# Le meilleur des cas pour l'ordonnancement de groupes : Un nouvel indicateur proactif-réactif pour l'ordonnancement sous incertitudes

Zakaria Yahouni

UNIVERSITE BRETAGNE LOIRE

CENTRALE NANTES

# Thèse de Doctorat

Zakaria YAHOUNI

Mémoire présenté en vue de l'obtention
**du grade de Docteur de l'École Centrale de Nantes**
***Docteur de l'Université de* Tlemcen**
sous le *sceau de l'*Université Bretagne Loire

**École doctorale : Sciences et Technologies de l'Information et Mathématiques**

**Discipline : Automatique, productique**
**Unité de recherche : Laboratoire des Sciences du Numérique de Nantes**

**Soutenue le 23 Mai 2017**

# Le meilleur des cas pour l'ordonnancement de groupes

## Un nouvel indicateur proactif-réactif pour l'ordonnancement sous incertitudes

### JURY

| | |
|---|---|
| Président : | **Djamila HADJ SLIMANE**, Professeur, Université de Tlemcen |
| Rapporteurs : | **Christian ARTIGUES**, Directeur de recherche, LAAS-CNRS Toulouse<br>**Rabah GOURI**, Maître de conférences HDR, École nationale supérieure de technologie d'Alger |
| Examinateurs: | **Farouk YALAOUI**, Professeur des Universités, Université de Technologie de Troyes<br>**Jean-Jacques LOISEAU**, Directeur de recherche, LS2N-CNRS École Centrale de Nantes<br>**David DAMAND**, Maître de conférences, École de management de Strasbourg |
| Directeur de thèse : | **Nasser MEBARKI**, Maître de conférences HDR, Université de Nantes |
| Co-directeur de thèse : | **Zaki SARI**, Professeur, Université de Tlemcen |

# Remerciements

Je voudrais tout d'abord exprimer mes plus profonds remerciements à **Nasser MEBARKI** qui a dirigé cette thèse et sans qui ce travail n'aurait pas vu le jour. Merci pour votre disponibilité, votre rigueur scientifique, vos remarques judicieuses et tous les travaux réalisés ensemble.

Je tiens aussi à remercier **Zaki SARI** de m'avoir donné la chance de préparer cette thèse et de sa confiance. Merci de vos conseils précieux, votre aide et vos qualités humaines que j'ai appréciées.

Je remercie particulièrement **Jean-jacques LOISEAU** de m'avoir accueilli à LS2N (IRCCyN) au sein de son équipe. Merci de toute l'aide que vous avez apportée à la thèse, de votre disponibilité et gentillesse et de vos conseils pertinents. Je remercie aussi les membres du comité de suivi de cette thèse **Jean-Charles BILLAUT** et **Pascal BERRUET** pour leur remarques et encouragements.

Je tiens également à remercier **Christian ARTIGUES** et **Rabah GOURI** de m'avoir fait l'honneur d'accepter d'être rapporteurs de cette thèse, ainsi que **Djamila HADJ SLIMANE**, **Farouk YALAOUI** et **David DAMAN** pour avoir accepté d'être membre de mon jury.

Merci à **mes proches**, **mes collègues** et tous **mes amis** de Tlemcen et Nantes. Je ne cite pas les noms pour n'oublier personne. Merci pour l'amitié, les moments agréables que nous avons passés et surtout le soutien et l'encouragement que vous m'avez apportés durant les moments difficiles de la thèse.

Pour finir, je remercie mes sœurs **Sarra *et* Chaimaa** et particulièrement ***mes parents*** de leur encouragement, patience et soutien tout au long de mes études. Les mots ne seront jamais assez explicites pour exprimer mes remerciements et ma gratitude. C'est à vous qu'est dédiée cette thèse.

