sum_{j\in g(i)} p_j - \underline{\tau}_i^{\Pi}\right) \\
&= \min\left(\mathrm{m}_{\mathrm{sp}}^{\Pi}(i), \mathrm{m}_{\mathrm{sg}}^{\Pi}(i)\right)
\end{aligned}$$

La marge libre séquentielle d'une opération est donc séparée en deux parties :
- $\mathrm{m}_{\mathrm{sp}}^{\Pi}(i)$, relative à la seule opération O_i, appelée sa marge propre ;
- $\mathrm{m}_{\mathrm{sg}}^{\Pi}(i)$, faisant intervenir les autres tâches du groupes, appelée sa marge groupe.

Ces marges possèdent différentes propriétés permettant de vérifier que l'ordonnancement de groupes en cours d'exécution ne peut pas donner d'ordonnancement possédant des opérations en retard (de tels ordonnancements de groupes sont appelés ordonnancements de groupes sans retard). Les opérations appartenant aux groupes en cours d'exécution nous suffisent pour vérifier cette propriété.

Si toutes les marges libres séquentielles (des opérations appartenant aux groupes en cours d'exécution) sont positives ou nulles, alors toutes les permutations d'opérations possibles sur ce groupe donnent des ordonnancements de groupes sans retard. Dans ce cas, l'opérateur peut choisir l'opération à exécuter suivant ses préférences. Pour maximiser les marges, [Tho80] suggère d'exécuter l'opération possédant la plus grande marge groupe.

Si, dans un groupe en cours d'exécution, au moins une marge libre séquentielle est négative, alors toutes les séquences de ce groupe ne donnent pas un ordonnancement de groupes sans retard. Plusieurs cas sont possibles :
- Si la marge propre d'au moins une opération est négative, alors il n'existe aucune séquence de ce groupe donnant un ordonnancement de groupes sans retard.

– Si toutes les marges propres sont positives, alors on ne peut pas savoir s'il existe une séquence permettant d'aboutir à un ordonnancement de groupes sans retards. Exécuter l'opération possédant la plus grande marge groupe semble être un choix judicieux, car il garantit qu'aucune marge ne diminuera (elles peuvent augmenter). Ainsi, il est possible de rendre toutes les marges positives après l'exécution de quelques opérations.

La marge libre séquentielle est donc un outil efficace permettant de surveiller un ordonnancement de groupes en cours d'exécution. Son principal inconvénient est de se focaliser sur le groupe, et donc de ne pas prendre en compte les autres groupes dans la recherche d'une solution. En effet, l'exécution des opérations sur les autres groupes peuvent influencer les dates de début au plus tôt dans le pire des cas, et donc rendre les marges positives.

3.2.4 Génération d'un ordonnancement de groupes

Le calcul d'un ordonnancement de groupes est le cœur de la phase réactive. Son but est de générer un ordonnancement de groupes réalisable, tout en optimisant certains objectifs (qualité dans le pire des cas, flexibilité, *etc.*).

3.2.4.1 Les différents algorithmes

Différents algorithmes existent pour générer des ordonnancements de groupes.

La méthode ORABAID a pour but de générer un ordonnancement réalisable satisfaisant les dates de livraisons dans tous les cas. La méthode de génération de l'ordonnancement de groupes [Bil93, Art97] est divisée en trois parties : une étape de construction d'un ordonnancement de groupes réalisable, une étape cherchant à respecter les dates de disponibilités et une étape d'augmentation de la flexibilité. L'étape de construction est basée sur une règle de priorité, en regroupant au maximum les opérations dans des groupes. L'étape suivante, qui cherche à respecter les dates de livraisons, est un algorithme utilisant la méthode tabou basé sur la suppression et réinsertion d'opérations. Si, à la fin de cette étape, certaines dates de livraisons ne sont pas respectées, chaque date de livraison est actualisée à la date de fin dans le pire des cas de l'opération retardataire correspondante. La dernière étape d'augmentation de la flexibilité effectue un maximum de fusions de groupes de manière gloutonne tout en respectant les dates de livraisons.

Une autre méthode est proposée par [Ess03]. Après une étude de problèmes à deux machines, Esswein présente trois heuristiques pour minimiser le nombre de groupes en donnant comme contrainte une certaine qualité dans le pire des cas (Cette méthode de résolution est une approche ε-contrainte du problème biobjectif flexibilité qualité dans le pire des cas). Ces algorithmes commencent avec une solution du problème d'ordonnancement classique, et retourne un ordonnancement de groupes contenant cet ordonnancement initial. Le premier algorithme, nommé OGAJ, est un algorithme de type *shifting bottleneck*, basé sur une relaxation du

problème à une machine. Le deuxième, AG, est un algorithme génétique. Le troisième, EBJG, est un algorithme glouton. OGAJ a comme principal inconvénient de ne pas respecter strictement la contrainte de qualité, et donne la plupart du temps un ordonnancement de groupes possédant une moins bonne qualité que celle demandée. Une autre particularité de cette méthode est qu'elle ne répartit pas la flexibilité équitablement sur les machines : les premières machines regroupées possèdent en général plus de flexibilité que les autres. OGAJ et EBJG sont aussi performants l'un que l'autre sur les petites instances, mais EBJG donne des ordonnancements de groupes plus flexibles que AG sur les instances plus grandes.

Dans la suite de ce travail, nous utilisons une version modifiée de l'algorithme EBJG pour générer nos ordonnancements de groupes. EBJG est décrit à la section suivante, puis nous présentons nos améliorations.

3.2.4.2 EBJG

EBJG est un algorithme constructif effectuant un regroupement à chaque itération. Un regroupement consiste à fusionner deux groupes consécutifs sur une machine. L'algorithme démarre avec un ordonnancement de groupes initial (généralement, un ordonnancement de groupes avec une opération par groupe, en provenance d'un algorithme classique d'ordonnancement). Des regroupements sont effectués jusqu'à ce qu'aucun regroupement ne soit possible sans obtenir un ordonnancement de groupes possédant une qualité dans le pire des cas moins bonne qu'une borne donnée en paramètre de l'algorithme. À chaque itération, un regroupement est sélectionné de la façon suivante :

- les regroupements donnant un ordonnancement de groupes réalisable sont sélectionnés ;
- parmi ces regroupements, ceux possédant la meilleure qualité dans le pire des cas sont sélectionnés ;
- parmi ces regroupements, on sélectionne le regroupement qui a le moins d'opérations dans son groupe en contenant le plus.

Les différents regroupements possibles sont stockés tout au long de l'exécution de l'algorithme. En effet, si un regroupement donne lieu à un ordonnancement de groupes non réalisable, ou de qualité trop faible, ce regroupement ne sera pas sélectionné par la suite, car il conservera toujours ces défauts. Ainsi, le nombre de regroupements à évaluer tout au long de la procédure est fortement diminué.

La complexité d'EBJG est évalué à $O(n^4)$, n le nombre d'opérations.

3.2.4.3 Améliorations apportées à EBJG

Pour notre implémentation d'EBJG, nous avons modifié l'algorithme original. Ces deux implémentations donnent des ordonnancements de groupes de flexibilité à peu près identiques. Nous avons effectué trois modifications : une simplification d'implémentation de notre algorithme, une amélioration diminuant le temps

d'exécution et une modification permettant d'obtenir un ensemble de solutions non dominées plutôt qu'une solution unique.

La première modification consiste à changer la dernière règle de sélection du regroupement : au lieu de sélectionner le regroupement « qui a le moins d'opérations dans son groupe en contenant le plus », nous choisissons le regroupement dont le nouveau groupe possède le minimum d'opérations. Cette modification rend le code plus simple et plus lisible.

D'après nos expérimentations, cette modification est sans impact significatif sur la solution finale. Les premiers regroupements sont des regroupements de deux groupes possédant chacun une opération. Sur ces cas, les deux algorithmes sont identiques. D'après nos analyses de l'exécution de l'algorithme, lorsque tous les regroupements possibles de groupes d'une opération ont été effectués, c'est généralement la condition de qualité qui devient discriminante. De ce fait, les deux algorithmes effectuent des regroupements similaires.

La plus grosse différence entre les deux implémentations semble être le choix du regroupement au début : en effet, au début de l'exécution de l'algorithme, il y a beaucoup de regroupements satisfaisant les contraintes imposées, et il n'est pas précisé quel choix effectuer en cas d'égalités. Les deux implémentations ne font donc pas les mêmes choix à ce moment, ce qui peut expliquer les légères différences de résultats.

La deuxième modification est une optimisation permettant de diminuer le nombre d'évaluations de regroupement. Elle diminue le temps d'exécution de l'algorithme. Pour éviter des évaluations de regroupements, nous utilisons les évaluations des regroupements des itérations précédentes comme des bornes inférieures. En effet, si un regroupement donne un ordonnancement de groupes d'une certaine qualité dans le pire des cas à un instant donné, le même regroupement effectué dans les itérations suivantes sera de la même qualité ou de qualité moins bonne, car ce dernier ordonnancement de groupes inclut le précédent. Ainsi, il n'est nécessaire d'évaluer que les regroupements possédant une borne inférieure plus petite ou égale à notre regroupement possédant la meilleure qualité.

On remarquera que, si la qualité recherchée est la qualité de l'ordonnancement de groupes initial, cette amélioration n'apporte aucun changement. Par contre, si la qualité dans le pire des cas demandée est inférieure à la qualité de l'ordonnancement de groupes de départ, le nombre d'évaluations est diminué.

Pour mesurer l'impact de cette modification, nous comparons les temps d'exécutions de EBJG original [Ess03] et de notre implémentation en fonction du nombre de regroupements (le temps d'exécution en secondes est pondéré par la fréquence du processeur de façon à avoir un temps correspondant à une vitesse de processeur de 600MHz). L'instance considéré est la32 de [Law84]. Les résultats sont représentés sur la figure 3.3.

Le premier point à 102 regroupements correspond au cas où la qualité de l'or-

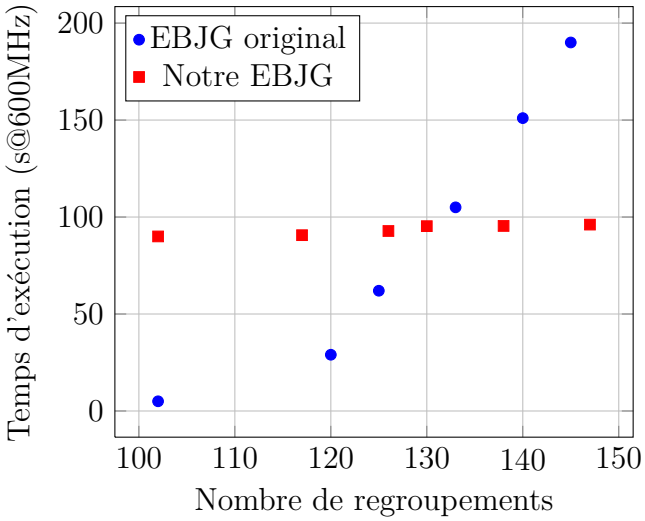


FIG. 3.3: Comparaison des temps d'exécutions entre EBJG original et notre implémentation sur l'instance la32

donnancement de groupes final est la même que la qualité de l'ordonnancement initial. Théoriquement, les mêmes calculs sont effectués par les deux méthodes. On remarque que l'algorithme original est bien plus efficace que notre implémentation, sûrement à cause du langage de programmation, ou de l'optimisation de l'implémentation de l'évaluation. Par contre, d'un point de vue global, on se rend compte que notre implémentation ne demande presque pas de calcul supplémentaire pour obtenir des ordonnancements plus flexibles contrairement à l'implémentation d'origine. En effet, à partir du moment où chaque regroupement possible dégrade la qualité de l'ordonnancement d'origine, notre algorithme n'évalue que très peu de candidats grâce à l'utilisation de la borne inférieure.

La troisième modification utilise EBJG pour obtenir un ensemble de solutions non-dominées. À chaque itération, EBJG possède un ordonnancement de groupes réalisable. De plus, comme les regroupements sont réalisés de manière à garder la qualité dans le pire des cas minimale, l'algorithme va rencontrer un certain nombre de solutions non dominées tout au long de son exécution. Nous pouvons alors garder ces solutions durant l'exécution de l'algorithme pour obtenir un ensemble de solutions non dominées. L'exécution de cet algorithme est aussi couteux que d'exécuter EBJG sans la contrainte epsilon. La figure 3.4 montre l'ensemble des solutions proposées par cette variante d'EBJG pour le problème la40 avec comme objectif le *makespan*. Pour cet exemple, 107 solutions non dominées sont proposées par l'algorithme.

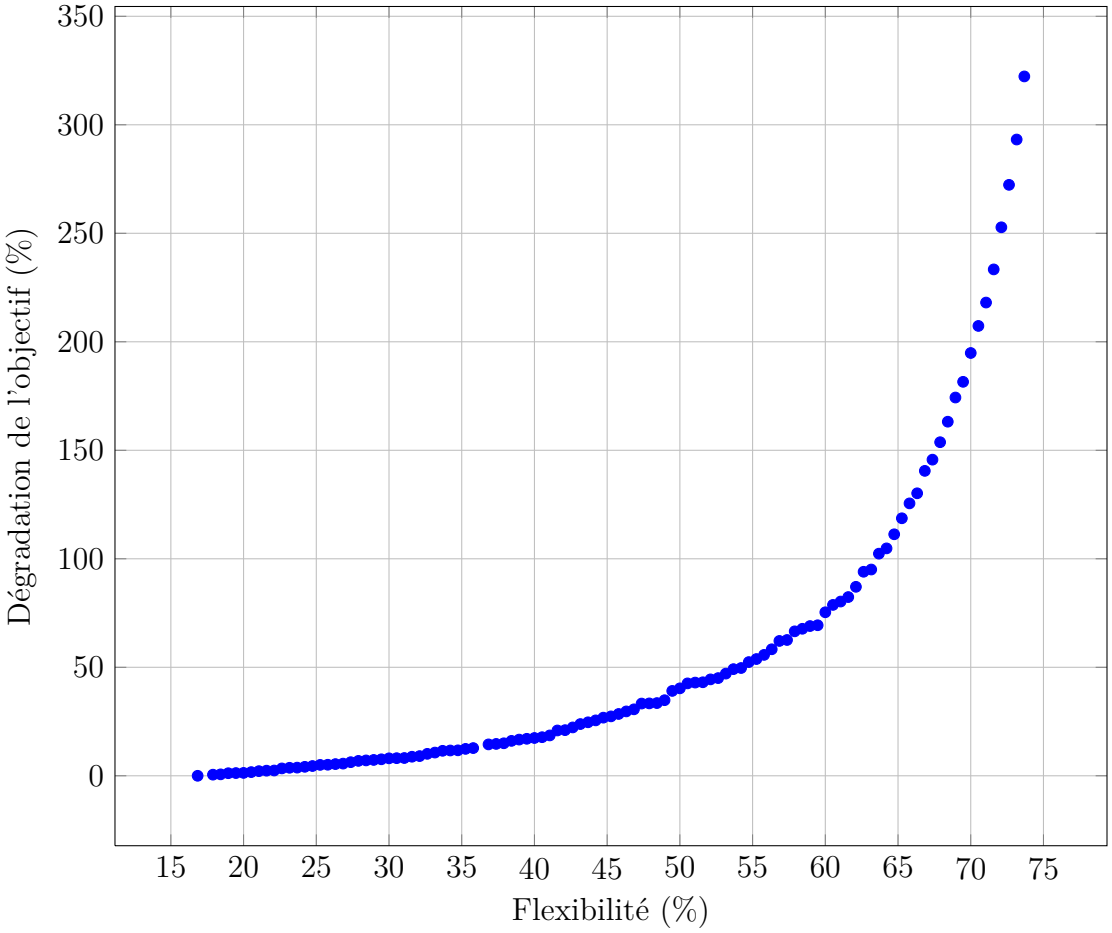


FIG. 3.4: Solutions non dominées proposées par notre implémentation d'EBJG pour le problème la40 avec comme objectif le *makespan*

3.3 Robustesse de l'ordonnancement de groupes

Le but de l'ordonnancement de groupes est de proposer de la flexibilité séquentielle durant la phase réactive de l'ordonnancement, dans un but de robustesse. Deux études ont été réalisées pour vérifier que cette flexibilité permet à l'ordonnacnement de groupes d'être une méthode robuste : [WBS99, Ess03]. La suite de cette section résume ces études de robustesse.

3.3.1 Étude de Wu *et al*

Constatant que l'ordonnancement d'atelier est soumis à incertitudes, et que la connaissance exacte de l'environnement de production n'est disponible qu'au moment de son exécution, [WBS99] propose de retarder un certain nombre de décisions au moment de l'exécution de l'ordonnancement. Étudiant le problème de *job shop* avec minimisation de la somme pondérée des retards ($\sum w_i T_i$), cet article propose une méthode appelée *Preprocess First Schedule Later* (PFSL). [ABE05] montre que cette méthode est une reformulation de l'ordonnancement de groupes. La phase réactive utilise, pour effectuer les décisions restantes, des règles de priorités.

Pour évaluer PFSL, les auteurs la comparent à une méthode statique et à une méthode réactive utilisant la règle de priorité ATC. Les incertitudes étudiées sont la variation des durées opératoires. PFSL montre de meilleurs performances que les deux autres méthodes lorsque les perturbations sont petites à modérées.

La phase prédictive de PFSL utilisant des algorithmes exponentielles, son utilisation sur de grands problèmes de *job shop* ne semble pas réaliste.

Cette étude nous montre que l'ordonnancement de groupes absorbe les incertitudes pour des problèmes peu à moyennement perturbés. Nous pouvons également remarquer que cet article n'utilise pas la marge libre séquentielle, et donc que l'ordonnancement de groupes peut être robuste sans utiliser cet indicateur.

3.3.2 Étude d'Esswein

[Ess03] étudie la robustesse de l'ordonnancement de groupes pour les problèmes à une machine ($1|r_i, d_i|f$) en prenant en compte différents objectifs : $C_{\max}$, $L_{\max}$, $T_{\max}$ et $\sum T_i$. L'étude compare l'ordonnancement de groupes avec un ordonnancement statique. L'ordonnancement de groupes utilise la marge libre séquentielle lors de la phase réactive. L'ordonnancement statique est calculé grâce à la règle de priorité EDD. Durant l'exécution de l'ordonnancement statique, les opérations sont exécutées strictement dans l'ordre trouvé par l'ordonnancement statique (la flexibilité temporelle naturelle est utilisée).

Différentes perturbations sont étudiées : les augmentations des dates de disponibilités des opérations, les augmentations des temps d'exécution et les diminutions des dates de livraisons. Ces différentes perturbations peuvent être plus ou moins fortes.

Les résultats montrent que l'ordonnancement de groupes donne un ordonnancement qui est toujours meilleur que la méthode statique, quel que soit l'objectif considéré. L'ajout de flexibilité au détriment de la qualité dans le pire des cas semble globalement profitable à la qualité finale de l'ordonnancement.

Esswein rappelle à la fin de son étude que, bien que les résultats sont prometteurs, l'environnement étudié est simple (une machine) et les perturbations sont limitées. Ces résultats donnent donc des premiers éléments de réponse à la question de la robustesse de l'ordonnancement de groupes.

3.4 L'ordonnancement de groupes sur une chaîne de production flexible

L'atelier inter-établissement de productique de Nantes met à notre disposition une chaîne de production flexible. Pour valider notre choix de l'ordonnancement de groupes, nous avons adapté l'ordonnancement de groupes à cette chaîne de production.

Cette chaîne de production flexible repose sur un système de transport automatique spécifique, et plutôt complexe à modéliser de manière exacte pour l'ordonnancement de groupes. Une bonne approximation serait d'utiliser des temps de transport simple, c'est-à-dire un temps minimal entre la fin d'une opération et le début de la suivante sur un produit donné.

Comme nous avons à notre disposition une implémentation de l'ordonnancement de groupes sans temps de transport, nous décidons d'utiliser ce modèle pour cette chaîne. Connaissant la différence entre le modèle et la réalité, nous pouvons évaluer le potentiel de robustesse de l'ordonnancement de groupes, puisque la méthode devra réagir aux temps de transports, non prévu durant la phase d'optimisation. Ce travail est également décrit dans [PCM07].

3.4.1 Adaptation de l'ordonnancement de groupes pour une chaîne de production flexible

Pour effectuer des expérimentations sur de grand horizons, nous utilisons une émulation [Bre00] d'une chaîne de production flexible. Le système (voir la figure 3.5 pour une photographie du système réel) peut être modélisé comme un *job shop* à 6 machines avec des transferts automatisés (voir la figure 3.6). Chaque machine possède une file d'attente de capacité finie. Quatre machines ne peuvent être utilisées qu'en FIFO alors que pour les deux dernières, l'ordre des opérations peut être choisi. Les transferts automatisés entre les machines sont effectués par 42 transporteurs unidirectionnels, stockés dans un magasin au centre de la chaîne pendant les périodes d'inactivité. Il ne peuvent prendre en charge qu'un produit à la fois. Ils possèdent un système d'enregistrement leur permettant de transporter les informations du produit avec une mise-à-jour en temps réel (référence, gamme opératoire,

état, *etc.*).