# Table des matières

# Liste des tableaux

# Liste des figures

# Liste des notations

— $O_i$ opération $i$.
— $r_i$ date de disponibilité de l'opération $O_i$.
— $t_i$ date de début de l'opération $O_i$. .
— $\rho_i$ durée de traitement de l'opération $O_i$.
— $d_i$ date d'échéance de l'opération $O_i$.
— $C_i$ date de fin de l'opération $O_i$.
— $M_k$ machine $k$.
— $M_{O_i}$ la machine exécutant l'opération $O_i$.
— $\Gamma^-$ les prédécesseurs de l'opération $O_i$.
— $\Gamma^+$ les successeurs de l'opération $O_i$.
— $f$ fonction objective de l'ordonnancement.
— $\mathcal{J}$ ensemble de jobs $\{J_j\}_{j=1,\dots,n}$
— $\mathcal{M}$ ensemble de machines $\{M_k\}_{k=1,\dots,m}$
— $J_j$ le job $j$.
— $O_{k,j}$ opération du job $j$ sur la machine $k$.
— $r_{k,j}$ la date de disponibilité du job $j$ sur la machine $k$.
— $t_{k,j}$ la date de début du job $j$ sur la machine $k$.
— $\rho_{k,j}$ la dure de traitement du job $j$ sur la machine $k$.
— $d_{k,j}$ la date d'échéance du job $j$ sur la machine $k$.
— $C_{k,j}$ la date de fin du job $j$ sur la machine $k$.
— $q_{k,j}$ la durée de latence de l'opération du job $j$ exécuté sur la machine $k$.
— $O^-_{k,j}$ l'opération prédécesseur du job $j$ exécuté sur la machine $k$.
— $O^+_{k,j}$ l'opération successeur du job $j$ exécuté sur la machine $k$.
— $O_{\pi^k_j,j}$ la dernière opération prédécesseur de l'opération $O_{k,j}$ dans le job $j$
— $O_{\omega^k_j,j}$ la première opération prédécesseur de l'opération $O_{k,j}$ dans le job $j$
— $C_j$ la date de fin maximale du job $j$.
— $C_{\max} = \max_{J_j \in \mathcal{J}} C_j$ le makespan.
— $T_{\max} = \max_{J_j \in \mathcal{J}} T_j$ le retard maximum.
— $O_0$ et $O_*$ opérations fictives.
— $I$ un ensemble d'opérations traitées par la machine $M_k$.
— $C$ une clique de plus d'une opération de l'ensemble $I$.
— $O_e$ opération d'entrée de la clique $C$.
— $O_s$ opération sortie de la clique $C$.

— $P$ un problème d'ordonnancement.
— $S^p$ une solution du problème $P$.
— $N(S^p)$ la solution voisinage de $S^p$.
— $S^r$ la solution réalisée dans l'atelier (a posteriori).
— $G$ un ordonnancement de groupes d'opérations permutables.
— $G_i^k$ $i^{eme}$ groupe sur la machine $k$.
— $g_j^k$ la position du groupe contenant l'opération $O_{k,j}$
— $G^k_{g_j^k-1}$ le groupe prédécesseur sur la machine $k$ du groupe contenant l'opération $O_{k,j}$.
—
— $G^k_{g_j^k+1}$ le groupe successeur sur la machine $k$ du groupe contenant l'opération $O_{k,j}$.
— $\#Seq$ le nombre d'ordonnancement semi-actifs caractérisés par $G$.
— $\#\phi$ la flexibilité de $G$.
— $\epsilon$ la valeur maximale (tolérée) de $f$.
— $\underline{\tau}_{k,j}$ la date de début au plus tôt dans le pire des cas de l'opération $O_{k,j}$.
— $\underline{C}_{k,j}$ la date de fin au plus tôt dans le pire des cas de l'opération $O_{k,j}$.
— $\overline{\tau}_{k,j}$ la date de début au plus tard dans le pire des cas de l'opération $O_{k,j}$.
— $\overline{C}_{k,j}$ la date de fin au plus tard dans le pire des cas de l'opération $O_{k,j}$.
— $m_{\text{seq}}(O_{k,j})$ la marge libre séquentielle de l'opération $O_{k,j}$.
— $m_{\text{sn}}(O_{k,j})$ la marge libre propre de l'opération $O_{k,j}$.
— $m_{\text{sg}}(O_{k,j})$ la marge libre séquentielle de l'opération $O_{k,j}$ dans le groupe $G^k_{g_j^k}$.
— $r'_{k,j}$ la borne inférieure de la date de début dans le meilleur des cas de l'opération $O_{k,j}$.
— $q'_{k,j}$ la borne inférieure de la durée de latence dans le meilleur des cas de l'opération $O_{k,j}$.

# Introduction générale

L'ordonnancement d'un atelier de production consiste à allouer dans un horizon de temps précis un ensemble de machines limitées à un ensemble de jobs tout en satisfaisant un ensemble de contraintes et en optimisant un ou plusieurs objectifs. La résolution de cette problématique a constitué un nombre important de sujets de recherches depuis les années cinquante. Différentes méthodes exactes et approchées ont été proposées sous l'hypothèse que les variables de l'atelier sont déterministes. Cependant, en réalité et dans un contexte industriel, cette hypothèse est généralement incorrecte ; les variables de l'atelier subissent des changements liés à des perturbations incontrôlables, telles que, les pannes des machines, les retards de livraison, l'arrivée des jobs urgents, etc. Ces perturbations, une fois qu'elles se produisent dans l'atelier, empêchent l'exécution de l'ordonnancement exactement comme planifié.

Afin de palier ces incertitudes, différentes méthodes, dites robustes, sont proposées dans la littérature. Ces méthodes motivées par leurs applications industrielles, cherchent soit, à intégrer des modèles d'incertitudes *a priori* dans l'ordonnancement, soit à corriger ce dernier à l'aide d'algorithmes dits réactifs, tout en évitant la détérioration des performances attendues.

Différentes études de la littérature montrent que l'opérateur humain est considéré comme l'acteur approprié pour la prise de décision en ordonnancement. Malgré des tentatives pour remplacer le travail de l'humain par des programmes et algorithmes de l'intelligence artificielle, la recherche actuelle avec des cas d'études réels prouvent que l'humain expert est irremplaçable surtout dans des situations imprévues grâce à sa capacité d'adaptation à l'environnement. Cependant, le cerveau humain est limité face à des calculs complexes où la machine s'avère très efficace. Pour toutes ces raisons, une coopération homme-machine est crucial.

Une telle coopération est pilotée par un système d'aide à la décision. Dans ce cadre, nous nous intéressons à améliorer le système d'aide à la décision afin de tirer pleinement avantage de la coopération homme-machine en ordonnancement sous incertitudes. Le contexte de la résolution proposée est centré sur la méthode d'ordonnancement de groupes. Cette méthode est une des approches les plus étudiées dans la littérature consacrées à l'ordonnancent sous incertitudes et a été implémentée dans plusieurs entreprises industrielles en France. Elle est constituée en deux phases, une première phase dite hors-ligne permettant de proposer un ensemble de solutions admissibles non énumérées au lieu d'un seul ordonnancement. Ensuite, une de ces

solutions est sélectionnée en temps-réel durant la phase de décision. L'opérateur doit choisir à l'aide d'un système d'aide à la décision, la solution/l'ordonnancement la/le plus adapté(e) aux changements survenus dans l'atelier.

La *marge libre séquentielle* est une adaptation de la "marge libre" d'une opération dans une solution d'ordonnancement et représente le critère d'aide à la décision le plus utilisé dans la littérature sur l'ordonnancement de groupes. Cependant, un nouveau critère d'aide à la décision "meilleur des cas" a été proposé dans la thèse de (Pinot 2008). Ce critère représente la meilleure performance atteignable par un ordonnancement de groupes et le calcul de cette mesure est un problème NP-difficile. Ce calcul représente la problématique principale de notre thèse et s'intègre dans les perspectives des travaux de thèse de Guillaume PINOT (Pinot 2008). L'auteur a proposé comme résolution, une borne inférieure ainsi qu'une méthode de séparation et d'évaluation pour le calcul du meilleur des cas. Les expérimentations réalisées ont montré un potentiel intérêt de ce critère dans un système d'aide à la décision. Mais, certaines limites ont été soulevées :

— Les bornes inférieures, calculées avec un algorithme de complexité polynomiale, permettent de donner en temps réel une évaluation du meilleur des cas. Mais cette évaluation est parfois imprécise. Il faut améliorer la qualité de ces bornes inférieures pour permettre une meilleure évaluation en temps réel de la qualité dans le meilleur des cas.
— La procédure de séparation et d'évaluation doit également être améliorée afin de donner une évaluation précise de la qualité dans le meilleur des cas en un temps acceptable. En particulier, des conditions de réduction de l'espace de recherche peuvent être implémentées permettant de réduire les temps de réponse surtout pour des problèmes de grandes tailles.
— Les expérimentations menées ont montré une certaine désaffection des opérateurs humains concernant l'utilité du meilleur des cas.

Notre objectif tout au long de ce travail est d'abord d'améliorer cette mesure pour ensuite l'utiliser efficacement dans un système d'aide à la décision. L'approche adaptée et les méthodes de résolutions proposées sont exposées dans les trois parties du manuscrit :

— La **première partie** regroupe les différentes définitions et notions préliminaires pour expliquer la problématique traitée. Cette partie est elle même constituée de trois chapitres. Le *premier chapitre* est dédié aux définitions et notations du problème d'ordonnancement déterministe. Ensuite, l'atelier à cheminement multiple appelé job shop, choisi comme modèle théorique pour la problématique de cette thèse est détaillé dans le *second chapitre*. Le *troisième chapitre* soulève les limites d'une solution d'ordonnancement déterministe en pratique, pour ensuite présenter un survol des principales méthodes de résolutions d'un problème d'ordonnancement sous incertitudes. Enfin, le choix de la méthode utilisée (l'ordonnancement de groupes) est justifiée dans la dernière partie de ce chapitre.