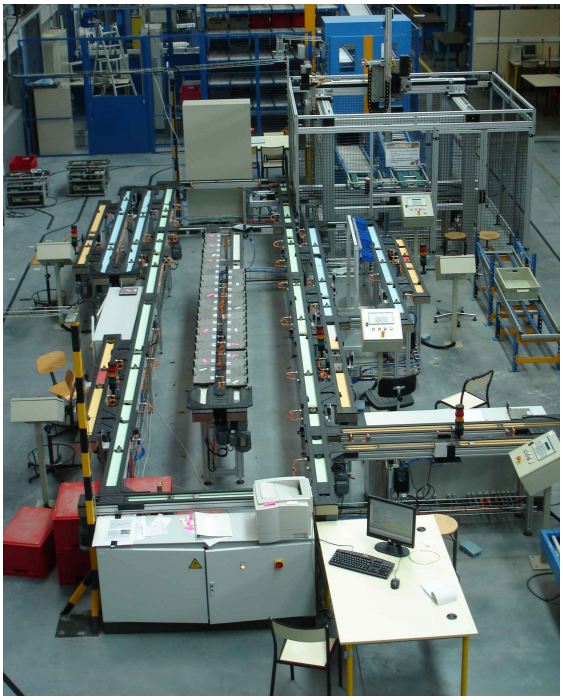


FIG. 3.5: Photographie de la chaîne de production

Le système de transfert est construit autour d'une boucle centrale. Chaque transporteur se déplace sur la boucle jusqu'à ce qu'il puisse entrer dans la file d'une machine. Plus précisément, quand un transporteur arrive à l'entrée d'une machine, une décision, basée sur le nombre de produits dans la file d'attente de la machine, est prise pour autoriser le transporteur à entrer dans la file d'attente de la machine. Si le transporteur n'est pas autorisé à entrer dans la file d'attente, il continue son chemin sur la boucle centrale jusqu'à la prochaine machine.

Le modèle d'émulation nous permet de simuler différents scénarios, caractérisés par deux paramètres :

- la vitesse des transporteurs (de zéro à l'infini) ;
- la règle de décision permettant d'entrer dans une file d'attente.

En fonction de la règle de décision utilisée, trois modes d'exécution sont définis :

- Un ordonnancement prédictif basé sur une séquence d'opérations : pour exécuter l'ordonnancement sur la chaîne, la séquence des opérations sur chaque machine doit suivre la séquence prédéfinie donnée au début de l'exécution. La règle de décision peut être formulée ainsi : « le produit entre dans la file d'attente de la machine si la file n'est pas pleine et si sa prochaine opération à réaliser correspond à la prochaine opération à effectuer sur le produit ». Cette règle de décision est appelé la *règle de décision par séquence d'opérations*.
- Un ordonnancement proactif-réactif utilisant l'ordonnancement de groupes. La règle de décision est alors « le produit entre dans la file d'attente si

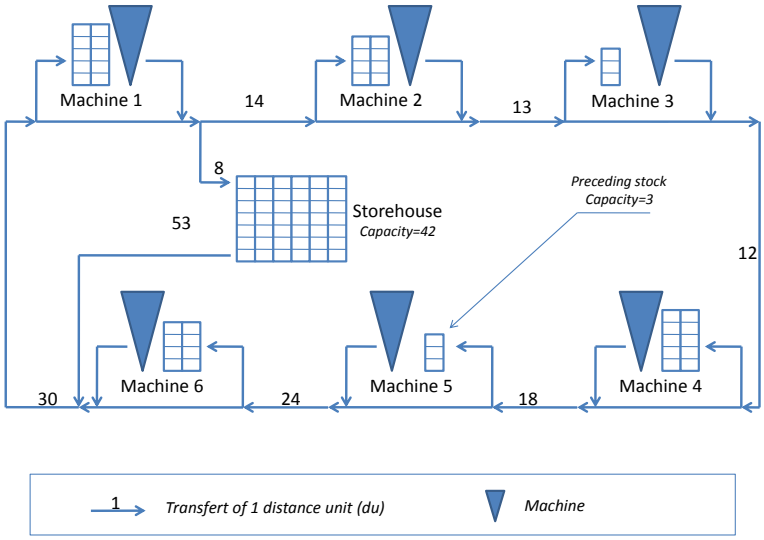


FIG. 3.6: Modélisation sous forme de *job shop* du système de production étudié

la file n'est pas pleine et que sa prochaine opération correspond à une des opérations du groupe en cours d'exécution sur la machine ». Cette règle est appelée la *règle de décision par groupes d'opérations*.

– Un ordonnancement dynamique, reposant sur le concept de premier arrivé, premier servi. La règle est alors « le produit entre dans la file d'attente si la file n'est pas pleine et si la machine est capable d'effectuer la prochaine opération du produit ». Cette règle est appelée la *règle de décision dynamique*.

3.4.2 Expérimentations

Pour vérifier l'efficacité de l'ordonnancement de groupes pour ce système de production réel, nous avons effectué différentes mesures à partir un problème d'ordonnancement connu de la littérature nommé la14 de [Law84].

Ce problème a un double intérêt : il est utilisable sur notre chaîne de production et il est possible d'y ajouter beaucoup de flexibilité. Par contre, cette instance du problème de *job shop* ne prend pas en compte les temps de transport présents sur la chaîne de production. Ces temps de transport seront les incertitudes de notre problème. L'objectif utilisé est le *makespan*.

Nous comparons quatre modes d'exécution différents de l'instance la14 :

– Un ordonnancement obtenu avec une méthode prédictive. La séquence d'opération est une solution optimale au problème sans temps de transport. La qualité de cette solution est $C_{\max} = 1292$. Cet ordonnancement utilise la *règle de décision par séquence d'opérations*. Il sera appelé OSS pour *operation sequence schedule*.

– Un ordonnancement de groupes dont la qualité dans le meilleur des cas est la

même que dans le pire des cas quand il n'y a pas de temps de transport. Dans ce contexte, la qualité de tous les ordonnancements décrits est $C_{\max} = 1292$. Cet ordonnancement utilise la *la règle de décision par groupes d'opérations*. Il sera appelé OGSS pour *optimal group sequence schedule*.
- Un autre ordonnancement de groupes dont la qualité dans le pire des cas sans temps de transport est dégradée. Dans ce contexte, la qualité dans le pire des cas est $C_{\max} = 1382$ tandis que la qualité dans le meilleur des cas est $C_{\max} = 1292$. Cet ordonnancement de groupes possède plus de flexibilité que OGSS. Il utilise également *la règle de décision par groupes d'opérations*. Il sera appelé DGSS pour *degraded group sequence schedule*.
- Un ordonnancement dynamique obtenu grâce à la *règle de décision dynamique*. Cet ordonnancement sera appelé DS pour *dynamic schedule*.

Pour générer les ordonnancements de groupes, l'algorithme EBJG amélioré (voir la section 3.2.4.3) est utilisé. La solution initiale utilisée par EBJG ainsi que l'ordonnancement prédictif sont obtenus grâce à la méthode exacte proposée par [BJS94].

Le rapport entre le temps d'exécution moyen d'une opération et le temps de transport moyen entre deux opérations est fonction de la vitesse des transporteurs. Dans le problème la14, le temps d'exécution moyen est de 50 ut (unités de temps). Dans notre chaîne de production, la distance moyenne entre deux opérations est d'environ 55 ud (unités de distance). Ainsi, si la vitesse des transporteurs est plus grande que 1 ud/ut, le temps d'exécution d'une opération est plus grand que le temps moyen de transport entre deux opérations. Plus la vitesse des transporteurs est importante, plus les temps d'exécution des opérations sont importants par rapport aux temps de transport. Autour d'une vitesse de 1 ud/ut, les temps d'exécution des opération sont du même ordre que les temps de transport entre deux opérations.

Nous avons donc réalisé différentes simulations pour mesurer la qualité des quatre ordonnancements en fonction de la vitesse des transporteurs (VT). Les résultats numériques sont exposés sur la tableau 3.2 tandis que la figure 3.7 montre une représentation de leur évolution.

3.4.3 Discussion

Trois différentes zones ressortent de la figure 3.7 :
- VT > 1,4 ud/ut ;
- 0,7 ud/ut < VT < 1,4 ud/ut où les courbes se croisent ;
- VT < 0,7 ud/ut.

3.4.3.1 VT > 1,4 ud/ut

Dans ce cas, en moyenne, les temps d'exécutions des opérations sont plus grands que les temps de transport entre deux opérations. OSS, OGSS et DGSS sont stables et tendent à vitesse infinie vers le *makespan* optimal de la14. Ce résultat est normal

Vitesse (ud/ut)	OSS	OGSS	DGSS	DS
0.1	13851	10482	10489	5617
0.2	7582	5101	5093	3255
0.3	4508	3370	3370	2446
0.4	3613	2535	2563	2000
0.5	2842	2204	2155	1820
0.6	2416	2050	1973	1742
0.7	2228	1840	1869	1570
0.8	1930	1734	1671	1632
0.9	1885	1633	1592	1706
1.0	1708	1415	1523	1684
1.1	1664	1550	1477	1545
1.2	1596	1502	1425	1651
1.3	1535	1433	1376	1576
1.4	1481	1366	1366	1654
1.5	1445	1356	1356	1401
1.6	1419	1352	1352	1491
1.7	1448	1349	1349	1549
1.8	1387	1346	1346	1436
1.9	1376	1343	1343	1592
2.0	1375	1340	1340	1465
5.0	1311	1311	1311	1374
10.0	1301	1301	1301	1301
∞	1292	1292	1292	-

TAB. 3.2: *Makespan* des différents ordonnancements en fonction de la vitesse des transporteurs

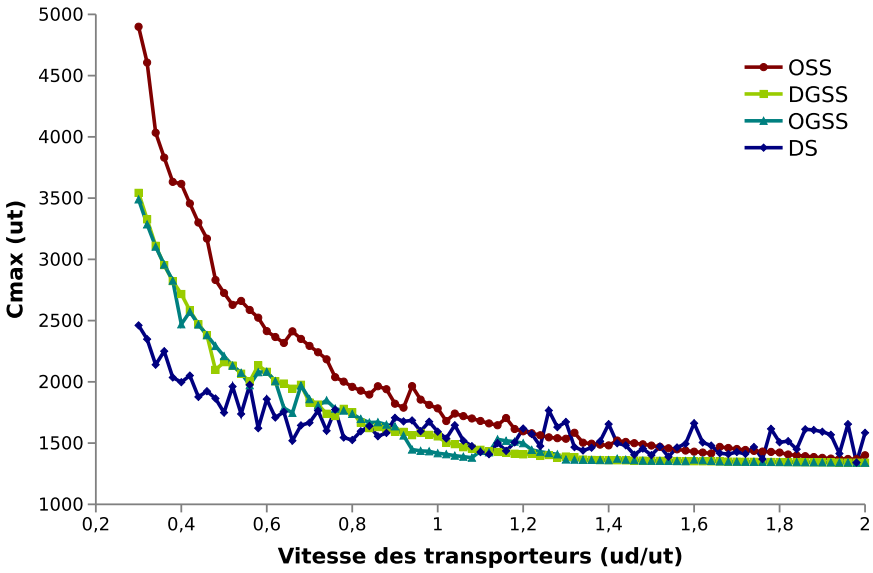


FIG. 3.7: *Makespan* des différents ordonnancements en fonction de la vitesse des transporteurs

pour OSS et OGSS, mais ce n'était pas prévisible pour DGSS : l'ordonnancement sur la chaîne correspond à un des meilleurs des cas représenté par DGSS. Par contre, la qualité de DS est globalement faible et instable en comparaison avec les autres. En effet, la règle de décision, choisie à cause de la configuration et des contraintes techniques du système, n'est pas très efficace pour le *makespan*. De plus, ses performances erratiques sont dûes à la myopie de la règle.

Lorsque la vitesse des transporteurs diminue, les temps de transports deviennent plus importants et OSS obtient une moins bonne qualité que OGSS et DGSS. La flexibilité présente dans ces deux derniers permet bien d'obtenir une meilleure robustesse dans ce cas.

3.4.3.2 0,7 ud/ut < VT < 1,4 ud/ut

Dans ce cas, les temps de transport et les temps d'exécution sont du même ordre de grandeur. Dans cette situation, OSS obtient la pire qualité. Nous pouvons donc dire que la séquence d'opérations n'est plus adaptée au problème, même en comparaison à DS. La qualité de DS reste erratique mais donne des performances correctes. Enfin, OGSS et DGSS donnent les meilleures performances. Ce résultat est encourageant : pour cet exemple, l'ordonnancement de groupes est suffisamment robuste pour absorber les temps de transport de durée comparable aux temps d'exécution. Les qualités de OGSS et DGSS sont comparables, la différence de flexibilité n'est pas suffisante pour être significative.

3.4.3.3 VT < 0,7 ud/ut

Finalement, lorsque les temps de transports sont plus grands que les temps d'exécution, l'optimisation réalisée dans OSS, OGSS et DGSS n'a plus de sens, puisque les temps d'exécution sont insignifiants par rapport aux temps de transport. Sans surprise, DS obtient les meilleurs performances. La flexibilité de OGSS et DGSS limite les mauvaises performances alors que la qualité de OSS est vraiment mauvaise.

3.4.3.4 Analyse globale

Tout d'abord, dans ces expérimentations, l'ordonnancement de groupes est toujours meilleur que l'ordonnancement prédictif. Ce résultat est encourageant pour la robustesse de la méthode : la flexibilité ajoutée permet une certaine robustesse sans perte de qualité par rapport à un ordonnancement prédictif.

Ensuite, l'ordonnancement de groupes montre de meilleurs résultats que l'approche dynamique, tant que les temps de transport ne sont pas plus importants que les temps d'exécution. Tant que le modèle n'est pas trop éloigné de la réalité, l'ordonnancement de groupes offre donc des avantages par rapport à l'ordonnancement dynamique et l'ordonnancement prédictif. Bien sur, quand le modèle devient trop éloigné de la réalité, DS devient bien meilleur.

Ainsi, l'ordonnancement de groupes est une méthode prometteuse qui permet d'obtenir de bonne performances, même avec des temps de transport non modélisés, tant que les temps d'exécutions sont les informations les plus importantes. Cette étude montre que l'ordonnancement de groupes peut être robuste.

3.5 Robustesse et coopération homme-machine

Les études décrites aux sections 3.3 et 3.4 semblent montrer que l'ordonnancement de groupes est une méthode robuste d'ordonnancement lorsque les incertitudes ne sont pas trop importantes. Cette robustesse est obtenue grâce à la présence de flexibilité séquentielle proposée par la méthode.

Mais la présence de flexibilité séquentielle n'est pas suffisante, encore faut-il savoir l'utiliser. En effet, un ordonnancement de groupes pour lequel on choisirait une séquence prédéterminée à l'avance ne serait rien de plus qu'une méthode statique classique. C'est donc l'utilisation de la flexibilité en fonction de l'état de l'atelier qui est la clef de la robustesse, et non la flexibilité en elle-même.

La marge libre séquentielle permet d'utiliser cette flexibilité avec un double objectif de respect des dates de livraison et de maximisation de la flexibilité temporelle. La méthode ORABAID permet d'utiliser cette marge séquentielle en collaboration avec un opérateur humain, mais cette méthode peut être améliorée pour favoriser la coopération avec l'humain.

En effet, cet opérateur humain peut apporter beaucoup à l'utilisation de la flexibilité proposée par l'ordonnancement de groupes. Il possède notamment sa propre vision de l'atelier, différente de celle de la machine. De plus, sa capacité à négocier et à dialoguer avec les différents niveaux de l'entreprise lui permettent de faire des choix plus judicieux, que le machine ne pourrait pas connaître.

C'est pourquoi, dans la partie suivante, nous étudions une nouvelle méthode permettant d'améliorer cette coopération homme-machine, pour favoriser l'utilisation de la flexibilité de l'ordonnancement de groupes, mais également pour placer l'humain au centre du système d'ordonnancement.

Deuxième partie

Coopération homme-machine pour la mise en œuvre d'un ordonnancement de groupes

Chapitre 4

Coopération homme-machine pour l'ordonnancement

Dans ce chapitre, nous abordons les notions générales de coopération homme-machine que nous estimons nécessaire pour notre étude. Dans un premier temps, nous abordons des notions générales. Dans un deuxième temps, nous étudierons des études sur les systèmes homme-machine spécifiques à l'ordonnancement.

4.1 Notions sur la coopération homme-machine

4.1.1 L'outil et la prothèse

Les systèmes experts sont généralement des systèmes homme-machine dont l'objectif est de placer l'expertise dans la machine. L'interface entre le monde réel et la machine est réalisée par un humain, qui fournit à la machine des données, et qui effectue les instructions demandées par la machine. La machine utilise alors ces données pour donner son diagnostic de la situation. Dans ce cas, l'humain n'est utilisé que comme interface avec le monde, et n'a plus aucune fonction de diagnostic.

On dit qu'un tel système homme-machine utilise le paradigme de la prothèse, c'est-à-dire qu'une partie de la fonction de l'humain est remplacé par une machine, tout comme une prothèse remplace une partie du corps (que nous nommerons membre). Ce paradigme interdit l'usage de la redondance entre l'humain et la machine. Si la prothèse est meilleure en tout point au membre qu'elle remplace, alors elle donnera sûrement un système homme-machine efficace. Mais une prothèse en tout point meilleure est peu probable : dans le cas général, la prothèse sera meilleure que le membre original sur certains points, mais pas sur tous. Or, dans le cas des systèmes homme-machine, l'humain est toujours disponible, et une partie de son savoir est donc perdu. De plus, la redondance, interdit par le paradigme de la prothèse, est souvent source de fiabilité dans les systèmes homme-machine.

Délester la fonction de diagnostic de l'humain peut avoir d'autres répercussions

sur son travail. Il est en effet possible qu'il devienne moins attentif à certains données utilisées par d'autres activités. Modifier une partie du système peut donc avoir des effets inattendus indirectement liés à la prothèse. La modification d'une partie du système homme-machine modifie entièrement le système homme-machine, et pas uniquement la partie modifiée.

C'est pourquoi le paradigme de l'outil semble être bien plus pertinent que celui de la prothèse : un outil est un prolongement du corps, non un remplaçant. Un système homme-machine utilisant le paradigme de l'outil n'utilise pas la machine pour remplacer une partie de l'humain ; mais tente d'être une extension de l'humain, lui facilitant son travail. La machine peut par exemple faire des suggestions, aider l'humain à accéder à des informations, ou effectuer des tâches répétitives complexe pour l'humain. Par contre, elle ne doit pas avoir le dernier mot. Ainsi, le système homme-machine est plus efficace, car il combine les avantages de l'humain et de la machine. On peut alors parler de coopération homme-machine.

4.1.2 Utilisation active et passive

[RBW88] étudie le diagnostic de panne par des techniciens sur un équipement électromécanique avec l'aide d'un système expert. Ce système expert utilise le paradigme de la prothèse. Le but est de diagnostiquer une panne sur un équipement récent. Les techniciens possèdent différents degrés de connaissance de cet équipement et d'équipements plus anciens.

L'efficacité n'est pas dépendante des connaissances des techniciens, mais de leur utilisation de la machine. Les techniciens utilisant la machine comme elle a été conçue, c'est-à-dire en suivant à la lettre ses instructions, trouvent peu souvent le bon diagnostic. Par contre, certains techniciens détournent l'utilisation de la machine : exécutant un diagnostic en parallèle, ils filtrent les données à fournir à la machine et évitent ainsi de l'engager sur de mauvaises pistes. Cette utilisation de la machine produit un système homme-machine beaucoup plus efficace.

Ce qui différencie ces deux groupes de personnes est leur méthode d'interaction avec la machine : ceux qui l'utilisent de manière passive, et ceux qui l'utilisent de manière active. L'activité de l'utilisateur favorise donc l'efficacité du système homme-machine.

Favoriser l'activité permet donc de favoriser la coopération homme-machine. Cette coopération permet de mieux utiliser les avantages de chacun des protagonistes, et rend le système homme-machine plus efficace.

4.1.3 Influences des systèmes d'aide à la décision sur l'humain

L'utilisation d'une machine peut avoir des conséquences indésirables sur l'utilisateur. [GPS97] montre l'influence d'un système d'aide à la décision sur l'apprentissage de l'utilisateur. Deux populations sont comparées : la première utilisant un

système d'aide à la décision efficace dans la plupart des cas, mais qui donne parfois des résultats erronés, et la seconde n'utilisant pas de système d'aide à la décision. L'étude montre que globalement, la première population obtient de meilleurs résultats que la seconde. Par contre, pour les cas où le système d'aide à la décision donne des résultats erronés, la seconde population obtient de bien meilleurs résultats. Cette étude nous montre donc que les utilisateurs d'un système d'aide à la décision apprennent beaucoup moins de leurs expériences que des personnes sans expérience.

Contrairement à ce que l'on pourrait supposer, le système d'aide à la décision ne rend pas l'utilisateur plus attentif et efficace dans son travail. L'utilisateur risque de ne pas apprendre son métier, mais d'apprendre à utiliser la machine, ce qui le transforme en interface entre le monde et la machine.

Une solution permettant de contrer ces effets indésirables est de favoriser l'activité de l'humain. Ainsi, l'humain est poussé à prendre ses propres décisions avec l'aide de la machine, et non de lui accorder une confiance aveugle. Il pourrait alors apprendre de ses expériences. De plus, cette implication devrait engendrer d'autres effets bénéfiques. Se sentir utile devrait accroître sa motivation. Cette implication devrait également permettre à l'humain de se sentir plus responsable du système, et d'avoir un point de vue plus critique sur la machine.