— La **deuxième partie** est composée de deux chapitres. Le *premier chapitre* (chapitre 4) est dédié entièrement à la méthode d'ordonnancement de groupes. L'accent est mis sur les différentes évaluations de cette méthode, telles que, la flexibilité et la performance dans le pire des cas. Ensuite, nous soulignons la problématique d'évaluation du meilleur des cas. Les différentes bornes inférieures ainsi que la méthode de séparation et d'évaluation pour résoudre cette problématique sont proposées dans le *deuxième chapitre* de cette partie (chapitre 5).
— Dans la **troisième partie** (*chapitre* ), l'accent est mis principalement sur l'utilité du meilleur des cas dans un modèle de coopération homme-machine piloté par un système d'aide à la décision. Un état de l'art sur l'intervention de l'humain dans le processus d'ordonnancement est d'abord présenté. Ensuite, un modèle d'interface homme-machine adapté à l'ordonnancement de groupes est proposé. Avant d'expérimenter ce modèle avec des opérateurs humain dans le dernier chapitre, des tests de simulations de prise de décision avec le critère du meilleur des cas sont réalisés sur des instances de benchmark. Le *dernier chapitre* présente ensuite l'implémentation du meilleur des cas dans un cas d'étude réel. Enfin, une conclusion des résultats obtenus et des perspectives de travail sont présentées dans la dernière partie de ce manuscrit.

# Première partie

# Ordonnancement

# Préliminaires

Dans ce chapitre, nous présentons les définitions et notations préliminaires relatifs au problème d'ordonnancement d'ateliers de production. Nous définissons ensuite les différents types d'atelier de production, notamment les job shop, titre du chapitre suivant.

## 1.1 Définition de l'ordonnancement

L'ordonnancement consiste à affecter un ensemble de travaux (jobs) à un ensemble de ressources à capacité limitées (machines) en un temps précis afin de satisfaire un ou plusieurs objectifs. Dans la pratique, les notions de jobs et machines peuvent prendre différents sens, selon le contexte de mise en œuvre. Par exemple :

- En informatique, il s'agit de choisir l'ordre des processus à exécuter et leur affectation à des processeurs parallèles.
- Dans un atelier manufacturier, il s'agit d'établir un planning de fabrication où la séquence des opérations à réaliser sur les différentes ressources de l'atelier est définie de manière très précise.
- Dans un système logistique, il s'agit d'établir l'ordre des livraisons aux clients et leur affectation aux différentes ressources de transport.
- Dans un hôpital, il s'agit par exemple d'établir une programmation pour des blocs opératoires ou d'organiser le circuit du patient hospitalisé.
- En gestion de projet il s'agit de déterminer les dates de début et de fin des activités du projet, sans qu'il y ait forcément de ressources dans ce dernier cas.
- etc.

Dans ce mémoire, nous nous intéressons exclusivement aux problèmes d'ordonnancement d'atelier. Nombreuses sont les définitions de l'ordonnancement d'atelier, certaines de ces définitions sont évoquées dans la littérature comme suit :

"*ordonnancer un ensemble de tâches, c'est programmer leur exécution en leur allouant les ressources requises et en fixant leurs dates de début*" (Gotha 1993)

"*Résoudre un problème d'ordonnancement c'est résoudre un problème d'optimisation combinatoire constitué d'un ensemble de tâches, d'un ensemble de ressources, d'un ensemble d'objectifs et d'un ensemble de contraintes. La solution à ce problème est un ordonnancement qui donne des indications sur les dates de début des tâches ou qui propose une relation d'ordre total ou partiel sur la séquence de réalisation des tâches*" (Artigues 2004)

"*Scheduling is a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives*" (Pinedo 2012)

## 1.2 Éléments d'un problème d'ordonnancement

Dans un problème d'ordonnancement, quatre éléments fondamentaux sont nécessaires à la définition et la résolution d'un problème d'ordonnancement : les tâches, les ressources, les contraintes et les objectifs. Les définitions représentées dans cette section sont issue des références suivantes Gotha (1993), Esswein (2003), Pinedo (2012), Mebarki (2012).

### 1.2.1 Tâches

Une tâche est une entité de travail localisée dans le temps par une date de disponibilité $r_i$, une date de début $t_i$, une durée de traitement $\rho_i$, une date d'échéance $d_i$ et une date de fin $C_i$. La réalisation d'une tâche nécessite une certaine durée ($\rho_i$) à passer sur un certain nombre d'unités de ressources tel que $\rho_i = C_i - t_i$ (Fig. 1.1).

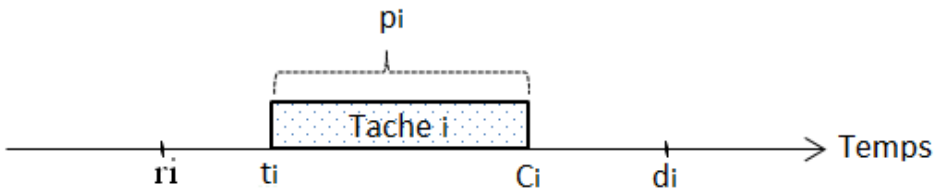



FIGURE 1.1 – *Représentation d'une tâche*

En ordonnancement d'atelier, une tâche correspond à *une opération* notée $O_i$. Le terme *Job* ou *travail* désigne l'ensemble d'opérations constituant un même travail.

### 1.2.2 Ressources

Une ressource peut être vue comme un moyen technique (machine) ou humain requis pour l'exécution d'une tâche dans la disponibilité limitée ou non est connue *a priori*. On distingue deux types de ressources :

— *Ressources renouvelables* : sont des ressources disponibles après consommation par d'autres tâches avec les mêmes quantités (machines, humains, processeur, etc).

— *Ressources consommables* (non renouvelables) : ces ressources sont représentées par une matière première ou un budget par exemple. Contrairement aux ressources renouvelables, elles ne deviennent pas disponibles après consommation.

Certaines ressources peuvent exécuter plusieurs tâches à la fois et sont appelées des ressources partageables. Contrairement à ces ressources, les ressources dites non partageables ou *disjonctives* ne peuvent exécuter qu'une seule tâche à la fois.

En ordonnancement d'atelier, on parle généralement de machines. Dans ce mémoire, nous nous intéressons qu'aux machines renouvelables et disjonctives. $k$ et $m$ représentent respectivement l'indice de la machine ($M_k$) et le nombre de machine dans un problème d'ordonnancement d'atelier. $M_{O_i}$ désigne la machine exécutant l'opération $O_i$.

### 1.2.3 Contraintes

Les contraintes sont les restrictions que peuvent prendre les valeurs de décisions. Trois types de contraintes sont distingués dans un problème d'ordonnancement :

— *Contraintes de temps* : elles sont généralement représentées par les dates d'échéances ($d_i$) et les dates de disponibilités ($r_i$) des opérations. Un ordonnancement admissible doit donc satisfaire la contrainte suivante :

$$t_i \geq r_i$$

De plus, dans un ordonnancement sans retard, la contrainte des dates d'échéances doit être satisfaite ($C_i \leq d_i$).

— *Contraintes d'enchaînement* : les opérations sont souvent liées entre elles par des contraintes de précédences $\Gamma^-$ (resp. de succession $\Gamma^+$). Une relation de précédence (resp. de succession) de l'opération $O_j$ par l'opération $O_i$ est représentée par la formule suivante :

$$t_j \geq C_i \text{ (resp. } t_i \geq C_j)$$

— *Contraintes de ressources* : une contrainte d'une ressource représente la nature et la quantité de la ressource utilisée pour l'exécution d'une opération $O_i$ au cours du temps.

Dans le cas des ressources disjonctives renouvelables dans un ordonnancement d'atelier, une machine ne peut traiter qu'une seule opération à la fois. Une relation de deux opérations disponibles à un même instant dans une même machine est représentée comme suit :

$$\forall (O_i, O_j) M_{O_i} = M_{O_j} \text{ tel que } C_i \leq t_j \text{ ou } C_j \leq t_i$$

### 1.2.4 Objectifs

La qualité d'un ordonnancement est mesurée au travers différents critères de performance, généralement par des fonctions des dates de fin des opérations ($C_i$).

Un critère de performance est dit régulier s'il s'agit d'une fonction croissante des dates de fin des opérations ($C_i$). Cette caractéristique signifie que, terminer une opération plus tôt ne peut dégrader un critère régulier. On cherche donc à exécuter les opérations au plus tôt dans le cas de critères réguliers.

Un critère régulier de type *minsum* est représenté par la formule $\sum f_i = \sum_i f_i(C_i)$.

Un critère régulier de type *minmax* est représenté par la formule $\max f_i = \max_i f_i(C_i)$. L'objectif *makespan* est l'objectif régulier le plus utilisé dans la littérature et représente la date de fin maximale de toutes les opérations de l'ordonnancement noté $C_{max} = \max_i C_i$.

### 1.2.5 Modèles d'ateliers

Dans un système manufacturier, le type d'atelier de production pourra être défini selon le nombre de machines et l'ordre d'utilisation des machines pour la fabrication d'un produit. Parmi les différents types d'ateliers, on distingue trois ateliers principaux :

— *Flow Shop* : cet atelier est à cheminement unique. L'ordre de passage des travaux (produits) sur les machines est unique.
— *Job Shop* : cet atelier est à cheminement multiple. Chaque travail a sa propre séquence de passage sur les machines et le nombre de machines d'exécution pour chaque travail n'est pas forcément le même.
— *Open Shop* : cet atelier est dit à cheminement libre. L'ordre de passage des travaux sur les machines est totalement libre.