4.1.4 Les compromis

Lors de prise de décision, un humain doit faire des compromis. En effet, il n'existe que rarement une solution répondant à tous les critères, et il faut donc faire un choix sur l'importance de chacun d'entre eux pour obtenir une solution.

Pour modéliser cette notion de compromis, la recherche opérationnelle utilise la notion de multiobjectif. Cette notion est décrite à la section 1.1.5.3. Pour réaliser le compromis, deux approches sont alors possibles :

- l'humain détermine l'importance des différents objectifs et les donne comme paramètres du problème à la machine, et celle-ci retourne une solution ;
- la machine génère un ensemble de solutions, et l'humain choisit dans cet ensemble la solution à mettre en œuvre.

Dans tout les cas, l'humain est nécessaire dans le processus de décision.

Ainsi, lorsqu'il faut choisir un compromis, l'humain joue un rôle central dans la prise de décision. La machine peut tout de même jouer un rôle important en facilitant la prise de décision, mais la coopération est alors critique : la machine doit donner des retours pertinents à l'humain pour lui permettre de réaliser son compromis dans les meilleures conditions.

4.2 Coopération homme machine et ordonnancement

4.2.1 Homme, machine et système homme-machine

[San89] rapporte qu'un système homme-machine est en général plus performant qu'un humain ou une machine seule. Des études ont permis de déduire cette affirmation. Ce résultat, en particulier dans le cas de l'ordonnancement, s'explique de différentes manières.

Tout d'abord, un humain réalisant un ordonnancement passe la plus grande partie de son temps à vérifier la satisfaction des contraintes. L'humain n'est donc pas adapté à la réalisation de cette tâche. Par contre, la puissance de calcul de la machine permet à l'humain de vérifier rapidement la satisfaction des contraintes. Ainsi, un système homme-machine aidant l'humain à gérer les contraintes grâce à une interface spécifique est plus efficace que l'humain seul. Pour l'ordonnancement, un système homme-machine est donc plus performant qu'un humain seul. Il nous reste donc à montrer que le système homme-machine (ou l'humain seul) est plus performant que la machine.

Diverses études montrent que l'humain est un meilleur ordonnanceur que la machine. Ces études comparent généralement l'humain à des algorithmes basés sur des règles de priorités plus ou moins complexes vis-à-vis des critères classiques de qualité. Cet argument est beaucoup plus discutable aujourd'hui au vu des méthodes récentes d'optimisation. Il est donc difficile d'affirmer que l'humain est plus performant que la machine.

Par contre, la supériorité d'un système homme-machine sur une machine seule semble tout de même avérée dans un contexte d'ordonnancement industriel. Une qualité propre à l'humain est alors primordiale : sa capacité à négocier le relâchement de contraintes. En effet, une machine va travailler sur un modèle purement théorique, et sera donc limitée par ce modèle. L'humain peut modifier ce problème, pour le rendre plus simple à résoudre. Cette modification du problème peut être réalisée de différentes façons : par négociation avec les autres entités de l'entreprise (reculer une date de livraison, modifier le processus de fabrication d'un produit, sous-traiter une partie de la production, *etc.*) ; ou par relaxation de contraintes du modèle (par exemple commencer une opération avant sa date de début au plus tôt).

L'utilisation d'un système homme-machine semble donc être la meilleure option, d'autant plus que l'humain est souvent indispensable pour d'autres motifs que la performance (sécurité, qualité, communication, *etc.*).

4.2.2 Compréhension du fonctionnement de la machine

[CHss] décrit une expérimentation évaluant l'influence de la compréhension d'un algorithme d'ordonnancement sur le réordonnancement manuel d'une solution

de cet algorithme suite à un aléa. Les résultats montrent qu'il n'est pas possible de mettre en évidence l'effet de la compréhension du résultat sur la performance. Par contre, ils montrent que la compréhension de l'algorithme ne permet pas d'améliorer les performances.

Ainsi, pour faciliter la coopération homme-machine, il n'est pas nécessaire de se restreindre à des algorithmes simples compréhensibles par l'humain. Nous pouvons donc utiliser des algorithmes d'optimisation complexes dans un système homme-machine pour l'ordonnancement d'atelier.

4.2.3 Les différences interindividuelles

[Cegss] étudie les stratégies de huit ordonnanceurs d'anciennetés différentes. Il en ressort que ces ordonnanceurs utilisent dans la résolution du problème d'ordonnancement des stratégies différentes pour résoudre le problème, ces stratégies n'étant pas liées à l'ancienneté des ordonnanceurs. De fortes différences interindividuelles sont donc repérées.

Les différences interindividuelles sont très courantes. Indépendamment de l'ancienneté ou des performances, les humains utilisent différents processus de réflexion. Ces différences ne doivent donc pas être négligées.

Un système de coopération homme-machine ne doit donc pas reposer sur une stratégie spécifique pour mieux s'adapter à l'opérateur. Il est alors nécessaire de laisser à l'ordonnanceur une certaine liberté dans sa tâche.

4.2.4 Niveau de coopération et caractéristiques de la tâche

[vWCH08] propose une méthode d'allocation de fonctions à l'humain et à la machine pour l'ordonnancement. Après une séparation d'un problème d'ordonnancement en un ensemble de sous-tâches, cet article propose un modèle d'allocation de fonctions entre l'homme et la machine.

Les auteurs constatent que l'allocation de fonctions est un sujet bien étudié pour les situations dynamiques, comme l'assistance à la conduite automobile. Comme l'ordonnancement diffère des précédentes études réalisées, une méthode plus adaptée à ce problème est proposée.

Pour catégoriser les différents sous-problèmes, quatre aspects sont proposés :
- le risque, qui peut être défini comme le coût d'une décision sous-optimale ;
- le besoin de connaissance de la situation de la part de l'opérateur, qui est dépendant du niveau d'automatisation de la tâche ;
- le temps de réaction et le temps pour la réparation d'erreur qui permet de donner une échelle temporaire pour la résolution du sous problème ;
- la complexité du problème, qui est plus liée aux interactions entre les sous problèmes qu'à la taille du problème lui-même.

Du point de vue de l'algorithme, la faisabilité technique de l'automatisation du sous problème est étudiée. Cette valeur est supposée continue, allant de la dispo-

nibilité immédiate de l'algorithme à l'impossibilité d'automatisation, en passant par les différents coûts de développement de l'algorithme.

Les caractéristiques du sous problème et de l'algorithme pour ce sous problème déterminent les activités qui doivent être réalisées par l'humain. Mais l'humain possède également sa place dans le processus, et doit donc être pris en compte pour obtenir une coopération homme-machine efficace. Quatre caractéristiques sont prises en compte :

- Le niveau d'implication de l'humain dans le sous problème. Dans certains cas, il est nécessaire que l'humain s'implique dans un sous problème. Par exemple, ne pas être impliqué peut mener l'humain à une perte de compétences. D'autre part, si un sous problème est traité intégralement par une machine, l'humain ne cherchera pas à en améliorer les performances, même s'il en est capable.
- La confiance vis-à-vis des performances de la machine. Si l'humain se considère plus performant que la machine, il ne l'utilisera pas. De plus, la compréhension des variations de performances de la machine permet de favoriser la coopération homme-machine. Par exemple, si l'humain sait que la machine ne donne pas de bons résultats dans certains cas identifiables, l'humain prendra alors le contrôle dans ces cas. Si par contre, l'humain n'arrive pas à savoir quand la machine est performante et quand elle ne l'est pas, l'humain n'aura pas confiance dans les résultats de la machine, et réalisera le travail seul.
- La pertinence des retours de l'algorithme. Si l'humain et la machine partagent un contexte proche, la coopération sera favorisée car l'humain comprendra plus facilement ce qui est proposé par l'algorithme.
- La flexibilité des contraintes. L'humain est capable de modifier les données du problème en utilisant des modes non prévus par le modèle de la machine. La machine doit prévenir l'humain s'il tente de violer une contrainte, mais l'humain doit être capable de violer les contraintes en connaissance de cause.

En fonction des résultats, les auteurs suggèrent différents modes d'interaction homme-machine.

Toutes ces notions sont importantes pour obtenir un système homme-machine efficace. C'est pourquoi les système homme-machine des deux chapitres suivants tentent de les prendre en compte au maximum pour concevoir des systèmes homme-machine les plus efficaces possibles.

Chapitre 5

Coopération pour la phase proactive

La phase proactive de l'ordonnancement de groupes est la partie où l'optimisation a lieu. Bien que l'optimisation soit plutôt du ressort de la machine, il est important que l'humain ait une place centrale pour garantir le respect des objectifs réels de l'atelier.

Dans ce chapitre, nous ne nous intéressons qu'au système homme-machine, et non au fonctionnement interne de la machine. Les approches utilisées par la machine sont décrites à la section 3.2.4.

Tout d'abord, nous présentons le système homme-machine de la phase proactive de l'approche ORABAID, la première approche à utiliser l'ordonnancement de groupes. Ensuite, nous étudions la nouvelle formulation d'Esswein, et proposons une amélioration du système homme-machine.

5.1 L'approche d'ORABAID

La phase proactive proposée par ORABAID est utilisée dans le progiciel ORDO. C'est à notre connaissance la seule phase proactive utilisée dans un environnement de décision réaliste.

5.1.1 Description du système homme-machine

L'approche proposée par ORABAID est centrée sur les dates de disponibilité et les dates de fin des travaux. Ces différentes données proviennent de la planification : les dates de fin correspondent aux dates de livraison au client et les dates de disponibilité sont déduites des dates de livraisons des matières premières, et éventuellement du jalonnement. L'humain peut modifier ces dates avant de donner le problème à la machine. Deux cas peuvent se produire : soit la machine trouve un ordonnancement de groupes satisfaisant les dates de fin des travaux, soit elle ne trouve pas un tel ordonnancement.

Si la machine trouve un ordonnancement de groupes satisfaisant les dates de fin des travaux, elle donne à l'humain l'ordonnancement de groupes à exécuter. La phase proactive est alors terminée.

Si la machine ne trouve pas d'ordonnancement de groupes satisfaisant les dates de fin des travaux, la machine prévient l'humain en lui spécifiant les travaux qui seront en retard, avec les retards correspondants. Deux possibilités s'offrent alors à l'humain :

- Il accepte les retards. Les dates de fin des travaux trouvées par la machine deviennent alors les nouvelles dates de fin dans le problème. Un ordonnancement de groupes correspondant est alors proposé par la machine.
- L'humain modifie les données du problème, particulièrement les dates de disponibilité et les dates de fin des travaux. La machine recommence alors au début avec le nouveau problème.

Une fois que la machine donne un ordonnancement de groupes satisfaisant les dates de fin des travaux (modifiées ou non), la phase proactive d'ORABAID se termine. La phase réactive peut alors commencer.

5.1.2 Analyse du système homme-machine

Dans ce système homme-machine, l'humain donne un problème à la machine, puis par la suite, vérifie la proposition. Ce système homme-machine pour la phase proactive de l'ordonnancement de groupes possède différents avantages et inconvénients.

Ce système homme-machine permet à l'humain de négocier avec les autres instances de l'entreprise pour ajuster les dates de fin des travaux. Cette fonctionnalité permet au système homme-machine d'être fonctionnel : les dates de fin des travaux sont une notion compréhensible par l'humain et la machine. Ainsi, grâce à la compréhension de la solution proposée par la machine, l'humain peut interagir pour obtenir un ordonnancement satisfaisant.

D'un autre côté, le référentiel commun entre l'homme et la machine (le respect des dates de fin des travaux) est très pauvre. Aucune information n'est donnée à l'humain autre que le respect ou non des dates de fin des travaux. Notamment, aucune information sur la flexibilité de l'ordonnancement proposé, alors que cette flexibilité est la force de l'ordonnancement de groupes. La machine n'informe pas non plus l'humain sur l'utilisation de l'atelier. En effet, rien ne permet à l'humain de savoir si l'atelier tournera à pleine charge ou si les machines seront sous exploitées.

Le système homme-machine proposé par ORABAID permet d'obtenir de manière efficace un ordonnancement satisfaisant les dates de fin des travaux. Mais, vu que l'interaction se fait uniquement à ce niveau, les performances du système sont limitées. En utilisant un référentiel commun plus important entre l'homme et la machine, il serait possible d'utiliser plus efficacement l'ordonnancement de groupes.

5.2 Reformulation multiobjectif d'Esswein

[Ess03] propose une nouvelle formulation de la phase proactive. Plutôt que de formuler le problème sous la forme de satisfaction des dates de fin des travaux, un modèle multiobjectif est proposé, introduisant le compromis entre la flexibilité et la qualité d'un ordonnancement de groupes. Ce modèle rend la génération d'ordonnancements de groupes plus flexible.

5.2.1 Le modèle multiobjectif

Pour effectuer une reformulation multiobjectif de la phase proactive, il faut tout d'abord identifier des objectifs pertinents. Esswein propose la flexibilité et la qualité. La qualité est divisée en deux objectifs : la qualité dans le pire des cas et la qualité dans le meilleur des cas.

La flexibilité permet de mesurer la liberté proposée par l'ordonnancement de groupes. Plus la flexibilité est grande, plus l'ordonnancement devrait être robuste. Cet objectif devient d'autant plus nécessaire que les incertitudes sont importantes.

La qualité dans le pire des cas donne la qualité garantie par l'ordonnancement de groupes. Cet objectif est primordial : il permet d'évaluer la qualité minimale de l'ordonnancement. De plus, c'est le critère de qualité utilisé pour l'ordonnancement de groupes dans les études précédentes.

La qualité dans le meilleur des cas apporte une information supplémentaire sur l'ordonnancement de groupes, complémentaire à la qualité dans le pire des cas. Ainsi, en combinant la qualité dans le pire des cas et la qualité dans le meilleur des cas, on obtient les bornes de qualité possibles d'un ordonnancement de groupes.

S'il n'y avait pas de conflit entre les objectifs, il ne serait pas nécessaire de recourir à une telle modélisation. Ce n'est malheureusement pas le cas : la flexibilité et la qualité dans le pire des cas sont deux objectifs conflictuels. Il est alors nécessaire de recourir à une approche multiobjectif considérant au moins la flexibilité et la qualité dans le pire des cas. La qualité dans le meilleur des cas n'est par contre pas un objectif conflictuel vis-à-vis des deux autres objectifs. Elle n'est alors pas prise en compte dans le modèle d'optimisation, mais ne perd pas son intérêt dans la description d'un ordonnancement de groupes.

5.2.2 Compréhension de la mesure de la flexibilité

La section 3.1.4 propose deux mesures de flexibilité pour l'ordonnancement de groupes. Ces mesures sont le nombre d'ordonnancements semi-actifs caractérisés et le nombre de groupes. Pour que le système homme-machine soit efficace, il faut que le référentiel soit commun, c'est-à-dire que la mesure de flexibilité soit comprise par l'humain et la machine.

Le nombre d'ordonnancements semi-actifs est une mesure complexe pour l'humain. En effet, cette mesure est fortement liée à la taille du problème, de façon

non-linéaire. De plus, les valeurs de cette mesure de flexibilité dépassent rapidement le milliard. L'humain a du mal à se représenter de si grands nombres, et aura donc du mal à manipuler cet objectif. Cette mesure de flexibilité nous semble donc être un bien mauvais choix vis-à-vis de la coopération homme-machine.

La mesure de flexibilité ϕ_{Π} proposé par Esswein possède un certain nombre d'avantages favorisant la compréhension de la mesure par l'humain. Tout d'abord, cette mesure s'exprime sous forme de taux. L'humain manipule beaucoup ce genre de mesure, facile à comparer, et ne nécessitant pas de contexte. Le nombre de groupes n'est par contre pas vraiment adapté au référentiel de l'humain. Heureusement, cette mesure est plus facilement appréhendable par l'humain grâce à notre explication de la mesure de la flexibilité ϕ_{Π} en fonction du nombre de décisions à prendre. Cette explication de la mesure de flexibilité est présentée à la section 3.1.4.2. Une flexibilité de $x\%$ signifie alors que l'humain devra décider de l'opération à exécuter x fois sur 100. Cette mesure de flexibilité est donc adaptée à la coopération homme-machine : elle favorise le référentiel commun entre l'homme et la machine.

5.2.3 Décision grâce à l'approche epsilon contrainte

Esswein propose d'utiliser l'approche epsilon contrainte pour générer l'ordonnancement de groupes. L'humain donne à la machine la qualité dans le pire des cas minimale qu'il accepte. La machine optimise alors la flexibilité tout en respectant la contrainte sur la qualité dans le pire des cas imposée par l'humain. La machine retourne alors un ordonnancement de groupes satisfaisant la contrainte sur la qualité dans le pire des cas, et donne la flexibilité et la qualité dans le meilleur des cas de l'ordonnancement de groupes proposé.

Cette approche est finalement une généralisation de l'approche d'ORABAID : la contrainte imposée par l'humain à la machine est plus flexible et plus générique. L'humain a alors plus de contrôle sur le résultat final.

Par contre, l'humain influe sur la machine avant la résolution du problème grâce à des paramètres. L'humain doit alors faire ses choix sans posséder de connaissance sur les caractéristiques du problème : il est difficile pour l'humain d'évaluer la pertinence de son choix car il ne possède pas d'information sur le potentiel de flexibilité du problème avant d'effectuer sa décision.

5.2.4 Décision grâce à un ensemble de solutions non dominées

Pour pallier ce problème, nous souhaitons que l'humain n'ait aucun paramètre à donner à la machine, et qu'il choisisse une solution parmi un ensemble de solutions non dominées. L'humain doit ensuite choisir parmi ces différents compromis la solution à exécuter. Cette approche est possible grâce à la génération d'un ensemble

de solutions non dominées. Nous proposons un tel algorithme à la section 3.2.4.3 page 37, et le résultat est visible sur la figure 3.3 page 39.

Cette méthode de coopération évite que l'humain ne donne des paramètres à l'algorithme, ce qui nécessite une bonne compréhension de la méthode de résolution. Elle permet également à l'humain de faire le compromis en étudiant les différentes options possibles, ce qui lui permet à d'effectuer son choix en connaissant les caractéristiques du problème. De plus, le fait que l'humain intervienne dans la résolution du problème lors de la dernière étape lui permet de se sentir plus impliqué dans le processus de décision.

Cette phase proactive permet de choisir l'ordonnancement de groupes qui sera exécuté dans l'atelier. Un compromis doit être fait entre la qualité et la flexibilité, cette flexibilité favorisant la robustesse de l'ordonnancement. Mais pour obtenir un ordonnancement réellement robuste, cette flexibilité doit être utilisée de façon efficace, sans quoi la flexibilité est inutile. Dans le chapitre suivant, nous présentons cette phase réactive de l'ordonnancement de groupes, avec comme objectif de favoriser la coopération homme-machine pour maximiser l'utilisation de la flexibilité.

Chapitre 6

Coopération pour la phase réactive

La phase réactive est primordiale pour obtenir une méthode robuste. En effet, durant cette phase, les données réelles sont disponibles. L'humain possède une place privilégiée dans ce système car sa connaissance de l'atelier est plus réaliste que celle de la machine. De plus, l'humain est souvent nécessaire dans l'atelier pour d'autres tâches que l'ordonnancement en lui-même, comme la sécurité et la qualité. Un bon système homme-machine doit donc combiner efficacement les différents atouts de l'humain et de la machine, tout en éveillant sa vigilance.

ORABAID (ORdonnancement d'Atelier Basé sur l'AIde à la Décision) est une méthode d'ordonnancement basée sur l'ordonnancement de groupes. Cette méthode est implémentée dans le progiciel ORDO [RBV95].

Tout d'abord, nous étudions la phase réactive d'ORABAID pour trouver ses avantages et ses défauts. Ensuite, nous proposons une nouvelle phase réactive pour l'ordonnancement de groupes, en tenant compte de l'analyse effectuée précédemment. Ce travail est également présenté dans [PMH08].

6.1 Analyse de la phase réactive d'ORABAID

La phase réactive d'ORABAID utilise la marge libre séquentielle. La machine donne à l'opérateur la marge libre séquentielle de chaque opération du groupe à exécuter. La machine suggère d'exécuter l'opération possédant la plus grande marge. L'opérateur peut néanmoins décider d'exécuter une autre opération, qu'elle fasse partie du groupe en cours d'exécution ou pas. La machine l'informe sur les répercussions de son choix sur les marges des opérations, et l'opérateur valide ou non en dernier ressort.

Ainsi, cette phase réactive permet un contrôle total du décideur sur l'ordonnancement tout en garantissant une certaine qualité. De plus, en cas de retard possible, le problème est détecté avant qu'il ne soit trop tard, ce qui permet au décideur d'anticiper et donc de réagir plus tôt.

Le principal inconvénient de cette phase réactive est qu'elle laisse le décideur choisir l'opération avec très peu d'aide : soit le décideur choisit l'opération proposée par le système, soit il doit évaluer les conséquences des autres décisions une à une, sans avoir d'autre indicateur que la marge libre séquentielle. Le principal risque est que le système soit sous-utilisé : devant la complexité de la tâche, le décideur suivra la proposition donnée par la machine sans se poser de question comme le remarque Cegarra dans [Ceg04]. La flexibilité offerte par l'ordonnancement de groupes serait alors sous-utilisée et l'humain n'interviendrait que très peu dans la réalisation de l'ordonnancement.

6.2 Propositions pour la phase réactive

Comme nous l'avons évoqué à la section 4.2.1, de nombreuses expérimentations montrent que les performances obtenues en combinant l'humain avec la machine sont meilleures que celles obtenues en utilisant uniquement l'un ou l'autre. Il est donc utile de favoriser cette coopération pour améliorer l'ordonnancement.

Pour cela, nous proposons une approche multicritère pour l'utilisation en ligne d'un ordonnancement de groupes. Un certain nombre d'indicateurs sont proposés à l'opérateur pour l'aider dans son choix. Nous pouvons remarquer que, dans ses perspectives [Ess03], Esswein indique qu'utiliser une approche multicritères pour la phase réactive pourrait permettre d'améliorer la méthode ORABAID. Cette modélisation multicritères semble donc adaptée.

Cette approche multicritères est nécessaire car la prise de décisions dans un atelier implique différents critères comme la flexibilité et la performance, leur importance respective étant dépendante du contexte. En effet, on peut imaginer que lorsque l'atelier est soumis à de nombreuses incertitudes, il est important de maximiser sa flexibilité afin d'obtenir une solution plus robuste. Par contre, lorsque l'atelier subit moins d'aléas, il semble judicieux de se focaliser sur les performances de l'atelier.

Plutôt que de proposer un unique indicateur à l'opérateur comme le fait ORABAID, nous suggerons d'en proposer plusieurs. Ces indicateurs devront représenter les connaissances de la machine sur l'ordonnancement de groupes en cours d'exécution, notamment celles pour lesquelles la machine est adaptée : réaliser des calculs. Chacun de ces indicateurs peut représenter soit un critère de décision, soit donner des informations complémentaires sur les opérations à l'opérateur.

Ainsi, l'opérateur reçoit des informations qu'il ne pourrait pas connaître seul, et peut les combiner à ses propres compétences et connaissances comme sa gestion des compromis et sa connaissance de l'état de l'atelier. L'opérateur peut ainsi faire son choix de manière éclairée, sans que la machine ne lui dicte directement une réponse. Il serait ainsi plus impliqué dans le processus de décision et la coopération homme-machine serait plus efficace.

Les indicateurs calculés propres à l'ordonnancement de groupes que nous proposons sont :

- la marge libre séquentielle d'une opération, également proposée par ORA-BAID ;
- la qualité dans le pire des cas pour une opération, qui donne la pire qualité prédite de l'ordonnancement si la décision de l'opérateur est d'exécuter cette opération ;
- la qualité dans le meilleur des cas pour une opération, qui donne la meilleure qualité prédite de l'ordonnancement si la décision de l'opérateur est d'exécuter cette opération.

Pour la qualité dans le meilleur et dans le pire des cas, différents objectifs réguliers peuvent être utilisés, notamment le *makespan* et le retard algébrique maximum. Ces indicateurs sont directement reliés à des critères tels que la flexibilité pour la marge libre séquentielle, le respect des dates de livraisons pour le retard algébrique maximum dans le pire et le meilleur des cas.

Des indicateurs plus simples peuvent être également proposés à l'opérateur :

- les durées des opérations ($p_{i,j}$) ;
- les temps d'exécution restant sur les travaux ;
- le nombre d'opérations restant dans le travail ;
- les dates de livraison des travaux.

Ces indicateurs donnent des informations complémentaires directement liées aux opérations, permettant à l'opérateur de mieux les appréhender.

Nous pensons que cette nouvelle formulation de la phase réactive de l'ordonnancement de groupes devrait favoriser la coopération homme-machine et ainsi améliorer les performances de l'ordonnancement. Nous proposons dans la section suivante un protocole expérimental permettant de vérifier ces hypothèses.

6.3 Expérimentation

6.3.1 Objectif

Pour valider notre système homme-machine, nous allons formuler un certain nombre d'hypothèses. Ces hypothèses seront ensuite évaluées empiriquement. Les résultats seront alors analysés.

Le but de cette expérimentation est de comparer la méthode réactive d'ORA-BAID avec notre nouvelle méthode réactive. Comme la différence entre les deux méthodes peut se résumer aux indicateurs proposés à l'opérateur, l'expérience doit étudier l'impact de ces indicateurs sur la coopération et sur les performances.

Nous espérons que notre nouvelle méthode rendra l'opérateur plus actif [RBW88] dans le système homme-machine, c'est-à-dire qu'il participera directement à la réalisation de l'ordonnancement avec l'aide de la machine et non qu'il utilisera la machine sans rien apporter au système (dans ce cas, l'opérateur sera passif). Il est bien sûr possible que l'opérateur soit plus ou moins actif, c'est pourquoi nous parlerons de niveau d'activité. Cette activité de l'opérateur devrait rendre le système homme-machine plus efficace et ainsi les performances de l'ordonnancement

devraient être meilleures. L'activité de l'opérateur ajouterait également un autre élément : l'opérateur serait plus attentif à ce qui se passe dans l'atelier, ce qui lui permettrait de mieux assumer ses responsabilités sur le bon fonctionnement de l'atelier.

6.3.2 Plan de l'expérience

Dans cette section, nous allons lister les facteurs de l'expérimentation, qui correspondent aux données que nous maîtrisons ; et les variables dépendantes, qui correspondent aux résultats de l'expérimentation.

Le premier facteur de cette expérimentation sera les indicateurs proposés. Suivant que nous proposons uniquement la marge libre séquentielle ou d'autres indicateurs, nous obtenons la méthode réactive d'ORABAID ou notre nouvelle méthode.

Le second facteur sera le niveau d'activité de l'opérateur. Nous devons donc contrôler le niveau d'activité durant l'expérience.

Nous utiliserons comme variable dépendante les performances de l'ordonnancement élaboré grâce au système homme-machine. Les objectifs classiques de l'ordonnancement, notamment le *makespan* et le retard algébrique maximum seront utilisés.

Une autre variable dépendante sera l'utilisation des différents indicateurs par l'opérateur. Cette variable nous permettra d'identifier les indicateurs les plus utilisés par l'opérateur, et donc les indicateurs les plus pertinents pour le bon fonctionnement du système homme-machine.

6.3.3 Procédure expérimentale

Pour cette expérimentation, nous avons décidé d'utiliser un simulateur de notre chaîne de production (voir section 3.4) plutôt que la chaîne elle-même. Utiliser un simulateur offre différents avantages comme pouvoir réaliser plusieurs expérimentations en parallèle, pouvoir effectuer ces expérimentations même lorsque la chaîne est utilisée et pouvoir effectuer ces expérimentations dans des lieux différents. De plus, effectuer ces expérimentations sur la chaîne réelle nous impose de réaliser ces expérimentations en temps réel. Cette contrainte présente deux inconvénients. Tout d'abord, les expérimentations seraient plus longues que sur le simulateur, et prendraient donc plus de temps. Ensuite, l'opérateur aurait un temps de réflexion fixe et important. Le fait que le temps de réflexion soit fixe nous empêche de connaître la durée de la prise de décision. Le fait que le temps de réflexion soit plus important ne reflète pas la réalité : dans un atelier de production, l'opérateur effectue d'autres tâches que ces prises de décision, et n'utilise donc pas la totalité de ce temps pour réfléchir à sa prochaine décision.

L'architecture logicielle sera composée de deux programmes : un programme simulant l'atelier de production appelé simulateur, et un programme de supervision surveillant le programme de simulation et calculant les différents indicateurs appelé

superviseur. Le superviseur donnera les indicateurs à l'opérateur. Il fait partie du système homme-machine étudié. Le simulateur possède également des interactions avec l'opérateur : l'opérateur lui donnera sa décision (l'opération à exécuter). Ce programme ne fait pas partie du système homme-machine étudié : il simule l'atelier de production.

La simulation ne s'effectue pas en temps réel : le simulateur avance dans le temps le plus rapidement possible jusqu'à ce qu'il arrive à un événement nécessitant une décision de l'opérateur. La simulation se met alors en pause et attend la décision de l'opérateur. L'opérateur peut alors consulter le superviseur (et observer l'état de l'atelier sur le simulateur) pour choisir l'opération à exécuter. Une fois cette opération choisie, l'opérateur indique au simulateur sa décision. Cette procédure se poursuit jusqu'à la fin de l'ordonnancement, moment où on peut obtenir les performances de l'ordonnancement.

Pour obtenir des résultats plus réalistes, des incertitudes seront présentes lors de l'exécution de l'ordonnancement : les données traitées par le simulateur (durées opératoires, temps de transfert des produits d'une machine à une autre) seront plus ou moins différentes des informations du superviseur. Ainsi, nous pourrons tester différents scénarios, comme par exemple :

- Il n'y a pas d'incertitudes (les données du simulateur et du superviseur sont les mêmes) : le superviseur donnerait alors des prédictions toujours exactes.
- De petites incertitudes sont présentes tout au long de l'ordonnancement (par exemple, les durées opératoires sont légèrement différentes entre les données du simulateur et du superviseur ou des temps de transports entre les opérations sont présents dans la simulation, mais le superviseur ne les connaît pas) : le superviseur donnerait alors des prédictions approximatives.
- Une incertitude localisée, mais de grande ampleur (par exemple, une durée opératoire pour une opération donnée sur le superviseur est 20 fois plus longue que dans les données du simulateur) : Nous serions alors dans le cas d'un retard causé par un événement isolé, le superviseur ne prédirait pas ce retard, mais s'adapterait après l'incertitude à l'état du simulateur. Par la suite, le simulateur donnerait des prédictions exactes, mais aux performances bien moindres que celles prévues au départ.

Lors de ces expériences, nous allons enregistrer un certain nombre de données qui serviront au dépouillement :

- Pour chaque décision, nous enregistrerons une date de début et une date de fin. Cette information nous permettra tout d'abord de déterminer le temps passé par l'opérateur sur chaque décision. Elle nous permettra aussi de placer les différentes actions enregistrées dans le temps pour une décision donnée.
- Pour suivre l'utilisation des indicateurs par l'opérateur, notre version du superviseur ne montrera pas directement les indicateurs. L'utilisateur devra explicitement choisir les données à afficher en détail. Pour accéder à ces données, il devra cliquer sur la case vide correspondant à la donnée qu'il veut voir pour que celle-ci s'affiche. Ainsi, nous pouvons enregistrer les dates des

différents accès aux indicateurs. Nous pouvons ainsi savoir quels indicateurs sont utilisés pour une décision, ainsi que le moment où ils sont utilisés dans le processus de décision.
- Pour évaluer la coopération du système homme-machine, nous effectuerons un enregistrement sonore de l'opérateur durant l'expérimentation. L'opérateur aura pour consigne de verbaliser ce qu'il fait, c'est-à-dire de dire tout haut ce qui lui passe spontanément par la tête. Ainsi, nous pourrons suivre le processus de coopération avec la machine grâce à ces enregistrements.

Ces enregistrements seront ensuite dépouillés en utilisant la méthode d'analyse des activités cognitives de diagnostic et prise de décision de [HA99], en utilisant le logiciel d'analyse exploratoire des données MacSHAPA [SSJ+94].

Comme nous l'avons dit dans la section précédente, le niveau d'activité de l'opérateur sera un facteur, et nous avons donc besoin de le contrôler. Ceci permettra de voir l'impact du niveau d'activité de l'opérateur dans ce système homme-machine, et ainsi déterminer s'il est nécessaire de favoriser cette activité. Pour rendre actif un opérateur passif, après que l'opérateur ait pris sa décision, la machine suggérera une autre solution. Cette suggestion forcera l'opérateur à justifier son choix, où à justifier le choix de la machine, et ainsi le rendra actif dans le processus de décision.

Les participants à cette expérience seront des étudiants en troisième année d'études supérieures en gestion de la production. Bien que ce ne soient pas des professionnels de l'ordonnancement, leur formation leur donne une connaissance des ateliers de productions et de l'ordonnancement. Le premier échantillon sera composé d'une dizaine de personnes. Si les résultats ne sont pas suffisamment fiables, il sera par la suite possible d'effectuer d'autres expériences pour rendre les résultats concluants. Pour valider les résultats, l'analyse se basera sur des méthodes statistiques permettant de connaître les intervalles de confiances correspondant aux résultats.

6.3.4 Résultats

Le résultat le plus important sera l'influence du niveau d'activité sur les performances et la coopération. En effet, si l'activité permet d'obtenir de meilleures performances, proposer une méthode favorisant cette activité est primordiale. C'est l'hypothèse que nous favorisons. Bien qu'il soit peu probable que l'activité diminue la coopération, il est possible qu'elle diminue les performances. Dans ce cas, cela révélerait que la machine seule serait plus efficace que le système homme-machine. Il serait dans ce cas plus judicieux de proposer une méthode entièrement automatisée, plutôt qu'un système homme-machine. Par rapport à la littérature existante et à nos intuitions, cette hypothèse nous semble peu probable. Il reste une dernière possibilité : l'activité n'influe pas sur les performances et/ou la coopération. Dans ce cas, il n'est pas primordial de favoriser cette activité, bien qu'elle puisse avoir d'autres effets positifs, notamment sur l'attention de l'opérateur.

Un autre résultat important est celui pour lequel cette expérimentation sera réalisée : la comparaison de la méthode ORABAID et de notre nouvelle méthode vis-à-vis des performances et de la coopération. Nous pensons que les performances et la coopération seront meilleures avec notre nouvelle méthode. Si cette hypothèse est confirmée, l'apport de notre nouvelle méthode sera alors confirmé. Si, par contre, la méthode ORABAID surpasse notre nouvelle méthode, nous devrons analyser pourquoi, pour proposer par la suite une autre méthode mieux adaptée à ce problème.

Les autres résultats que nous apportera cette expérience nous permettrons de mieux comprendre le fonctionnement du système homme-machine pour la méthode ORABAID ainsi que pour notre nouvelle méthode. Grâce à ces informations, nous pourrons ensuite améliorer le système homme-machine, quels que soient les résultats précédents. Ces résultats sont :

- Les indicateurs utilisés par l'utilisateur, et leur contexte. Ceci nous permettra de sélectionner les indicateurs les plus pertinents pour l'utilisateur, et de les rendre plus accessibles. Suivant les informations utilisées dans les indicateurs (signe, ordre de grandeur, *etc.*), des outils graphiques pourront être utilisés pour rendre leur lecture plus simple.
- Le niveau de confiance accordé par l'opérateur dans le superviseur en fonction des incertitudes. Cette information nous permet d'évaluer la qualité ressentie des algorithmes par l'utilisateur, et éventuellement nous permettrait de cibler les algorithmes à améliorer.
- Les remarques de l'opérateur sur le superviseur. Ces remarques nous permettront d'améliorer le superviseur.

Ces résultats nous permettront donc d'analyser la phase réactive de l'ordonnancement de groupes, que ce soit la méthode ORABAID ou notre nouvelle méthode. Grâce à cette analyse, nous pourrons ensuite améliorer la phase réactive pour faciliter la coopération homme-machine et rendre le système homme-machine plus performant.

6.4 Réalisation de l'expérimentation

Cette expérience nécessite beaucoup de temps. Tout d'abord, nous sommes liés à l'emploi du temps des étudiants. Ensuite, vu le nombre de participants, l'expérience doit se dérouler sur plusieurs semaines. Enfin, les résultats de l'expérience prendront plusieurs semaines pour être analysées.

C'est pourquoi il n'a pas été possible de réaliser cette expérience durant ce doctorat. Cette expérimentation est planifiée pour l'année scolaire 2008–2009, et sera réalisée en collaboration avec Clément Guérin qui a commencé son doctorat en psychologie en 2008 sous la direction de Jean-Michel Hoc.

6.5 Outils nécessaires

Pour proposer cette nouvelle approche pour la phase réactive de l'ordonnancement de groupes, plusieurs outils propres à cette méthode d'ordonnancement sont nécessaires. La marge libre séquentielle ainsi que l'évaluation d'un ordonnancement de groupes dans le pire des cas sont des outils déjà disponibles, et décrits à la section 3.2. L'évaluation de l'ordonnancement de groupes dans le meilleur des cas n'est pas abordée dans la littérature : seul [Ess03] propose une borne supérieure obtenue par construction, inutilisable dans ce contexte. C'est pourquoi nous proposons, dans la partie suivante, plusieurs méthodes de résolution du meilleur des cas dans un ordonnancement de groupes : des bornes inférieures, des heuristiques et une méthode exacte.

Troisième partie

Le meilleur des cas dans un ordonnancement de groupes

Chapitre 7

Les bornes inférieures

Nous commençons l'étude du meilleur des cas d'un ordonnancement de groupes par des bornes inférieures. Ces bornes inférieures peuvent soit s'utiliser directement pour obtenir une borne inférieure de la qualité d'un ordonnancement de groupes dans le meilleur des cas, soit être utilisées comme outil pour des heuristiques ou des méthodes exactes.

Dans un premier temps, nous proposons une borne inférieure de la date de fin d'une opération. Ensuite, nous proposons des bornes inférieures de l'ordonnancement de groupes pour les objectifs réguliers. Finalement, nous proposons une borne inférieure adaptée au *makespan*. Cette borne inférieure repose sur une relaxation du problème en plusieurs *one machine problems*.

7.1 La date de fin au plus tôt dans le meilleur des cas

Dans un premier temps, nous proposons une borne inférieure de la date de fin d'une opération. Cette borne est le cœur de toutes les bornes inférieures proposées par la suite.

7.1.1 L'algorithme

Dans [ABE05], la date de fin d'une opération dans le pire des cas est calculée en temps polynomial grâce à la programmation dynamique. Le but de cette section est de calculer la date de début au plus tôt, qui correspond à la plus petite valeur que peut prendre C_i dans l'ensemble des ordonnancements semi-actifs décrits par un ordonnancement de groupes. Comme ce problème est NP-difficile [1], nous proposons d'en calculer une borne inférieure en temps polynomial.

1. Comme une réduction entre $F||C_{\max}$ et le meilleur *makespan* dans un ordonnancement de groupes est trivial, et que $F||C_{\max}$ est NP-difficile, alors la date de début au plus tôt d'une opération dans un ordonnancement de groupes est NP-difficile.

Il est facile de calculer une borne inférieure à notre problème en utilisant une relaxation sur les ressources (C'est-à-dire en considérant les ressources comme des ressources à capacité infinie). Dans ce cas, la borne inférieure de la date de début d'une opération dans le meilleur des cas (θ_i) est calculée comme le maximum (des bornes inférieures) des dates de fin au plus tôt (χ_i) de tous ses prédécesseurs. Pour l'opération O_i, les prédécesseurs correspondent aux prédécesseurs donnés par le problème ($\Gamma^-(i)$) mais également aux opérations du groupe précédent (les opérations du groupe $g^-(i)$). Par exemple, dans l'exemple décrit figure 3.1, les prédécesseurs de l'opération O_6 (exécutée sur M_1) sont l'opération O_5 (exécutée sur M_3) à cause de la contrainte de précédence, et les opérations O_1 et O_8 (exécutées sur la même machine M_1) car elles sont dans le groupe prédécesseur ($g^-(6)$). Nous avons donc :

$$\begin{cases} \theta_i = \max\left(r_i, \max\limits_{j \in g^-(i)} \chi_j, \max\limits_{j \in \Gamma^-(i)} \chi_j\right) \\ \chi_i = \theta_i + p_i \end{cases} \tag{7.1}$$

Calculer θ_i en utilisant (7.1) est équivalent au calcul de la tête de l'opération O_i comme décrit dans [CP89].

Mais cette borne peut être améliorée grâce à une propriété de l'ordonnancement de groupes : une opération dans un groupe donné ne peut être exécutée avant que toutes les opérations du groupe prédécesseur ne soient exécutées. Ainsi, une opération ne peut commencer qu'après la plus petite date de fin du groupe prédécesseur.

Il est donc nécessaire de calculer la plus petite date de fin d'un groupe ou une borne inférieure de cette date (que nous appelons $\gamma_{g_{\ell,k}}$). Nous avons précédemment calculé les θ_i du groupe, qui sont des bornes inférieures aux dates de disponibilité des opérations. Pour calculer une borne inférieure de la plus petite date de fin d'un groupe, nous pouvons générer un problème du type $1|r_i|C_{\max}$ qui correspond à notre problème, avec $r_i = \theta_i$. Ce problème est résolu en temps polynomial en ordonnant les opérations par ordre croissant de date de disponibilité [BK07, Law73].