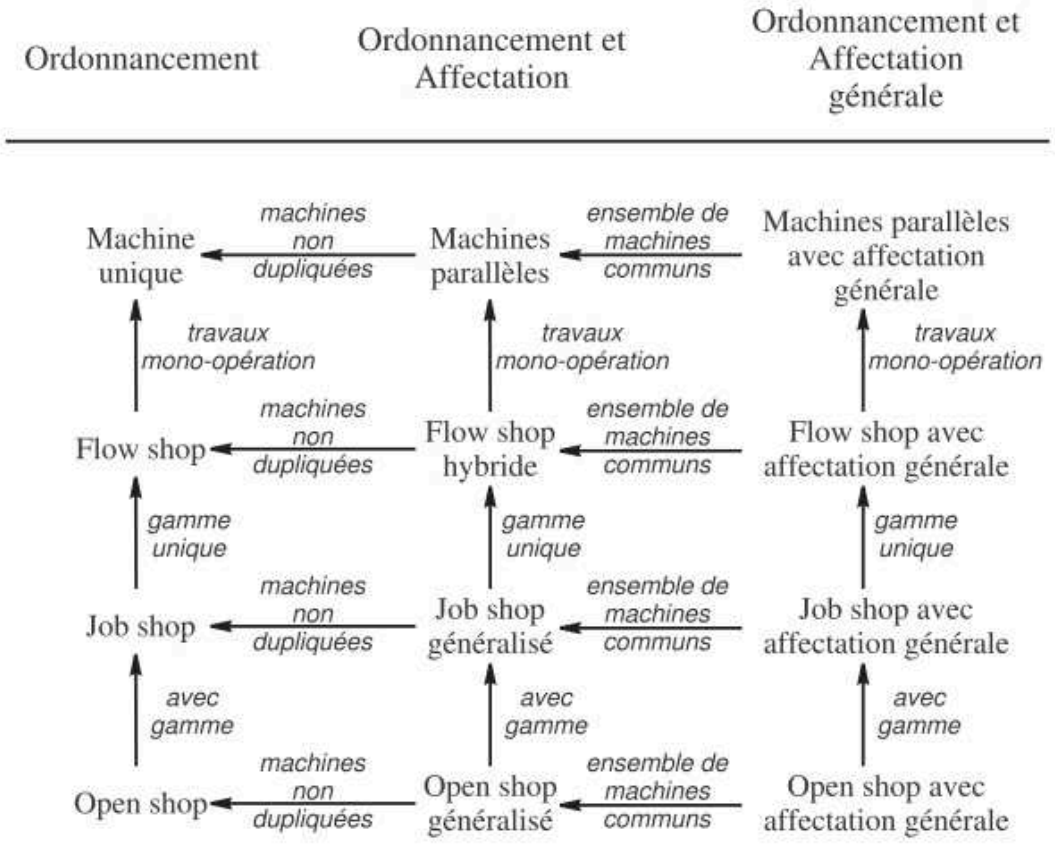



FIGURE 1.2 – *Typologie des problèmes d'ordonnancement (Esswein 2003)*

Fig. 1.2 présente la typologie des problèmes d'ordonnancement. Dans ce mémoire,

nous nous intéressons exclusivement aux problèmes à cheminement multiple. Dans la section suivante, nous présentons en détails ce problème de Job Shop.

### 1.2.6 Notation des variables d'un problème d'ordonnancement

La notation de Graham et Lawler (1979) est très populaire et a été après étendu par Blazewicz (1996) pour la représentation des variables d'un problème d'ordonnancement. Cette notation permettra de présenter de manière précise à quel problème d'ordonnancement nous avons affaire. Cette notation se compose de trois variables et se présente sous la forme suivante $\alpha|\beta|\gamma$ :

— $\alpha$ : cette variante décrit la structure du problème et se décompose en deux champs $\alpha_1$ qui représente le nombre de machines pour le problème d'ordonnancement, ce champ est optionnel et pourra être absent de la représentation du problème d'ordonnancement. Dans ce cas le nombre de machines est variable. Le champ $\alpha_2$ précise la catégorie du type de problème.

— $\beta$ : cette variante décrit les contraintes additionnelles au problème d'ordonnancement comme les dates de disponibilité $r_i$ et les dates d'échéance $d_i$.

— $\gamma$ : cette variante représente la fonction objective du problème.