La borne inférieure améliorée se formule donc ainsi :

$$\begin{cases} \theta_i = \max\left(r_i, \gamma_{g^-(i)}, \max\limits_{j \in \Gamma^-(i)} \chi_j\right) \\ \chi_i = \theta_i + p_i \\ \gamma_{g_{\ell,k}} = C_{\max} \text{ of } 1|r_i|C_{\max}, \forall O_i \in g_{\ell,k}, r_i = \theta_i \end{cases} \tag{7.2}$$

7.1.2 Exemple de l'exécution de l'algorithme

Voyons un exemple pour illustrer ces algorithmes. Pour garder cet exemple compréhensible, nous utiliserons un problème simple de type *flow shop*. Le problème, l'ordonnancement de groupes ainsi que les valeurs calculées par les algorithmes sont présentés dans le tableau 7.1. Une représentation dans le pire des cas de l'ordonnancement de groupes est disponible figure 7.1. Les quatre ordonnance-

Description du problème					Équation (7.1)		Équation (7.2)			Val. optimale	
i	$\Gamma^-(i)$	m_i	p_i	$g(i)$	θ_i	χ_i	θ_i	χ_i	$\gamma_{g(i)}$	θ_i	χ_i
1	$\varnothing$	M_1	1	$g_{1,1}$	0	1	0	1	4	0	1
2	$\{1\}$	M_2	2	$g_{2,1}$	1	3	1	3	5	1	3
3	$\varnothing$	M_1	3	$g_{1,1}$	0	3	0	3	4	0	3
4	$\{3\}$	M_2	2	$g_{2,1}$	3	5	3	5	5	3	5
5	$\varnothing$	M_1	1	$g_{1,2}$	3	4	**4**	**5**	5	4	5
6	$\{5\}$	M_2	1	$g_{2,2}$	5	6	5	6	6	**6**	**7**

TAB. 7.1: Exemple d'un problème de *flow shop*

ments semi-actifs décrits par l'ordonnancement de groupes sont représentés sur la figure 7.4. Ils sont utilisés pour calculer la valeur optimale dans le tableau 7.1.

La figure 7.2 et la figure 7.3 sont une représentation des calculs de (7.1) et (7.2) respectivement. La relaxation sur les ressources est illustrée par le parallélisme de l'exécution des opérations des opérations dans le même groupes (par exemple O_1 et O_3). L'exécution des groupes (calculé par $\gamma_{g_{\ell,k}}$) est représenté en gris sur la figure 7.3.

Nous pouvons remarquer l'amélioration de (7.2) par rapport à (7.1) pour l'opération O_5. Le calcul du *makespan* du groupe $g_{1,1}$ améliore la borne inférieure χ_5 jusqu'à sa valeur optimale.

Mais le résultat n'est pas toujours optimal. Par exemple, (7.2) ne donne pas la valeur optimale pour χ_6 : la valeur optimale est 7, mais (7.2) donne 6. Cette approximation est causée par le fait que deux dates de début dans le meilleur des cas n'apparaissent pas forcément dans le même ordonnancement semi-actif. Dans notre exemple, il ne peut y avoir d'ordonnancement semi-actif avec $t_2 = \theta_2 = 1$ (premier ordonnancement de la figure 7.1) et $t_4 = \theta_4 = 3$ (quatrième ordonnancement de la figure 7.1). En effet si $t_2 = 1$, alors t_4 ne peut être égale à 3 mais seulement à 4 à cause des contraintes de précédences. Ainsi, $\gamma_{g_{2,1}}$ n'est pas égale au *makespan* optimal du groupe $g_{2,1}$ (6, comme dans le premier ordonnancement de la figure 7.1), mais à une borne inférieure (5). Par la suite, l'erreur se propage à l'opération O_6.

7.1.3 Complexité

L'équation (7.2) peut être représenté par un graphe, avec un nœud pour chaque variable calculée (θ_i, χ_i et $\gamma_{g_{\ell,k}}$) et, pour chaque nœud, un arc qui vient des variables utilisées pour la calculer. Ce graphe est un graphe orienté acyclique. La complexité de (7.2) correspond à la complexité du calcul du plus long chemin dans ce graphe plus la complexité des algorithmes exécutés à chaque nœud. La complexité du

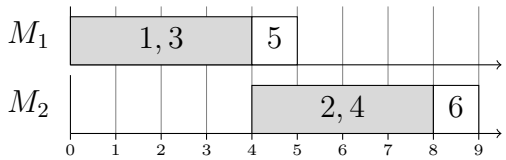


FIG. 7.1: L'ordonnancement de groupes décrit par le tableau 7.1

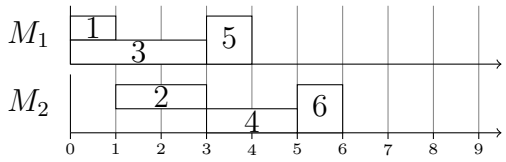


FIG. 7.2: Représentation graphique des résultats de (7.1)

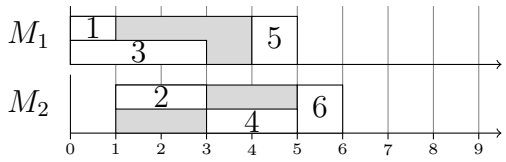


FIG. 7.3: Représentation graphique des résultats de (7.2)

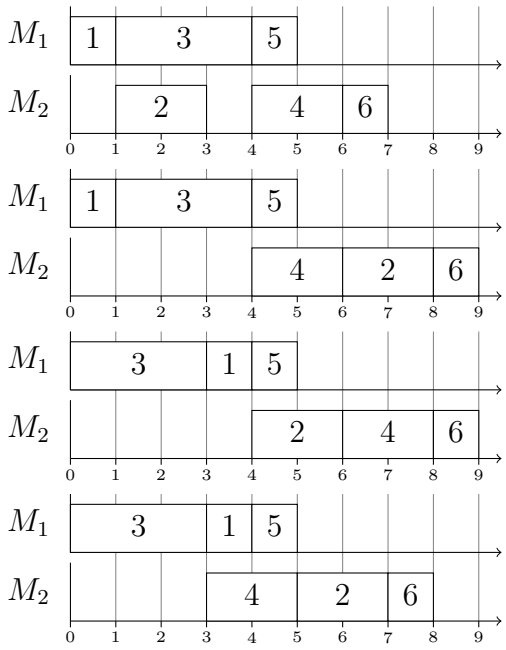


FIG. 7.4: Les ordonnancements semi-actifs décrits par la figure 7.1

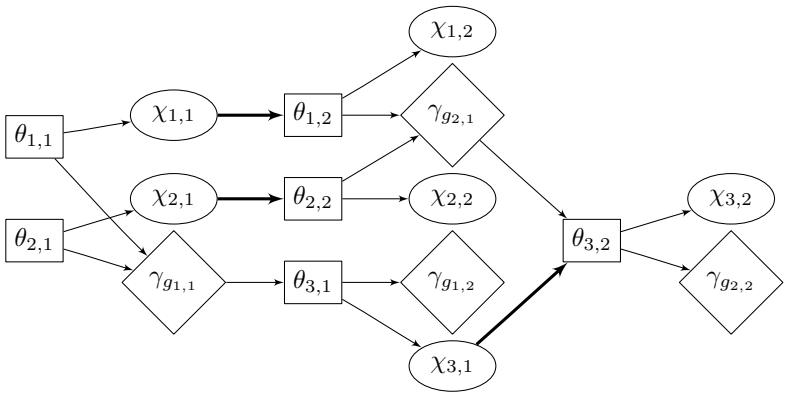


FIG. 7.5: Le graphe correspondant au calcul de (7.2) pour le problème décrit sur le tableau 7.1

calcul du plus long chemin dans un graphe orienté acyclique est $O(M)$, avec M le nombre d'arcs, en utilisant l'algorithme de Bellman.

Pour chaque nœud χ_i, il y a un arc venant du nœud θ_i. Donc, dans le graphe, il y a n arcs de ce genre, n étant le nombre d'opérations.

Soit k_i le nombre de prédécesseur de l'opération O_i, il y a $k_i + 1$ nœuds allant vers chaque θ_i : k_i venant des prédécesseurs de O_i et un venant du nœud du groupe $g^-(i)$. Il y a donc $O((\max k_i + 1)n)$ arcs allant vers tous les nœuds θ_i. On pose $\max k_i = k_{\max}$.

Les arcs incidents à un nœud $\gamma_{g_{\ell,k}}$ viennent des nœuds θ_i avec O_i appartenant à $g_{\ell,k}$. Comme les groupes forment une partition de l'ensemble des opérations, chaque nœud θ_i possède un unique arc allant vers un nœud $\gamma_{g_{\ell,k}}$. Il y a donc n arcs de ce type dans tout le graphe.

Le nombre d'arcs dans le graphe est donc

$$O(n) + O((k_{\max} + 1)n) + O(n) = O(k_{\max} n)$$

Sur les nœuds $\gamma_{g_{\ell,k}}$, un algorithme avec une complexité en $O(h_{\ell,k} \log h_{\ell,k})$ est utilisé, avec $h_{\ell,k}$ le nombre d'opération dans le groupe $g_{\ell,k}$. La complexité globale des algorithmes effectués sur les nœuds est donc :

$$\begin{aligned} O(h_1 \log h_1 \times \cdots \times h_n \log h_n) &= O(n \log(\max h_i)) \\ &= O(n \log n) \end{aligned}$$

La complexité de (7.2) est donc :

$$O = O(k_{\max} n) + O(n \log n) = O(k_{\max} n + n \log n).$$

Pour les problèmes classiques de *job shop*, $k_{\max} = 1$, et la complexité de (7.2) est de $O(n \log n)$. Dans le pire des cas, $k_{\max} = n - 1$, et sa complexité est de $O(n^2)$.

7.2 Bornes inférieures pour les objectifs réguliers

Dans cette section, nous proposons des bornes inférieures de l'ordonnancement de groupes pour tout objectif régulier, et plus particulièrement pour ceux de type minmax ou minsum (voir la section 1.1.5.2). Ces bornes inférieures utilisent les bornes décrites à la section précédente.

Comme une fonction objective régulière est une fonction croissante des C_i par définition, une borne inférieures des C_i permet de calculer une borne inférieure pour tout objectif régulier. Pour cela, nous pouvons utiliser χ_i.

Étudions le cas des fonctions régulières de type minsum ($\sum f_i$, voir la section 1.1.5). χ_i permet de calculer directement une borne inférieure d'une telle fonction objectif :

$$\mathrm{LB}\left(\sum f_i\right) = \sum_i f_i(\chi_i)$$

Mais, grâce au fait que nous utilisons des groupes d'opérations permutables, cette borne peut être améliorée. Dans chaque groupe, il y a forcément une opération qui sera exécutée en dernière position. La date de fin de cette opération a pour borne inférieure $\gamma_{g(i)}$ (la borne inférieure de la fin d'un groupe). L'idée est d'utiliser le $f_i(\gamma_{g(i)})$ minimal dans chaque groupe comme borne inférieure de $f_i(C_i)$. Cette borne inférieure, noté $GLB_{f_i}(O_i)$ (GLB pour *Group Lower Bound*), se formule ainsi :

$$GLB_{f_i}(O_i) = \begin{cases} f_i(\gamma_{g(i)}) & \text{si } f_i(\gamma_{g(i)}) = \min\limits_{j|O_j \in g(i)} f_j(\gamma_{g(i)}) \\ f_i(\chi_i) & \text{sinon} \end{cases}$$

Cette borne inférieure est valide uniquement si on considère le groupe dans sa globalité, mais pas pour une borne inférieure pour une opération précise.

L'utilisation de GLB_{f_i} améliore la borne inférieure des objectifs réguliers de type minsum :

$$\mathrm{LB}\left(\sum f_i\right) = \sum_i GLB_{f_i}(O_i)$$

Le même raisonnement peut être effectué pour les objectifs réguliers de type minmax. La formulation est simplifiée grâce aux propriétés de la fonction maximum :

$$\begin{aligned} \mathrm{LB}(f_{\max}) &= \max_i GLB_{f_i}(O_i) \\ &= \max_{\ell,k} \left(\max \left(\min_{i|O_i \in g_{\ell,k}} f_i(\gamma_{g_{\ell,k}}), \max_{i|O_i \in g_{\ell,k}} f_i(\chi_i) \right) \right) \end{aligned} \tag{7.3}$$

L'objectif régulier de type minmax le plus étudié est le *makespan*. Dans la section suivante, nous étudions cet objectif.

7.3 Bornes inférieures pour le *makespan*

Dans cette section, nous proposons une borne inférieure adaptée au *makespan*, en s'inspirant de la borne inférieure classique du *job shop*. Cette borne inférieure repose sur une relaxation du problème en plusieurs *one machine problems*.

7.3.1 Utilisation de la formulation pour les objectifs réguliers

L'équation (7.3) permet d'obtenir une borne inférieure du *makespan* :

$$\begin{aligned} \mathrm{LB}(C_{\max}) &= \max_{\ell,k} \left(\max \left(\gamma_{g_{\ell,k}}, \max_{i|O_i \in g_{\ell,k}} \chi_i \right) \right) \\ &= \max_{\ell,k} \gamma_{g_{\ell,k}} \end{aligned}$$

Pour améliorer cette borne, nous nous inspirons de la borne inférieure utilisée pour le problème du *job shop*.

7.3.2 La borne inférieure classique du *job shop*

La borne inférieure classique du *job shop*, décrite dans [CP89], est basée sur la relaxation du problème en différents problèmes à une machine : pour chaque machine, toutes les opérations s'exécutant sur la machine sont prises. Pour chaque opération, trois valeurs sont utilisées :

- la tête r_i (*head* en anglais), qui est une borne inférieure de la date de début au plus tôt de l'opération dans le problème de *job shop* (avec un ordonnancement partiel) ;
- la durée de latence q_i (*tail* en anglais), qui est une borne inférieure de la durée entre la fin de l'opération et la fin de l'ordonnancement (avec un ordonnancement partiel) ;
- le temps d'exécution p_i, qui est le même que dans le problème de *job shop*.

Nous avons alors un *one machine problem* comme défini par [Car82] pour chaque machine : chaque opération ne peut commencer avant r_i, s'exécute pendant une durée p_i sur la machine et l'ordonnancement se termine au moins avec une durée q_i après la fin de l'exécution de l'opération. L'objectif est de minimiser le *makespan* de l'ordonnancement, calculé $\max_i(C_i + q_i)$. Ce problème est équivalent à $1|r_i|L_{\max}$ avec $d_i = -q_i$. Il est NP-difficile ([LRKB77] cité par [BK07]). [Car82] propose un algorithme exact efficace pour résoudre ce problème. La borne inférieure classique pour ce problème s'appelle le *Jackson's preemptive schedule* et a une complexité de $O(n \log n)$, avec n le nombre d'opérations.

La valeur optimale d'un tel *one machine problem* est une borne inférieure du *makespan* du *job shop*. Ainsi, en pratique, le plus grand *makespan* optimal (ou la plus grande borne inférieure) des *one machine problems* générés est prise comme

borne inférieure. L'efficacité de cette borne dépend de la qualité des têtes r_i et durées de latence q_i comme montré dans [CP90, CP94].

7.3.3 Une borne inférieure améliorée pour le *makespan*

La première étape de la borne inférieure classique du *job shop* est la relaxation en *one machine problem*. Pour l'ordonnancement de groupes, la relaxation se fera naturellement au niveau des groupes plutôt qu'au niveau des machines. Les problèmes seront alors plus petits, ce qui facilitera leur résolution exacte.

Ensuite, des têtes et durées de latence efficaces doivent être trouvés pour ces *one machine problems*. En utilisant (7.2), θ_i est une tête valide et devrait être efficace. À cause de la symétrie des têtes et durées de latence, les durées de latence peuvent être calculées de la même manière que θ_i en utilisant une version renversée de (7.2) : plutôt que de commencer le calcul au début de l'ordonnancement, le calcul commence à la fin. Ainsi, en remplacant les prédécesseurs par les successeurs, la nouvelle formulation est :

$$\begin{cases} \theta'_i = \max\left(\gamma'_{g^+(i)}, \max\limits_{j\in\Gamma^+(i)} \chi'_j\right) \\ \chi'_i = \theta'_i + p_i \\ \gamma'_{g_{\ell,k}} = C_{\max} \text{ of } 1|r_i|C_{\max}, \forall O_i \in g_{\ell,k}, r_i = \theta'_i \end{cases} \tag{7.4}$$

avec θ'_i une durée de latence valide. Les *one machine problems* sont alors générés.

À ce stade, les *one machine problems* doivent être évalués en utilisant le *Jackson's preemptive schedule* ou l'algorithme de Carlier. L'évaluation maximale sera la borne inférieure de l'ordonnancement de groupes.

La borne inférieure basée sur l'algorithme de Carlier sera appelée « Optimal OMP LB » (pour *Optimal resolution of the One Machine Problems Lower Bound*) et la borne inférieure basée sur le *Jackson preemptive schedule* sera appelée « OMP JPS LB » (pour *Jackson Preemptive Schedule resolution of the One Machine Problems Lower Bound*).

7.4 Utilisation des bornes inférieures

Ces bornes inférieures peuvent s'utiliser dans différents contextes.

Tout d'abord, il est possible d'utiliser directement les différentes bornes pour évaluer un ordonnancement de groupes. Cette utilisation est détaillée au chapitre 10.

Ensuite, les bornes inférieures sont des outils indispensable pour les méthodes exactes de type séparation et évaluation. Un tel algorithme utilisant les bornes inférieures présentées dans ce chapitre est décrit au chapitre 9.

Finalement, les travaux présentés dans ce chapitre peuvent également être utiles pour l'élaboration d'heuristiques. C'est le sujet du chapitre suivant.

Chapitre 8

Les heuristiques

Pour compléter notre étude sur le meilleur des cas dans un ordonnancement de groupes, nous proposons des heuristiques adaptées au *makespan*. Ces heuristiques permettent d'obtenir une borne supérieure à la qualité dans le meilleur des cas, ainsi qu'une exécution de l'ordonnancement de groupes.

Pour commencer, nous proposons des règles de priorité. De telles heuristiques ont l'avantage d'être simples et rapides à exécuter. Ensuite, nous proposons une adaptation du *shifting bottleneck* à l'ordonnancement de groupes. Ces heuristiques sont ensuite évaluées grâce à une expérimentation sur des instances de *job shop* de la littérature. Ce travail est également présenté dans [PM08b].

8.1 Règles de priorité

Les heuristiques utilisant des règles de priorité des files d'attente, généralement appelées « règles de priorité », sont parmi les méthodes les plus simples et les plus utilisées pour ordonnancer un atelier de type job-shop. [BPH82] définit une règle de priorité comme une méthode permettant de sélectionner parmi tous les travaux en attente de traitement sur une ressource, le prochain travail à traiter, selon une priorité calculée. Pour un état de l'art sur les règles de priorité, le lecteur pourra se référer à [Hau89].

8.1.1 *Most work remaining*

Comme heuristique de référence, nous utilisons la règle de priorité classique la mieux adaptée au *makespan*. Cette règle s'appelle *most work remaining*, et elle consiste à exécuter le travail possédant le plus long temps d'exécution restant. Cette heuristique sera par la suite notée MWR.

8.1.2 Une règle utilisant la durée de latence

Nous proposons une règle de priorité basée sur la queue de l'opération. L'idée est de donner une priorité importante à une opération possédant une grande durée de latence, c'est-à-dire une opération risquant de perturber le *makespan*. Comme les durées de latence des opérations sont parfois égales, la règle est combinée avec *Shortest Processing Time* (SPT), qui consiste à sélectionner la tâche ayant la plus petite durée opératoire. La règle s'écrit donc ainsi :

$$\min_{\forall O_i} p_i - {\theta'_i}^2$$

Cette heuristique sera par la suite notée SQUTAIL.

8.1.3 Utilisation d'une borne inférieure dans une règle de priorité

Dans la section 7.3, nous définissons une borne inférieure efficace pour le *makespan*. Une borne inférieure étant une estimation de la qualité d'un ordonnancement, son utilisation dans une règle de priorité semble judicieuse.

La règle de priorité que nous proposons est définie comme suit :

1. pour chaque opération présente dans la file d'attente, on génère un ordonnancement (de groupes) partiel où cette opération est sélectionnée immédiatement ;
2. on calcule la borne inférieure pour ces ordonnancements partiels générés ;
3. on sélectionne l'opération correspondant à la plus petite borne inférieure.

Néanmoins, cette règle de priorité conduit à de nombreuses égalités de priorité (*i.e.* des opérations donnant la même borne inférieure). L'utilisation de la borne inférieure seule n'est donc pas suffisante.

C'est pourquoi nous proposons de combiner la borne inférieure avec une autre règle de priorité. Dans un premier temps, les opérations dont l'exécution donne une borne inférieure minimale sont sélectionnées. Dans un second temps, l'opération à exécuter est sélectionnée par une règle de priorité parmi les opérations sélectionnées précédemment.

Les règles de priorité utilisées sont celles décrite dans la section 8.1.1. L'utilisation de MWR avec la borne inférieure sera par la suite nommée LB+MWR (LB pour *lower bound*). De la même façon, l'utilisation de SQUTAIL avec la borne inférieure sera nommée LB+SQUTAIL.

8.2 Un *shifting bottleneck* pour l'ordonnancement de groupes

8.2.1 Le *shifting bottleneck*

Le *shifting bottleneck*, décrit dans [ABZ88], est une heuristique très connue et efficace pour le problème de *job shop* avec comme objectif le *makespan*.

Nous présentons ici une idée globale de l'algorithme. Pour une explication détaillée de l'algorithme, voir [ABZ88].

À chaque itération de l'algorithme, la machine goulot est déterminée, l'ordre d'exécution des opérations sur cette machine est alors fixé. Les machines déjà séquencées sont ensuite réoptimisées. Cette procédure est répétée jusqu'à ce que toutes les machines soient séquencées. On obtient alors l'ordonnancement.

Pour le choix de la machine à séquencer, ainsi que pour son séquencement, la relaxation en *one-machine problem* (présentée à la section 7.3.2) est utilisée. L'algorithme de Carlier [Car82] est utilisé pour séquencer les machines.

Théoriquement, cet algorithme peut donner des ordonnancements non réalisables. En effet, la relaxation en *one-machine problem* ne prend pas en compte les contraintes de précédence. L'algorithme peut donc donner un ordonnancement violant des contraintes donc non réalisable. Pour pallier ce défaut, les auteurs traitent ce cas rare comme une exception.

8.2.2 Adaptation pour l'ordonnancement de groupes

Pour réaliser un *shifting bottleneck* pour l'ordonnancement de groupes, nous pouvons utiliser la relaxation en *one-machine problem* décrit à la section 7.3.3. L'algorithme n'est alors plus effectué au niveau des machines mais au niveau des groupes. Le nombre de réoptimisations est alors beaucoup plus important que dans le *shifting bottleneck* classique. Nous nous attendons donc à de meilleures performances, mais également à un temps de calcul plus élevé.

La relaxation au niveau des groupes permet d'obtenir à coup sûr des ordonnancements réalisables, contrairement à l'algorithme original : comme le choix des séquences est effectué au niveau des groupes, et que toutes les permutations des opérations dans un groupe donne par définition un ordonnancement valide, il ne peut y avoir de violation de contrainte.

Cette heuristique sera par la suite notée SB.

Dans la section suivante, nous comparons ces heuristiques sur un ensemble d'instances très utilisé dans la litterature du *job shop*.

8.3 Expérimentation

8.3.1 Protocole

Le but de ces expérimentations est de comparer les différentes heuristiques présentées dans ce chapitre pour l'ordonnancement de groupes.

Nous utilisons un ensemble d'instance très utilisée nommé `la01` à `la40` de [Law84]. Ce sont des instances de *job shop* classiques, avec m opérations pour chaque travail (m le nombre de machines), chaque opération d'un travail s'exécutant sur une machine différente. Cet ensemble est composé de 40 instances de 8 tailles différentes (5 instances pour chaque taille).

Pour chaque instance, nous générons un ordonnancement de groupes avec une qualité optimale connue et une très grande flexibilité. Pour générer ces ordonnancements, nous utilisons l'algorithme EBJG décrit à la section 3.2.4.2. Cet algorithme commence avec un ordonnancement de groupes à une opération par groupe calculé par l'algorithme exact décrit dans [BJS94] (donc, par construction, le *makespan* optimal de ces ordonnancements de groupes est le *makespan* de cet ordonnancement de groupes à une opération par groupe).

Pour chaque ordonnancement de groupes, nous calculons l'écart entre le *makespan* de chaque heuristique et le *makespan* optimal de l'ordonnancement de groupes. Les résultats sont présentés sous la forme de boîtes à moustache (*boxplots*) dans la figure 8.1. Les résultats et les temps d'exécutions correspondant aux tailles des instances sont présentés dans le tableau 8.1.

La figure 8.2 représente les heuristiques vis-à-vis de deux objectifs : leur temps d'exécution et leur performance. Cette figure utilise les valeurs moyennes, et ne montre donc pas l'influence de la complexité des algorithmes sur les temps d'exécution.

8.3.2 Discussion

MWR possède des performances décevantes par rapport aux autres heuristiques avec un écart moyen à 0,3. Elle est par contre très rapide.

SQUTAIL est bien meilleure que MWR avec une moyenne à 0,14. Par contre, bien qu'elle ne soit qu'une simple règle de priorité, elle est beaucoup plus coûteuse que MWR. En effet, le calcul de θ'_i doit être exécutée à chaque décision, ce qui engendre beaucoup de calculs supplémentaires.

LB+MWR et LB+SQUTAIL ont des performances et des temps d'exécution comparables. La borne inférieure seule comme décrit à la section 8.1.3 serait donc une bonne règle de priorité. Par contre, la règle de priorité associée à la borne inférieure semble avoir moins d'impact sur les résultats, bien que LB+MWR soit légèrement meilleure que LB+SQUTAIL. Ceci est peut-être du au fait que la borne inférieure utilise abondamment les queues θ'_i, et donc MWR apporterait une information supplémentaire. Les deux heuristiques ont des temps de calcul équivalent,

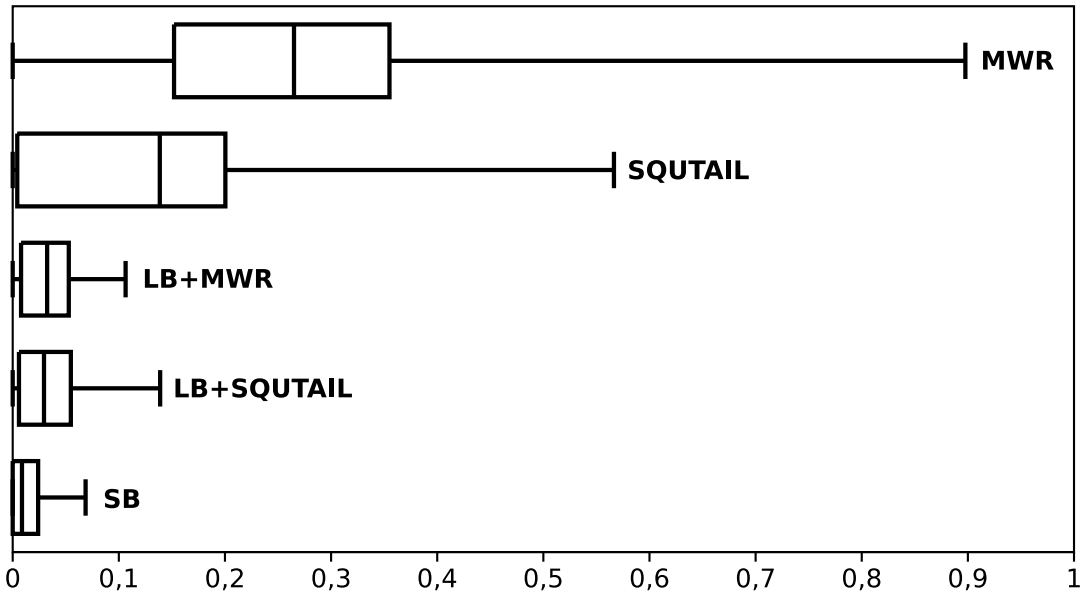


FIG. 8.1: Écarts des heuristiques par rapport à l'optimal

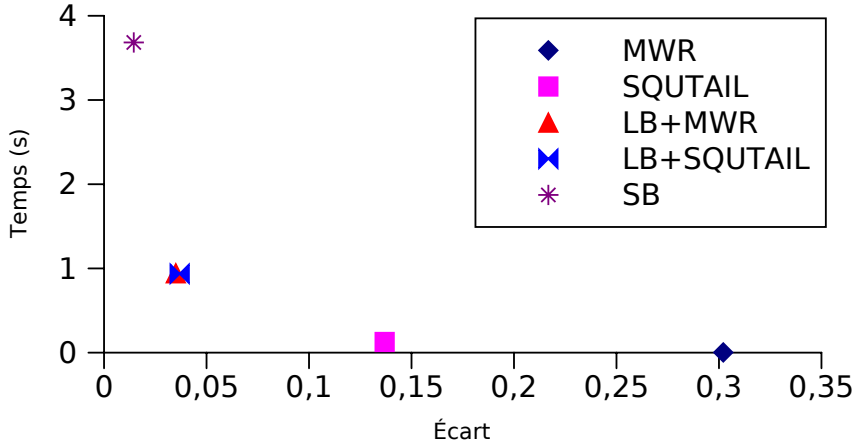


FIG. 8.2: temps d'exécutions et performances moyennes pour les différentes heuristiques

	MWR		SQUTAIL	
Taille	Écart	Tps (s)	Écart	Tps (s)
10×5	0,3014	0,0005	0,0905	0,0072
15×5	0,0972	0,0007	0,0075	0,0202
20×5	0,1347	0,0010	0,0000	0,0399
10×10	0,4509	0,0019	0,2956	0,0322
15×10	0,2941	0,0022	0,1807	0,0862
20×10	0,3094	0,0029	0,1356	0,1720
30×10	0,1847	0,0050	0,0616	0,4559
15×15	0,6441	0,0065	0,3237	0,1996
Moyenne	0,3021	0,0026	0,1369	0,1266

	LB+MWR		LB+SQUTAIL		SB	
Taille	Écart	Tps (s)	Écart	Tps (s)	Écart	Tps (s)
10×5	0,0302	0,0542	0,0385	0,0542	0,0211	0,0524
15×5	0,0136	0,1737	0,0124	0,1714	0,0046	0,1259
20×5	0,0030	0,3392	0,0017	0,3373	0,0000	0,2587
10×10	0,0374	0,1866	0,0318	0,1861	0,0182	0,8292
15×10	0,0426	0,5511	0,0426	0,5506	0,0197	2,2478
20×10	0,0410	1,1926	0,0668	1,1888	0,0201	4,6298
30×10	0,0362	3,8018	0,0269	3,7752	0,0000	10,5885
15×15	0,0758	1,2366	0,0749	1,2273	0,0321	10,7351
Moyenne	0,0350	0,9420	0,0369	0,9364	0,0145	3,6834

La taille est notée $n \times m$ avec n le nombre de travaux et m le nombre de machines.

TAB. 8.1: Écarts moyens et temps d'exécution des différentes heuristiques par rapport à la taille du problème

mais bien supérieurs aux temps d'exécution de SQUTAIL et MWR. En effet, la borne inférieure doit être calculée pour chaque opération dans la file d'attente.

SB est de loin la meilleure heuristique. Avec 17 valeurs optimales trouvées sur 40 et un écart moyen de 0,015, ses performances sont plus de deux fois supérieures à celles de LB+MWR et LB+SQUTAIL et 50 fois supérieures à celle de SQUTAIL. Son temps d'exécution est bien sûr également plus important : elle est environ 4 fois plus coûteuse que LB+SQUTAIL.

8.4 Conclusion

Les différentes heuristiques correspondent à des compromis différents. Le choix de l'heuristique dépendra donc de son utilisation, et aucune ne peut prétendre surpasser les autres. Ainsi, si la rapidité est primordiale, MWR ou SQUTAIL seront des choix judicieux, alors que SB sera intéressant lorsque la performance est importante, et que le temps de calcul n'est pas une contrainte importante. Les règles de priorité utilisant la borne inférieure se situant entre ces deux extrêmes

Mais dans certains cas, il peut être nécessaire de connaître la valeur exacte, plutôt qu'une borne inférieure ou supérieure. C'est pourquoi nous proposons dans le chapitre suivant une méthode de résolution exacte du meilleur des cas pour les critères réguliers.

Chapitre 9

La méthode exacte

Les bornes inférieures et les heuristiques présentées aux chapitres précédents permettent d'obtenir un intervalle de qualité pour la qualité d'un ordonnancement de groupes dans le meilleur des cas. Pour obtenir une évaluation exacte de cette qualité, une méthode exacte est nécessaire. Une telle méthode est présenté dans ce chapitre. Comme ce problème est NP-difficile, une méthode de séparation et évaluation est proposée.

Tout d'abord, nous présentons une méthode exacte originale, basée sur l'énumération des ordonnancements actifs. La procédure de séparation, la méthode de parcours de l'espace de recherche ainsi qu'une condition suffisante permettant de réduire l'espace de recherche sont présentées. Ensuite, des expérimentations sont analysées pour évaluer l'impact des différents composants de cette méthode exacte.

9.1 Description de l'algorithme

Cette section décrit une méthode exacte pour trouver le meilleur ordonnancement dans un ordonnancement de groupes. Cet algorithme est une procédure de séparation et évaluation (*branch and bound* en anglais) qui peut être utilisée pour tout objectif régulier : seule la borne inférieure doit être adaptée à l'objectif.

Une procédure de séparation et évaluation est composée de deux parties : une procédure de séparation et une d'évaluation.

La procédure de séparation prend un problème et le remplace par plusieurs sous problèmes dont l'union comprend le premier problème. Ces problèmes et sous problèmes sont appelé nœuds. Résoudre tous les nœuds résout alors le problème global.

La procédure d'évaluation utilise deux bornes : une borne supérieure correspondant à la qualité de la meilleure solution trouvée jusqu'à présent, et une borne inférieure de la meilleure qualité possible sur un nœud. Si la borne inférieure d'un nœud est plus grande ou égale à la borne supérieure, toutes les solutions découlant de ce nœud auront une qualité supérieure ou égale à la meilleure solution actuelle. Un tel nœud peut alors être abandonné.

9.1.1 Procédure de séparation

9.1.1.1 Description de la procédure

La procédure de séparation se base sur les ordonnancements actifs (voir la section 1.1.6). Comme dans l'ensemble des ordonnancements actifs il existe un ordonnancement optimal pour tout objectif régulier, une telle procédure peut être utilisée dans notre cas.

La méthode classique d'énumération des ordonnancements actifs n'est pas utilisée. Les propriétés des groupes d'opérations permutables sont prises en compte pour simplifier la procédure. Pour un ordonnancement de groupes réalisable, les groupes sont partiellement ordonnés par les contraintes de précédence entre opérations des groupes et l'ordre des groupes sur les machines. Ainsi, séquencer les groupes en suivant cet ordre partiel garantit que toutes les opérations précédant (vis-à-vis des contraintes de précédence et de l'ordre des groupes) les opérations du groupes sont déjà séquencées. Il est donc possible d'énumérer tous les ordonnancements actifs en ordonnançant les groupes les un après les autres en respectant l'ordre partiel.

Un nœud dans l'espace de recherche est représenté par une séquence de groupes et une liste ordonnée de groupes à séquencer. Une solution est un ordonnancement de groupes avec seulement une opération par groupe (et une liste vide de groupes).

En pratique, au début de l'algorithme, les groupes sont ordonnés en respectant leur ordre partiel. Ensuite, les groupes avec une opération sont enlevés car ils sont déjà séquencés. Cette liste de groupes représente les groupes à séquencer et leur ordre.

La génération de nœud se réalise ainsi :

- les dates de début au plus tôt (r_i) de chaque opération dans le premier groupe de la liste sont calculées ;
- la plus petite date de fin est calculée : $C_{\min} = \min_{i|O_i \in g_{k,\ell}} r_i + p_i$;
- toutes les opérations avec une date de début plus petite que la plus petite date de fin ($r_i < C_{\min}$) sont sélectionnées ;
- pour chacune de ces opérations, un nouveau nœud est généré : l'opération est séquencée (un groupe avec l'opération est créé devant le groupe courant et l'opération est enlevée de ce groupe) puis, si le groupe courant ne possède plus qu'une opération, il est enlevé de la liste.

9.1.1.2 Exemple

Pour illustrer la méthode, nous énumérons tous les nœuds de l'ordonnancement de groupes décrit à la figure 3.1. Pour commencer, les groupes sont ordonnés. Une solution est

$$[\{O_4\}, \{O_7\}, \{O_1, O_8\}, \{O_2\}, \{O_9\}, \{O_3, O_5\}, \{O_6\}],$$

Cet ordre n'est pas unique : par exemple, les deux premiers groupes peuvent être inversés, mais $\{O_7\}$ doit être avant $\{O_1, O_8\}$ à cause de la contrainte de précédence

entre O_7 et O_8. Ensuite, les groupes ne possédant qu'une opération sont enlevés. La liste devient alors $[\{O_1, O_8\}, \{O_3, O_5\}]$.

La figure 9.1 montre les différents nœuds générés. Le nœud global qui décrit tout le problème est composé de l'ordonnancement de groupes original et de la liste de groupes calculée précédemment. La procédure de séparation est exécutée sur ce nœud, avec $\{O_1, O_8\}$ le nœud courant :

- $r_1 = 0, r_8 = 2$ à cause de la contrainte de précédence entre O_7 et O_8.
- $C_{\min} = \min(C_1, C_8) = \min(3, 4) = 3$.
- Comme $r_1 < r_8 < C_{\min}$, les deux opérations du groupe sont sélectionnées.
- Deux nœuds sont générés : le nœud 1 avec O_1 séquencé en premier et le nœud 2 avec O_8 séquencé en premier. Comme le groupe n'a plus qu'une opération sur les nœuds 1 et 2, le groupe est enlevé.

Ensuite, la procédure de séparation est effectuée sur le nœud 1, avec comme groupe courant $\{O_3, O_5\}$:

- $r_3 = 7, r_5 = 4$.
- $C_{\min} = \min(C_3, C_5) = \min(10, 7) = 7$.
- Comme $r_5 < C_{\min} \leq r_3$, seule O_5 est sélectionnée.
- Seul le nœud 1,1 est généré avec O_5 séquencée en premier. Tous les groupes de ce nœud ont une seule opération, donc ce nœud est solution.

Finalement, la procédure de séparation s'effectue sur le nœud 2. Comme pour le nœud 1,1, seul le nœud 2,1 est généré et est une solution. Une solution optimale de n'importe quel objectif régulier est soit l'ordonnancement du nœud 1,1, soit du 2,1.

9.1.2 Diminution de l'espace de recherche

9.1.2.1 Une condition suffisante pour diminuer l'espace de recherche

Pour diminuer l'espace de recherche, nous proposons une condition suffisante pour le séquencement complet du groupe courant tout en conservant la solution optimale.

La date de fin d'une opération interfère avec tout objectif régulier de deux manières :

- la date en elle-même, car la fonction objectif est une fonction des dates de fin des opérations ;
- en interférant avec les dates de fin des autres opérations, à cause des contraintes de précédence ou de ressources.

Ainsi, une condition suffisante au séquencement d'un groupe courant complet tout en conservant la solution optimale est :

- le séquencement ne dégrade pas la fonction objectif ;
- le séquencement n'interfère pas avec les dates de début au plus tôt des autres opérations non séquencées.

La dernière condition peut être exprimée sous la forme d'un problème à une machine en utilisant θ_i : les opérations doivent finir avant la date de début au

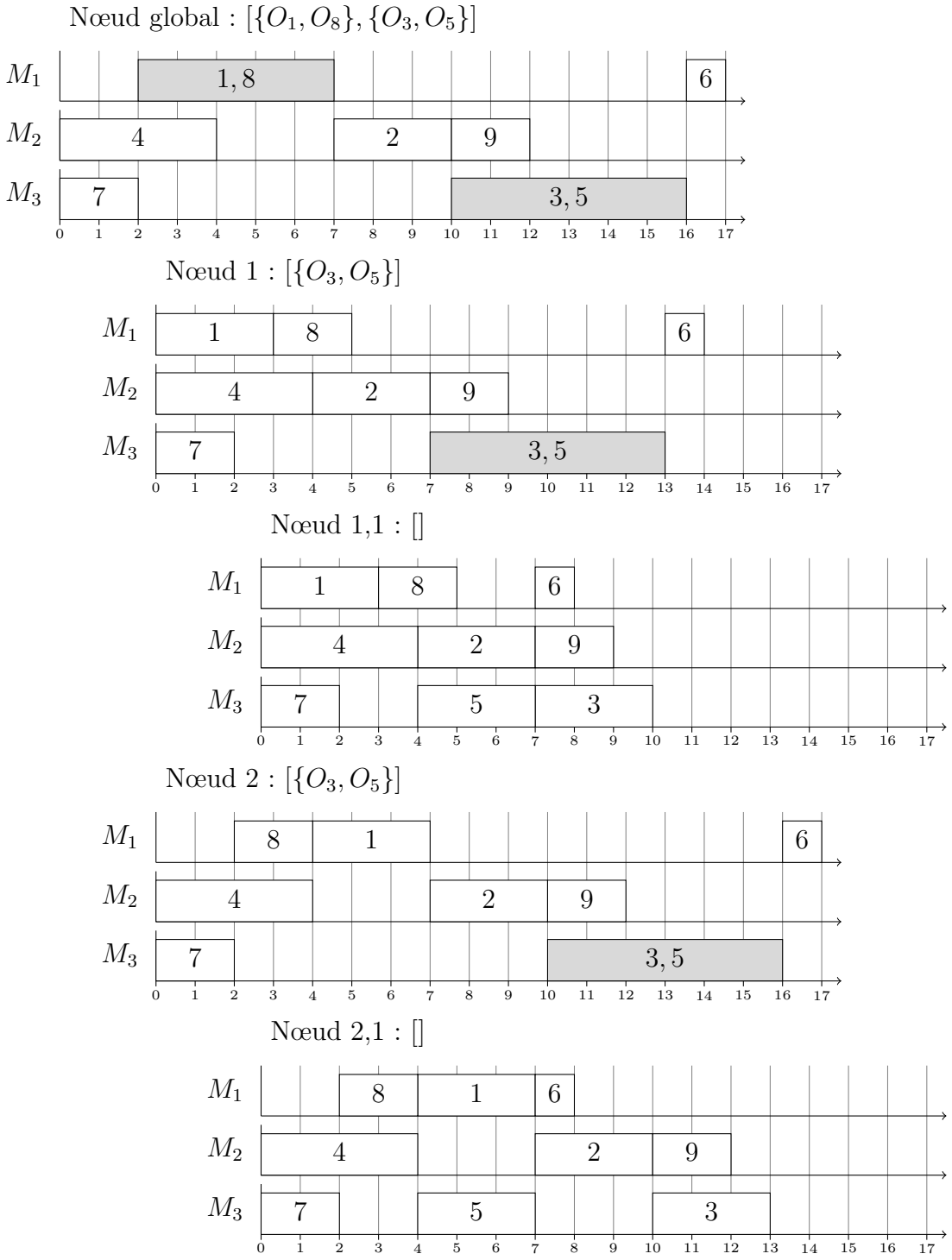


FIG. 9.1: Espace de recherche du problème présenté à la figure 3.1

plus tôt des opérations la succédant et avant la plus petite date de début au plus tôt des opérations dans le groupe suivant le groupe courant. Nous avons donc un problème à une machine avec des dates de début au plus tôt et des dates de fin impératives :

$$1|r_i, \tilde{d}_i|-, \forall O_i \in g_{\ell,k}, \tilde{d}_i = \min_{j|O_j \in \Gamma^+(O_i) \cup g_{\ell,k+1}} \theta_j$$

avec r_i la date de début au plus tôt de l'opération O_i dans l'ordonnancement de groupes.