Le tableau 1.1 décrit les principales valeurs de ces trois variantes :

| Variante | Notation | Description |
|---|---|---|
| $\alpha_1$ | $J$ | Job Shop |
| | $F$ | Flow Shop |
| | $FH$ | Flow Shop Hybride |
| | $O$ | Open Shop |
| | ... | ... |
| $\alpha_2$ | $\{\}$ | Nombre de machines variable |
| | $\{m\}$ | Problème à $m$ Machines |
| $\beta$ | $r_i$ | contraintes de dates de début au plus tôt des opérations |
| | $d_i$ | Contraintes de dates d'échéance des opérations |
| | $prec$ | Contraintes de précédences entre les opérations |
| | $pmtn$ | Préemption des opérations autorisé |
| | ... | ... |
| $\gamma$ | $C_{max} = max_{i\in\{0..n\}} C_i$ | *Makespan* : la plus grande date de fin |
| | $\sum w_i C_i$ | La date de fin pondéré des travaux |
| | $T_{max} = \max_{i\in\{0..n\}}(C_i - d_i, 0)$ | Le retard maximum des travaux |
| | $\sum w_i T_i$ | Le retard pondéré des travaux |
| | $\sum T_i$ | Le retard total |
| | $L_{max} = \max_{i\in\{0..n\}}(C_i - d_i)$ | Le retard algébrique maximum des travaux |
| | $\sum w_i L_i$ | Le retard algébrique pondéré des travaux |
| | $\sum \lvert L_i \rvert$ | Le retard total absolu |
| | $\sum U_i = \lvert(J_i\rvert$ if $C_i > d_i)$ | Le nombre de travaux en retard |
| | - | L'objective est de trouver une solution admissible |
| | ... | ... |

TABLE 1.1 – *Principales variantes de la notation $\alpha|\beta|\gamma$ pour les problèmes d'ordonnancement*

# Problème de Job Shop

Ce chapitre est focalisé sur la représentation d'un problème de job shop. Les différentes notations sont introduites et les différentes méthodes principales de résolutions sont soulignées. Deux types de méthodes sont distingués et la méthode de séparation et d'évaluation est définie. Cette méthode est ensuite utilisée dans le chapitre 5 de ce manuscrit.

## 2.1 Définition

Au cours des dernières années, un nombre important d'études s'est focalisé sur les différentes méthodes et solutions pour la résolution de la problématique d'ordonnancement d'atelier à cheminement multiple. Dans un tel problème de job shop $P$, $n$ travaux : $\mathcal{J} = \{J_1, J_2, ..., J_n\}$ doivent être traités par un ensemble $\mathcal{M}$ machines = $M_1, M_2, ..., M_m$ sous réserve de respecter les contraintes du problème. Les machines sont supposées être disponible à l'instant zéro et peuvent traiter une seule opération à la fois et sans préemption.

Une opération d'un job $J_j$ représente l'exécution du job sur une seule machine. Un job se compose de plusieurs opérations qui doivent être exécutées successivement sur les machines et sont représentées par une chaîne. Associé à chaque opération du job $J_j$ exécuté sur une machine $M_k$ : une date de disponibilité $r_{k,j}$, une date de début d'exécution $t_{k,j}$, un temps de traitement $\rho_{k,j}$, une date de fin $C_{k,j}$ et une durée de latence (*tail*) entre la date de fin de l'ordonnancement et la date de fin de l'opération $q_{k,j}$. $O^-_{k,j}$ (resp. $O^+_{k,j}$) représente une opération prédécesseur (resp. successeur) de l'opération $O_{k,j}$ dans le job $J_j$. $O_{\pi^k_j,j}$ (resp. $O_{\omega^k_j,j}$) représente la dernière opération prédécesseur (resp. première opération successeur) de l'opération $O_{k,j}$ dans le job $J_j$, $C_j$ représente la date de fin de la dernière opération du job $J_j$.

Un ordonnancement est représenté par un vecteur des dates de début/fin des jobs ($t_{k,j}/C_{k,j}$) sur les machines. Un ordonnancement est dit *admissible* s'il satisfait toutes les contraintes décrites dans la section 1.2.3. Ce problème est noté $J/r_{k,j}, q_{k,j}/f$ selon la classification de Graham et Lawler (1979).

La qualité d'un ordonnancement admissible est représentée par une mesure basée sur les dates de fin des jobs ($C_j$). Dans ce manuscrit, nous nous intéressons aux deux objectifs les plus utilisés dans la littérature : le *makespan* ($C_{\max}$) et le retard maximum des jobs ($L_{\max}$) et qui sont représentées respectivement par les formules suivantes :

$$C^*_{\max} = \min(C_{max}) = \underbrace{Min}_{\text{ordo admissible}} (\max(C_j))\forall j \in J$$

$$L^*_{\max} = \min(L_{max}) = \underbrace{Min}_{\text{ordo admissible}} (\max(C_{k,j} - d_{k,j}))\forall k \in M, j \in J$$