Si une séquence pour le groupe courant est une solution de ce problème, et que les dates de fin des opérations séquencées ne dégrade pas la fonction objectif, alors ce groupe peut être séquencé tout en conservant la valeur optimale.

Avant d'appliquer la procédure de séparation à un nœud, une séquence satisfaisant la condition suffisante est recherchée. Si une telle séquence existe, alors le groupe courant est séquencé, est enlevé de la liste et un unique nœud est créé.

Grâce à cette condition suffisante, nous pouvons éviter la génération de nœuds, et ainsi, diminuer l'espace de recherche.

9.1.2.2 Exemple

Nous allons illustrer cette méthode de diminution de l'espace de recherche sur le nœud global de la figure 9.1. Tout d'abord, le problème à une machine correspondant au groupe $\{O_1, O_8\}$ est généré en utilisant (7.2).

$$r_1 = 0, \tilde{d}_1 = \min(\theta_2, \theta_6) = \min(4, 7) = 4$$
$$r_8 = 2, \tilde{d}_8 = \min(\theta_9, \theta_6) = \min(7, 7) = 7$$

La séquence O_1, O_8 résout ce problème. Si C_1 et C_8 ne sont pas utilisés dans le calcul de la fonction objectif, alors ce groupe peut être séquencé. Nous avons alors uniquement le nœud 1. Par exemple, si notre objectif est le *makespan*, C_1 et C_8 n'auront aucune influence sur ce critère car C_6 sera toujours supérieure à C_1 et C_8. Grâce à cette condition suffisante, il est possible de ne pas générer le nœud 2, et ainsi réduire l'espace de recherche.

9.1.3 Stratégies de recherche

La procédure de séparation génère des nœuds, mais l'ordre d'exploration de ces nœuds influe sur les performances de l'algorithme : si une meilleure solution est trouvée plus tôt, la borne supérieure sera meilleure, et le nombre de nœud visités sera plus petit.

Il y a deux méthodes pour explorer l'espace de recherche dans une procédure de séparation et évaluation : la recherche en profondeur, et la recherche des meilleures bornes en premier.

La recherche en profondeur va directement vers une solution : les nœuds sont traités par une pile. Il est très important de classer efficacement les nœuds nouvellement générés pour trouver une bonne solution rapidement. Dans ce mode

de fonctionnement, les nœuds sont classés en favorisant les nœuds possédant les meilleures bornes inférieures. En cas d'égalité, nous utilisons la règle de priorité SQUTAIL présenté à la section 8.1.2.

Une autre méthode de parcours consiste à traiter en premier les nœuds possédant la meilleure borne inférieure. Dans ce cas, seulement les nœuds nécessaires sont traités, sauf pour quelques nœuds possédant comme borne inférieure la qualité optimale, mais qui ne sont pas solutions. L'inconvénient de cette méthode est que le nombre de nœuds stockés prend énormément de mémoire. C'est pourquoi cette méthode est généralement utilisée avec les procédure de séparation générant seulement deux nœuds. La notre ne générant pas exclusivement deux nœuds, il est nécessaire de limiter le nombre de nœuds stockés. Dans un premier temps, une recherche en meilleure borne en premier est effectuée, c'est-à-dire que le nœud possédant la meilleure borne est sélectionnée pour appliquer la procédure de séparation. En cas d'égalité, le nœud dont l'ordonnancement de groupes possède le plus de groupes est sélectionné. Si le nombre de nœuds dépasse un certain seuil, le nœud sélectionné est résolu grâce à une recherche en profondeur.

9.2 Expérimentations

Cette section présente des expérimentations de la méthode exacte avec comme objectif le *makespan*. Différentes variantes sont testées pour évaluer l'impact des différents composants de cette méthode.

9.2.1 Protocole

Les instances utilisées pour ces expérimentations sont les mêmes que celles utilisées pour évaluer les heuristiques à la section 8.3.1. Notons que, par construction, la qualité optimale est connue. Ces expérimentations sont exécutées sur un Intel Pentium Xeon 2.60GHz.

Différentes variantes de la méthode exacte sont utilisées pour résoudre les instances :

- Défaut : Optimal OMP LB est utilisée comme borne inférieure, la condition suffisante est utilisée, et une recherche par meilleure borne en premier est réalisée jusqu'à ce que 1000 nœuds soient stockés (le programme utilise toujours moins de 100 MB de mémoire avec ces réglages pour les instances utilisées) ;
- Recherche en profondeur : mêmes paramètres que Défaut avec une recherche en profondeur ;
- JPS OMP LB : mêmes paramètres que Défaut avec la borne inférieure JPS OMP LB ;
- Pas de condition suffisante : mêmes paramètres que Défaut sans la condition suffisante.

9.2.2 Résultats

Les résultats de ces expérimentations sont présentés dans les tableaux 9.1 et 9.2. Pour chaque instance et chaque variante de la méthode exacte, le nombre de nœuds explorés pour trouver la solution optimale (NO), le nombre total de nœuds explorés (le nombre de nœuds pour trouver la solution optimale plus le nombre de nœud pour prouver son optimalité, NT) et le temps d'exécution sont présentés. Les lignes avec des étoiles correspondent aux instances qui ne sont pas résolues après 24 heures de calcul. Pour finir, les écarts moyens entre Défaut et les autres variantes sont donnés (les résultats en italiques ne sont pas utilisés pour calculer cette moyenne).

Les résultats des instances non résolues par Défaut après 24 heures sont exposés sur le tableau 9.3. Pour cela, la variante utilisée est Défaut avec 10000 nœuds stockés avant la recherche en profondeur. Pour chaque instance, la valeur optimale est donnée (comme l'algorithme exact pour le *job shop* n'a pas résolu optimalement l'instance la27, 1252 n'est qu'une borne supérieure). Ensuite, les résultats sur la borne inférieure sont donnés : la borne inférieure après 24 heures de calcul ainsi que le nombre de nœuds explorés pour trouver cette valeur sont présentés. Ensuite, les résultats sur la borne supérieure sont exposés. Finalement, le nombre de nœuds explorés après 24 heures de calcul est présenté.

9.2.3 Discussion

9.2.3.1 Analyse de la variante Défaut

La méthode exacte trouve la solution optimale pour la plupart des instances en moins de 24 heures. La variante Défaut est la plus efficace sur ces instances : elle obtient de meilleures performances que les autres variantes pour presque toutes les instances.

Les instances à 5 machines sont toutes résolues en moins d'une seconde. L'algorithme est très efficace sur les petites instances. Pour les petits systèmes de production, il est envisageable d'utiliser cet algorithme même dans des conditions nécessitant des temps de réponse faibles.

Les résultats pour les instances de taille 30×10 sont également très bons. Ceci est possible grâce à la qualité de la borne inférieure, et à la facilité de trouver une bonne borne supérieure, même avec une heuristique simple, pour ces instances.

La plupart du temps, le nombre total de nœuds pour trouver la solution optimale est égale au nombre total de nœuds explorés. Ce résultat s'explique par la rapidité à laquelle la borne inférieure correspond très vite à la qualité optimale. Cependant, la24, la36 et la38 sont des exceptions. Sur ces instances, la borne supérieure correspond rapidement à la qualité optimale, et prouver l'optimalité est coûteux sur ces instances.

Taille	Inst.	Défaut			Recherche en profondeur		
		NO	NT	temps (s)	NO	NT	temps (s)
10×5	la01	31	31	0.048	31	70	0.095
	la02	105	105	0.132	264	277	0.323
	la03	31	31	0.052	31	74	0.117
	la04	30	30	0.045	28	61	0.090
	la05	1	1	0.008	1	1	0.008
15×5	la06	1	1	0.018	1	1	0.018
	la07	43	43	0.140	43	131	0.399
	la08	95	95	0.258	179	246	0.637
	la09	44	44	0.130	44	124	0.353
	la10	1	1	0.016	1	1	0.016
20×5	la11	1	1	0.028	1	1	0.029
	la12	1	1	0.031	1	1	0.032
	la13	1	1	0.026	1	1	0.026
	la14	1	1	0.027	1	1	0.026
	la15	1	1	0.028	1	1	0.028
10×10	la16	4863	4863	12.789	97	1638	4.480
	la17	6232	6232	17.073	5360	10227	24.634
	la18	18611	18611	51.519	637	2324	5.504
	la19	1649	1649	4.860	693	1785	4.596
	la20	1998	1998	5.668	723	2159	5.296
15×10	la21	1967	1967	11.013	2945	3069	14.645
	la22	200	200	1.056	391	454	2.216
	la23	2178	2178	11.377	956027	956072	4314.690
	la24	20490	174539	803.026	81070	188952	829.016
	la25	124	124	0.608	14397	14417	61.796
20×10	la26	2157	2157	23.513	7617	7731	65.808
	la27	*	*	*	*	*	*
	la28	5496	5496	39.931	512799	513001	3826.920
	la29	*	*	*	*	*	*
	la30	*	*	*	*	*	*
30×10	la31	183	183	3.244	183	502	8.733
	la32	198	198	3.696	198	572	10.377
	la33	202	202	3.580	202	548	9.193
	la34	195	195	3.444	195	531	8.929
	la35	175	175	2.936	175	415	6.848
15×15	la36	*31367*	*1046665*	*69068.800*	*	*	*
	la37	*	*	*	*	*	*
	la38	4441	1633487	13074.800	4000518	4086855	285217.000
	la39	8662	28893	206.325	142705	169671	1181.990
	la40	1065	1065	8.785	59037	59077	393.093
Écart moyen		–	–	–	46.315	20.839	18.936

La taille est notée $n \times m$ avec n le nombre de travaux et m le nombre de machines.

TAB. 9.1: Résultats pour la méthode exacte 1

Taille	Inst.	JPS OMP LB			Pas de condition suffisante		
		NO	NT	temps (s)	NO	NT	temps (s)
10×5	la01	31	31	0.047	34	34	0.054
	la02	105	105	0.124	112	112	0.140
	la03	31	31	0.050	35	35	0.058
	la04	45	45	0.062	38	38	0.056
	la05	1	1	0.008	1	1	0.008
15×5	la06	1	1	0.018	1	1	0.018
	la07	43	43	0.132	59	59	0.170
	la08	139	139	0.366	134	134	0.367
	la09	44	44	0.125	57	57	0.167
	la10	1	1	0.016	1	1	0.016
20×5	la11	1	1	0.028	1	1	0.027
	la12	1	1	0.031	1	1	0.032
	la13	1	1	0.025	1	1	0.025
	la14	1	1	0.027	1	1	0.027
	la15	1	1	0.027	1	1	0.028
10×10	la16	12131	12131	31.286	12885	12885	31.010
	la17	29410	39835	98.350	76714	194161	490.559
	la18	3634	5784	16.165	18106	18106	47.663
	la19	3464	3464	10.061	2489	2489	7.452
	la20	8958	8958	26.578	2166	2166	6.368
15×10	la21	2687	2687	13.717	4449	4449	23.966
	la22	279	279	1.480	320	320	1.752
	la23	19033	19033	87.846	5790	5790	30.086
	la24	28594	295465	1405.280	236772	872319	4079.890
	la25	2488	2488	10.801	151	151	0.788
20×10	la26	41705	41705	292.466	4434	4434	42.139
	la27	*	*	*	*	*	*
	la28	1592222	1592260	10939.600	87842	87912	644.648
	la29	*	*	*	*	*	*
	la30	*	*	*	*	*	*
30×10	la31	183	183	3.208	231	231	4.140
	la32	198	198	3.732	231	231	4.684
	la33	203	203	3.544	231	231	4.260
	la34	195	195	3.424	230	230	4.176
	la35	174	174	2.852	224	224	3.816
15×15	la36	*	*	*	*	*	*
	la37	*	*	*	*	*	*
	la38	450407	8441712	64071.900	1318	3178686	23949.500
	la39	29589	44659	322.208	39926	97507	654.733
	la40	165326	165354	1079.240	2080	2080	16.337
Écart moyen		17.200	14.457	12.752	1.439	1.802	1.716

La taille est notée $n \times m$ avec n le nombre de travaux et m le nombre de machines.

TAB. 9.2: Résultats pour la méthode exacte 2

Taille	Inst.	Opt.	BI	Nœuds	BS	Nœuds	Tot. Nœuds
20×10	la27	1252*	1235	0	1279	5150695	9500000
20×10	la29	1202	1202	3836	1221	10343	10000000
20×10	la30	1355	1355	0	1359	2911199	12500000
15×15	la37	1397	1397	2	1412	7623146	9700000

BI : borne inférieure, BS : borne supérieure.

TAB. 9.3: Résultats pour les instances difficile après 24 heures de calcul

9.2.3.2 Comparaison des différentes variantes

En comparant les résultats des variantes Défaut et Recherche en profondeur, nous pouvons étudier l'impact de la stratégie de recherche sur les performances de l'algorithme. En moyenne, l'écart est d'environ 19, c'est-à-dire qu'utiliser une recherche en profondeur prend 20 fois plus de temps que d'utiliser le parcours des meilleures bornes. Cette dernière méthode d'exploration de l'espace de recherche réduit donc considérablement l'espace de recherche. Ceci est dû au fait que de mauvais choix ne sont pas réalisés au début de la procédure, et que donc de bonnes solutions sont très rapidement atteintes. Cependant, pour les instances la16 et la17, la recherche en profondeur est plus efficace, car la solution optimale est trouvée très vite par la recherche en profondeur.

Utiliser Optimal OMP LB donne toujours de meilleurs résultats que JPS OMP LB. En effet, utiliser une meilleure borne permet de réduire l'espace de recherche, et donc de réduire le nombre de nœuds parcourus. Par contre, cet avantage doit être contre balancé par le temps de calcul requis pour calculer la borne inférieure. Dans notre cas, l'écart entre le calcul des deux bornes est très faible, et utiliser la meilleure borne est profitable : l'écart moyen des temps d'exécution est de plus de 14.

La condition suffisante permet également de diminuer l'espace de recherche. Le surplus de calcul est rentable : l'écart moyen est d'environ 1,7. Bien que ce soit bien moins important que la réduction dûe à la borne inférieure, la réduction est dans certains cas très importante, comme pour la17 où l'écart est de plus de 27.

9.2.3.3 Instances difficiles

Certaines instances, dont les résultats sont présentés dans le tableau 9.3, ne sont pas résolus après une journée par la variante Défaut.

Les instances la30 et la37 possèdent une borne inférieure égale à la qualité optimale dès le début de la procédure. La borne inférieure égale à la qualité optimale est trouvée plus tard pour la29, mais tout de même avant la première recherche en profondeur. Pour ces instances, la principale difficulté est de trouver une solution optimale au problème.

Pour la27, la recherche d'une solution optimale est également difficile. Par contre, comme nous ne connaissons pas la solution optimale pour la27, nous ne pouvons savoir si la borne inférieure égale à la qualité optimale est obtenue rapidement, ou si l'obtenir est difficile.

9.3 Conclusion

Dans, ce chapitre, une méthode exacte pour le meilleur des cas dans un ordonnancement de groupes est proposée. Cette méthode est générique : elle peut être utilisée pour tout objectif régulier, juste en donnant une borne inférieure adaptée à l'objectif voulu.

Les résultats obtenus sont encourageants. La méthode est très efficace pour les petites instances, et relativement rapide pour les instances de tailles moyennes. Certaines instances restent tout de même non résolues après une journée de calcul. La recherche de la borne inférieure optimale est très efficace. La plupart du temps, la difficulté est de trouver une solution optimale, et non de prouver son optimalité.

Cette méthode exacte possède donc des performances lui permettant d'être utilisable en pratique dans de nombreux cas. Le chapitre suivant étudie ces différents cas, et, plus généralement, aborde l'utilisation des algorithmes traitant le meilleur des cas dans une méthode d'ordonnancement utilisant l'ordonnancement de groupes.

Chapitre 10

Utilisation du meilleur des cas

L'évaluation du meilleur des cas dans un ordonnancement de groupes peut être utilisée dans deux contextes différents : durant la phase proactive, et durant la phase réactive. Les contraintes de qualité et de temps d'exécution sont différentes dans ces deux contextes d'utilisation.

10.1 Utilisation dans la phase proactive

Durant la phase proactive, l'évaluation du meilleur des cas dans un ordonnancement de groupes est une information pour l'humain sur la qualité d'un ordonnancement de groupes comme proposée par [Ess03]. Cette mesure n'étant pas contradictoire avec la flexibilité et la qualité dans le pire des cas, il n'est pas nécessaire de l'inclure dans le processus d'optimisation. Le temps de calcul disponible pour effectuer l'évaluation dans le meilleur des cas est alors important.

10.1.1 Utilisation de la qualité optimale

Comme du temps est disponible et qu'une unique évaluation est nécessaire, utiliser la méthode exacte semble être un bon choix, surtout pour les instances de taille respectable. Le décideur possédera ainsi la meilleure qualité qu'il est possible d'avoir avec un ordonnancement de groupes.

Lorsque les problèmes sont de grande taille, il n'est par contre pas envisageable d'obtenir la qualité exacte dans le meilleur des cas.

10.1.2 Utilisation des bornes inférieures et supérieure

Pour les grandes instances, il semble alors logique d'utiliser une borne inférieure et une heuristique, qui donnerait ainsi un intervalle de la qualité dans le meilleur des cas. Utiliser l'heuristique SB et la borne inférieure Optimal OMP LB donnerait un intervalle de petite taille.

Mais la méthode exacte peut être utilisée dans un tel cas. En effet, à tout moment, la méthode exacte est capable de fournir la borne inférieure et la borne supérieure du problème. Il est alors possible de limiter le temps de calcul, et d'obtenir soit la qualité exacte si elle est trouvée durant le temps imparti, soit un intervalle sur la qualité qui sera meilleur que l'intervalle donné par une heuristique et une borne inférieure seules.

Par contre, la méthode exacte pourrait utiliser les connaissances disponibles lors de la génération de l'ordonnancement de groupes.

10.1.3 Adaptation au contexte de la phase proactive

L'algorithme EBJG (voir la section 3.2.4.2) utilisé dans la phase proactive commence avec un ordonnancement classique de qualité connu. Comme le fait remarquer [Ess03], la qualité de cet ordonnancement est une borne supérieure à la qualité dans le meilleur des cas de l'ordonnancement de groupes final, et cet ordonnancement fait partie des ordonnancements décrits par cet ordonnancement de groupes. Ainsi, il est possible d'utiliser cet ordonnancement comme borne supérieure initiale pour diminuer l'espace de recherche.

La méthode exacte est donc un outil efficace permettant une évaluation de la qualité dans le meilleur des cas durant la phase proactive, même pour des problèmes de taille conséquente.

La phase réactive est par contre bien différente, notamment au niveau du temps disponible pour effectuer les calculs.

10.2 Utilisation dans la phase réactive

La phase réactive est très différente de la phase proactive. Tout d'abord, le temps disponible pour effectuer des calculs est beaucoup plus court. Ensuite, les calculs à réaliser sont beaucoup plus nombreux : il faut une évaluation pour chaque opération exécutable.

La structure du problème est également différente. En effet, les différents problèmes à résoudre sont proches les un des autres, car seule une opération est séquencée différement. Cela implique deux choses : il est possible d'utiliser cette similitude pour résoudre plus rapidement les problèmes, et l'évaluation doit être précise, vu que les différentes qualités dans le meilleur des cas risquent d'être similaires.

10.2.1 Utilisation des bornes inférieure et supérieure

L'utilisation de bornes inférieure et supérieure nous semble peu adaptée à ce problème. En effet, n'obtenir que des bornes risque de rendre l'utilisation de cet indicateur inutilisable à cause du grand nombre de valeurs similaires dont on ne

peut dire si l'une est meilleure que l'autre. De plus, cette représentation est plus dificile à manipuler pour l'humain.

Malheureusement, comme l'évaluation du meilleur des cas est un problème NP-difficile, et dont il n'existe pas d'algorithme exact efficace pour toute instance de grande taille, l'utilisation de bornes inférieure et supérieure est le seul résultat qui peut être garanti.

Bien que cette représentation du meilleur des cas ne soit que peu adaptée au problème, c'est la seule représentation dont on peut garantir l'obtention en un temps imparti.

10.2.2 Utilisation de la qualité optimale

La qualité optimale est donc l'indicateur qui serait le plus intéressant.

L'utilisation de la méthode exacte est envisageable pour les petits problèmes d'ordonnancement, avec cinq machines par exemple. En effet, notre méthode exacte prend moins d'une seconde pour tout problème de cette taille. Ainsi, cet indicateur est utilisable pour notre expérimentation, puisque notre chaîne ne possède que 6 machines.

10.2.3 Adaptation au contexte de la phase réactive

Comme les différents problèmes à résoudre durant la phase réactive sont des problèmes très proches, il est possible d'améliorer notre méthode exacte en utilisant ces spécificités. Ainsi, de meilleurs bornes seront disponibles, voire les solutions exactes seront plus souvent atteintes.

Comme la principale faiblesse de notre méthode exacte est de trouver une solution de qualité optimale, utiliser une heuristique efficace permettrait de diminuer l'espace de recherche. Comme les problèmes sont similaires, il est facile de réaliser une heuristique utilisant la meilleure solution trouvée jusqu'à présent pour notre problème, mais ne satisfaisant pas forcément les contraintes de l'ordonnancement de groupes que l'on tente de résoudre. L'heuristique modifierait alors la meilleure solution trouvée, pour obtenir une solution de bonne qualité décrite par l'ordonnancement de groupes. Grâce à cette technique, il serait possible d'obtenir de très bons résultats en peu de temps.

Une autre piste envisageable est de mutualiser l'espace de recherche aux différents problèmes à résoudre. En effet, les différents problèmes sont suffisamment proches pour partager une partie de leur espace de recherche, et ainsi diminuer l'espace global de recherche. Cette technique serait applicable pour les différentes évaluations à faire en même temps à un moment donné, mais également pour tous les problèmes rencontrés tout au long de l'exécution de l'ordonnancement.

10.3 Conclusion

L'évaluation du meilleur des cas peut dès à présent être utilisée pour les problèmes de petites tailles, et ce de manière exacte. Nous allons d'ailleurs l'utiliser pour nos expérimentations décrites à la section 6.3.

Pour les instances de grandes tailles, seules les bornes inférieure et supérieure de la qualité dans le meilleur des cas sont envisageables. Une étude serait nécessaire pour évaluer l'intérêt d'une telle représentation de la qualité pour l'humain. Des spécialisations de la méthode exacte sont envisagées pour améliorer ses performances pour l'utilisation durant les phases proactive et réactive.

Conclusions et perspectives

Après avoir présenté l'ordonnancement d'atelier et l'ordonnancement sous incertitudes, nous avons choisi de fonder nos travaux sur l'ordonnancement de groupes. L'algorithme le plus efficace permettant de générer un ordonnancement de groupes semble être l'algorithme EBJG proposé par Esswein dans [Ess03]. Cependant, il peut être amélioré. Ainsi, les modifications que nous avons réalisées diminuent les temps de calculs. Nous proposons également de l'étendre pour générer un ensemble de solutions non dominées, plutôt que d'utiliser une méthode epsilon-contrainte. Nous avons également implémenté l'ordonnancement de groupes sur une chaîne de production flexible. Cette implémentation nous a permis d'évaluer le potentiel de robustesse de l'ordonnancement de groupes. Les résultats sont très prometteurs.

Ensuite, nous avons étudié les différents systèmes homme-machine pour l'ordonnancement de groupes. Ils sont constitués de deux ensembles : les systèmes pour la phase prédictive et les systèmes pour la phase réactive.

Pour la phase prédictive, deux système homme-machine existent : ORABAID et l'approche proposée par Esswein. Cette dernière nous semble bien plus efficace et flexible. Pour améliorer la phase prédictive, nous proposons d'utiliser un ensemble de solutions non dominées plutôt qu'une approche epsilon-contrainte afin que l'humain n'ait pas à donner de paramètres à l'algorithme. Il peut alors choisir le compromis entre flexibilité et qualité parmi un ensemble de solutions.

Seul ORABAID propose une phase réactive pour l'ordonnancement de groupes, basée sur la marge libre séquentielle. Après une analyse du système homme-machine, nous proposons une nouvelle approche, utilisant un ensemble d'indicateurs pour aider l'humain à prendre sa décision. Ce nouveau système homme-machine devrait rendre l'humain plus actif dans le processus d'ordonnancement. Cette hausse d'activité permettra d'augmenter les performances du système. Elle permettra également de rendre l'humain plus impliqué et plus attentif. Nous exposons ensuite le protocole expérimental pour évaluer ce système homme-machine.

Dans ce système homme-machine, une évaluation du meilleur des cas est nécessaire. C'est pourquoi nous proposons une étude de ce problème. Nous présentons une méthode pour réaliser une borne inférieure pour tout objectif régulier. Nous exposons également une borne inférieure spécifique au *makespan*, reposant sur une relaxation de l'ordonnancement de groupes en *one machine problem*.

Cette relaxation nous permet d'adapter l'heuristique du *shifting bottleneck* à l'ordonnancement de groupes. Des règles de priorités sont également proposées.

Notre variante du *shifting bottleneck* donne de très bons résultats pour le *makespan*, tandis que les règles de priorité permettent d'obtenir une solution de qualité correcte en très peu de temps.

Pour compléter l'étude du meilleur des cas dans un ordonnancement de groupes, une méthode exacte est proposée. Cette méthode est une procédure de séparation et évaluation. Basée sur l'énumération des ordonnancements actifs, elle est adaptable à tout objectif régulier. Pour diminuer l'espace de recherche, nous proposons une condition suffisante au séquencement d'un groupe sans perdre la solution optimale du problème. Les expérimentations sur le *makespan* montrent que cette méthode exacte est très efficace sur les petites instances. Elle est donc utilisable pour des petits problèmes (cinq machines ou moins). Nous analysons également l'utilisation de la méthode exacte pour les phases prédictive et réactive de l'ordonnancement de groupes, et proposons des pistes pour spécialiser nos algorithmes à ces cas particuliers.

Les premiers résultats ont montré tout l'intérêt de cette méthode exacte. Ces premières expérimentations nous ont également conduit à réfléchir à de prochaines améliorations.

La procédure de séparation et évaluation peut ainsi être améliorée en lui intégrant la condition suffisante présentée dans ce mémoire. Cette modification ne devrait pas détériorer la généricité de la méthode, car elle sera toujours adaptée à tout objectif régulier. Nous espérons ainsi réduire encore l'espace de recherche, afin de diminuer le nombre de nœuds explorés et trouver plus rapidement la solution optimale.

La méthode exacte peut également être optimisée pour son utilisation dans la phase réactive. Nous pouvons utiliser le fait que les problèmes à résoudre durant cette phase sont très proches les uns des autres. La première adaptation à ce contexte spécifique serait d'utiliser les résolutions précédentes pour trouver, de manière heuristique, une première solution efficace satisfaisant les contraintes du problème à résoudre. Ainsi, une borne supérieure de très bonne qualité serait disponible dès le début de la procédure, ce qui diminuerait fortement l'espace de recherche. Une autre adaptation possible serait effectuée directement au niveau de l'espace de recherche. L'objectif serait de mettre en commun les espaces de recherche des différents problèmes à résoudre simultanément pour diminuer l'espace de recherche total.

De nouvelles heuristiques peuvent être créées en utilisant la relaxation en *one machine problem*, mais de manière plus légère que le *shifting bottleneck*. En effet, le travail de réoptimisation de cette heuristique est beaucoup plus lourd que dans l'algorithme original. Limiter les réoptimisations devrait permettre d'obtenir des heuristiques légères, possédant des performances meilleures que les règles de priorité pour des temps de calcul comparables.

Pour évaluer notre nouveau système homme-machine pour la phase réactive, nous proposons un protocole expérimental complet à la section 6.3. Les résultats

de cette expérimentation vont nous ouvrir de nouvelles perspectives. Cette expérimentation est planifiée pour l'année universitaire 2008–2009, et sera réalisée en collaboration avec Clément Guérin qui a commencé son doctorat en psychologie en 2008 sous la direction de Jean-Michel Hoc. Elle permettra d'analyser les différences entre le système homme-machine d'ORABAID et celui que nous proposons. Elle nous permettra également d'analyser les points forts et les points faibles de notre système homme-machine. Nous pourrons alors l'améliorer en prenant en compte les résultats obtenus.

Bibliographie

[ABE05] Christian Artigues, Jean-Charles Billaut, and Carl Esswein. Maximization of solution flexibility for robust shop scheduling. *European Journal of Operational Research*, 165(2) :314–328, September 2005.

[ABZ88] Joseph Adams, Egon Balas, and Daniel Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3) :391–401, 1988.

[Alo02] Mohamed Ali Aloulou. *Structure flexible d'ordonnancements à performances contrôlées pour le pilotage d'atelier en présence de perturbations*. Thèse de doctorat, Institut National Polytechnique de Lorraine, 2002.

[AP04] Mohamed Ali Aloulou and Marie-Claude Portmann. Une approche proactive réactive efficace, cas d'un atelier d'assemblage. In Jean-Charles Billaut, Aziz Moukrim, and Eric Sanlaville, editors, *Flexibilité et robustesse en ordonnancement*, Traité IC2, pages 243–261. Hermes Science, Paris, December 2004.

[Art97] Christian Artigues. *Ordonnancement en temps réel d'ateliers avec temps de préparation des ressources*. Thèse de doctorat, Université Paul Sabatier, Toulouse, 1997.

[BC01] Gérard Bel and Jean-Bernard Cavallé. Approche simulatoire. In Pierre Lopez and François Roubellat, editors, *Ordonnancement de la production*, Traité IC2, pages 169–195. Hermes Science, Paris, January 2001.

[BHLL04] Cyril Briant, Marie-José Huguet, Hoang Trung La, and Pierre Lopez. Approches par contraintes pour l'ordonnancement robuste. In Jean-Charles Billaut, Aziz Moukrim, and Eric Sanlaville, editors, *Flexibilité et robustesse en ordonnancement*, Traité IC2, pages 191–218. Hermes Science, Paris, December 2004.

[Bil93] Jean-Charles Billaut. *Prise en compte des ressources multiples et des temps de préparation dans les problèmes d'ordonnancement en temps réel*. Thèse de doctorat, Université Paul Sabatier, Toulouse, 1993.

[BJS94] Peter Brucker, Bernd Jurisch, and Bernd Sievers. A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics*, 49(1-3) :107–127, 1994.

[BK07] Peter Brucker and Sigrid Knust. Complexity results for scheduling problems. `http://www.mathematik.uni-osnabrueck.de/research/OR/class/`, 2007. [online ; retrieved on 2008-06-18].

[BMS04a] Jean-Charles Billaut, Aziz Moukrim, and Eric Sanlaville, editors. *Flexibilité et robustesse en ordonnancement*. Traité IC2. Hermes Science, Paris, December 2004.

[BMS04b] Jean-Charles Billaut, Aziz Moukrim, and Eric Sanlaville. Introduction. In Jean-Charles Billaut, Aziz Moukrim, and Eric Sanlaville, editors, *Flexibilité et robustesse en ordonnancement*, Traité IC2, pages 15–34. Hermes Science, Paris, December 2004.

[BOB08] Cyril Briand, S. Ourari, and B. Bouzouia. Une approche coopérative pour l'ordonnancement sous incertitudes,. In *Actes de la 7ème Conférence Francophone de Modélisation et Simulation (MOSIM'08)*, 2008.

[BPH82] J. H. Blackstone, D. H. Phillips, and G. L. Hogg. A state of the art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research*, 20 :27–45, 1982.

[BR96] Jean-Charles Billaut and François Roubellat. A new method for workshop real-time scheduling. *International Journal of Production Research*, 34(6) :1555–1579, 1996.

[Bre00] R. W. Brennan. Performance comparison and analysis of reactive and planning-based control architectures for manufacturing. *Robotics and Computer-Integrated Manufacturing*, 16(2-3) :191–200, April 2000.

[Bru98] Peter Brucker. *Scheduling algorithms*. Springer, Heidelberg, 2nd edition, 1998.

[Car82] Jacques Carlier. The one-machine sequencing problem. *European Journal of Operational Research*, 11(1) :42–47, September 1982.

[Ceg04] Julien Cegarra. *La gestion de la complexité dans la planification : le cas de l'ordonnancement*. Thèse de doctorat, Université de Paris 8, 2004.

[Cegss] Julien Cegarra. A cognitive typology of scheduling situations : a contribution to laboratory and field studies. *Theoretical Issues in Ergonomics Science*, in press.

[CHss] Julien Cegarra and Jean-Michel Hoc. The role of algorithm and result comprehensibility of automated scheduling on complacency. *Human Factors and Ergonomics in Manufacturing*, in press.

[CP89] Jacques Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35(2) :164–176, 1989.

[CP90] Jacques Carlier and E. Pinson. A practical use of jackson's preemptive schedule for solving the job shop problem. *Annals of Operations Research*, 26(1-4) :269–287, 1990.

[CP94] Jacques Carlier and E. Pinson. Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research*, 78(2) :142–147, October 1994.

[DB00] Andrew J. Davenport and J. Christopher Beck. A survey of techniques for scheduling with uncertainty, 2000. Manuscrit non publié, disponible à `http://www.eil.utoronto.ca/profiles/chris/chris.papers.html`.

[Dim05] Christos Dimopoulos. A novel approach for the solution of the multiobjective cell-formation problem. In *Proceedings of the International Conference of Production Research (ICPR 05)*, 2005.

[ER89] Jacques Erschler and François Roubellat. An approach for real time scheduling for activities with time and resource constraints. In R. Slowinski and J. Weglarz, editors, *Advances in project scheduling*. Elsevier, 1989.

[Ess03] Carl Esswein. *Un apport de flexibilité séquentielle pour l'ordonnancement robuste*. Thèse de doctorat, Université François Rabelais Tours, 2003.

[GLLRK79] Ronald L. Graham, Eugene L. Lawler, Jan Karel Lenstra, and A. G. H. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling : a survey. *Annals of Discrete Mathematics*, 5 :287–326, 1979.

[Gol97] Eliyahu M. Goldratt. *Critical Chain*. North River Press, 1997.

[GPS97] Steven M. Glover, Douglas F. Prawitt, and Brian C. Spilker. The influence of decision aids on user behavior : Implications for knowledge acquisition and inappropriate reliance. *Organizational Behavior and Human Decision Processes*, 72(2) :232–255, 1997.

[Gro02] Groupe flexibilité du GOThA. Flexibilité et robustesse en ordonnancement. *Bulletin de la ROADEF*, 8 :10–12, 2002.

[HA99] Jean-Michel Hoc and René Amalberti. Analyse des activités cognitives en situation dynamique : d'un cadre théorique à une méthode. *Le Travail Humain*, 62 :97–130, 1999.

[Hau89] R. Haupt. A survey of priority rule-based scheduling. *OR Spektrum*, 11 :3–16, 1989.

[Joh54] Selmer Martin Johnson. Optimal two and three stage production schedules with setup time included. *Naval Research Logistics Quarterly*, 1 :61–68, 1954.

[La05] Hoang Trung La. *Utilisation d'ordres partiels pour la caractérisation de solution robustes en ordonnancement*. Thèse de doctorat, INSA, Toulouse, 2005.

[Law73] Eugene L. Lawler. Optimal sequencing of a single machine subject to precedence constraints. *Management Science*, 19(5) :544–546, January 1973.

[Law84] S. Lawrence. Resource constrained project scheduling : an experimental investigation of heuristic scheduling techniques (supplement). Technical report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

[LG89] A. Le Gall. *Un système interactif d'aide à la décision pour l'ordonnancement et le pilotage en temps réel d'atelier*. Thèse de doctorat, Université Paul Sabatier, Toulouse, 1989.

[LRKB77] Jan Karel Lenstra, A. H. G. Rinnooy Kan, and Peter Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1 :343–362, 1977.

[PCM07] Guillaume Pinot, Olivier Cardin, and Nasser Mebarki. A study on the group sequencing method in regards with transportation in an industrial FMS. In *Proceedings of the IEEE SMC 2007 International Conference*, 2007.

[PM97] H. Pierreval and N. Mebarki. Dynamic selection of dispatching rules for manufacturing system scheduling. *International Journal of Production Research*, 35(6) :1575–1591, 1997.

[PM08a] Guillaume Pinot and Nasser Mebarki. Best-case lower bounds in a group sequence for the job shop problem. In *Proceedings of the 17th IFAC World Congress*, 2008.

[PM08b] Guillaume Pinot and Nasser Mebarki. Heuristiques pour le meilleur des cas dans un ordonnancement de groupes. In *Actes de la 7ème Conférence Francophone de Modélisation et Simulation (MOSIM'08)*, 2008.

[PMH08] Guillaume Pinot, Nasser Mebarki, and Jean-Michel Hoc. Coopération homme-machine pour la mise en œuvre d'un ordonnancement de groupes. In *Conférence Internationale Francophone d'Automatique, CIFA 2008*, 2008.

[PMre] Guillaume Pinot and Nasser Mebarki. An exact method for finding the best case in a group sequence : Application to a general shop problem. *European Journal of Operational Research*, en relecture.

[RBV95] François Roubellat, Jean-Charles Billaut, and Michel Villaumie. Ordonnancement d'atelier en temps réel : d'ORABAID à ORDO. *Revue d'automatique et de productique appliquées*, 8(5) :683–713, 1995.

[RBW88] Emilie M. Roth, Kevin B. Bennett, and David D. Woods. Human interaction with an "intelligent" machine. In Giuseppe Mancini, David D. Woods, and Erik Hollnagel, editors, *Cognitive Engineering in*

Complex Dynamic Worlds, pages 23–69. Academic Press, London, 1988.

[San89] Penelope M. Sanderson. The human planning and scheduling role in advanced manufacturing systems : an emerging human factors domain. *Human Factors*, 31(6) :635–666, 1989.

[SBD94] K. Swanson, J. Bresina, and M. Drummond. Just-in-case scheduling for automatic telescopes. In W. Buntine and D. H. Fisher, editors, *Knowledge-Based Artificial Intelligence Systems in Aerospace and Industry*, pages 10–19, 1994.

[SSJ+94] Penelope M. Sanderson, Jay Scott, Tom Johnston, John Mainzer, Larry Watanabe, and Jeff James. MacSHAPA and the enterprise of exploratory sequential data analysis (ESDA). *International Journal of Human-Computer Studies*, 41 :633–681, 1994.

[Tho80] Véronique Thomas. *Aide à la décision pour l'ordonnancement d'atelier en temps réel*. Thèse de doctorat, Université Paul Sabatier, Toulouse, 1980.

[VAL96] Rob J.M. Vaessens, Emile H. L. Aarts, and Jan Karel Lenstra. Job shop scheduling by local search. *INFORMS Journal on Computing*, 8(3) :302–317, 1996.

[vWCH08] Wout van Wezel, Julien Cegarra, and Jean-Michel Hoc. Allocating functions to human and algorithm in scheduling, 2008.

[WBS99] S. David Wu, Eui-Seok Byeon, and Robert H. Storer. A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness. *Operations Research*, 47(1) :113–124, 1999.

Résumé

La plupart des travaux en ordonnancement repose sur un modèle déterministe, peu adapté à la réalité de l'ordonnancement d'atelier. En effet, les ateliers de production sont soumis à un certain nombre d'incertitudes. C'est pourquoi l'ordonnancement sous incertitudes est un domaine en pleine expansion.

D'autre part, l'humain n'est généralement pas pris en compte dans l'élaboration de la méthode d'ordonnancement. Pourtant, l'humain joue un rôle central dans le processus d'ordonnancement, et ses connaissances du terrain sont précieuses. C'est pourquoi nous pensons que des systèmes homme-machine efficaces sont nécessaires au bon fonctionnement des méthodes d'ordonnancement d'atelier.

Pour cela, nous nous reposons sur l'ordonnancement de groupes. Cette méthode d'ordonnancement d'atelier comporte différents avantages pour notre recherche : c'est une méthode d'ordonnancement sous incertitudes et sa structure est facilement manipulable par l'humain. Nous étudions les systèmes homme-machine existant pour cette méthode d'ordonnancement. Nous proposons ensuite un nouveau système homme-machine, afin d'améliorer la coopération. Dans ce système, nous utilisons la qualité dans le meilleur des cas dans un ordonnancement de groupes. Comme ce thème n'est pas encore abordé dans la littérature, nous proposons des bornes inférieures, des heuristiques et une méthode exacte pour résoudre ce problème.

Cooperation between human and machine for scheduling under uncertainties

Most of the research in scheduling is based on determinism models, not adapted to the reality of shop scheduling. Indeed, manufacturing systems are subject to uncertainties. This is why scheduling under uncertainties is a booming field.

Moreover, humans are not taken into account in most of the scheduling methods. However, humans have a central role in the scheduling process, and their knowleadges are precious. This is why we think that efficient human-machine systems is indispensible for an efficient scheduling process.

To achieve this goal, group sequencing is used. This scheduling method has different advantages : it is made to manage the uncertainties present in the shop floor, and its structure is easy to use by the human. We study existing human-machine systems for group sequencing. Then, we propose a new system to improve the cooperation between human and machine. In this new human-machine system, we use the best case quality in a group sequence. Because this subject is not studied by the literature, we propose lower bounds, heuristics and an exact method to solve this problem.

Mots clés : Ordonnancement, coopération homme-machine, incertitudes, flexibilité, robustesse, optimisation combinatoire.

Discipline : Génie Informatique, Automatique et Traitement du Signal

N° :