## 2.2 Représentation d'un problème de Job Shop par un graphe disjonctif

Le modèle de graphe disjonctif est le modèle le plus utilisé dans la littérature pour représenter un problème d'ordonnancement. Dans les notations de Roy et Sussmann (1964), $G$ est un graphe disjonctif définie comme suit : $\mathscr{G} = \{G, D\}$, où $G = \{X, U\}$ est un graphe conjonctif et $D$ est un ensemble de disjonctions. Dans ce graphe, les opérations de chaque job sont représentées par des nœuds liés par des arcs de précédence (conjonctifs). Les opérations des jobs partageant une même ressource sont liées par deux arcs conjonctifs opposés dits arcs disjonctifs ($\leftrightarrow$). Nous définissons une valeur entre chaque paire de nœuds relié par un arc conjonctif, cette valeur représente le temps de traitement de la première opération de la relation.

Deux opérations fictives sont ajoutées au graphe : $O_0$ et $O_*$ de telle sorte que $O_0$ (resp. $O_*$) représente l'opération prédécesseur (resp. successeur) de tous les jobs et la valeur de l'arc sortant (resp. entrant) de cette opération représente la date de disponibilité (resp. date d'échéance) du job.

$I$ est l'ensemble des opérations à traiter par la même machine. Une clique $C \subseteq I$ est un ensemble d'au moins deux opérations de $I$ $(O_1, O_2, ..., O_n)n > 1$ de telle sorte que chaque paire d'opérations de $C$ soit en disjonction. $O_e \in C$ (resp. $= O_s \in C$) est appelée "entrée" (resp. "sortie") de la clique $C$ si $0_e$ (resp. $O_s$) est ordonnancée dans $I$ avant (resp. après) toutes les autres opérations de $C$.

$A$ représente une sélection de l'ensemble des arcs disjonctifs du graphe, $G_A = \{X, U \bigcup A\}$ désigne le graphe conjonctif associé à la sélection $A$. Une solution admissible $S^p$ au problème $P$ représentée par un graphe disjonctif est une sélection consistante et complète de $A$ où toutes les disjonctions de $D$ sont sélectionnées. Le graphe conjonctif associé à la sélection $A$ est acyclique (ne contient aucun cycle).

Cette sélection permet de définir une séquence de jobs $\sigma$ parmi un ensemble de séquences possibles. Cette séquence représente un ensemble d'ordonnancements admissibles dont les dates de début/fin des opérations peuvent prendre différentes valeurs. Dans les objectifs réguliers, la solution dominante dans cet ensemble d'ordon-

nancement correspond à l'exécution des opérations *au plus tôt*. La date de fin de cette solution dominante est calculée par le plus long chemin entre $O_0$ et $O_*$.

**Exemple**

Pour illustrer les notations présentées ci-dessus, nous considérons un problème de job shop de trois jobs et trois machines comme représenté dans le tableau 2.1. Chaque job est composé de trois opérations qui doivent être exécutées successivement sur les trois machines selon le routage du job. Nous supposons que tous les jobs sont disponibles à l'instant 0 ($r_{k,j}$=0). Pour simplifier la représentation du graphe, les indexes de l'opération $O_{k,j}$ peuvent être remplacés par un seul index tel qu'illustré dans la troisième ligne du tableau 2.1.

TABLE 2.1 – *Exemple d'un problème de job shop*

| $J_j$ | $J_1$ | | | $J_2$ | | | $J_3$ | | |
|---|---|---|---|---|---|---|---|---|---|
| $O_{k,j}$ | $O_{1,1}$ | $O_{2,1}$ | $O_{3,1}$ | $O_{2,2}$ | $O_{3,2}$ | $O_{1,2}$ | $O_{1,3}$ | $O_{3,3}$ | $O_{2,3}$ |
| $O_i$ | $O_1$ | $O_2$ | $O_3$ | $O_4$ | $O_5$ | $O_6$ | $O_7$ | $O_8$ | $O_9$ |
| $M_k$ | $M_1$ | $M_2$ | $M_3$ | $M_2$ | $M_3$ | $M_1$ | $M_1$ | $M_3$ | $M_2$ |
| $\rho_{k,j}$ | 1 | 4 | 1 | 2 | 3 | 1 | 4 | 2 | 3 |
| $d_{k,j}$ | 5 | 7 | 10 | 2 | 8 | 9 | 5 | 8 | 10 |

Les figures 2.1 et 2.2 représentent le graphe disjonctif associé au problème décrit ci-dessus. Par ailleurs, la figure 2.3 représente un ordonnancement au plus tôt caractérisée par une sélection consistante et complète du graphe. Dans cet ordonnancement tous les arcs sont conjonctifs. Il ne reste qu'à calculer les dates de début/fin au plus tôt des opérations.

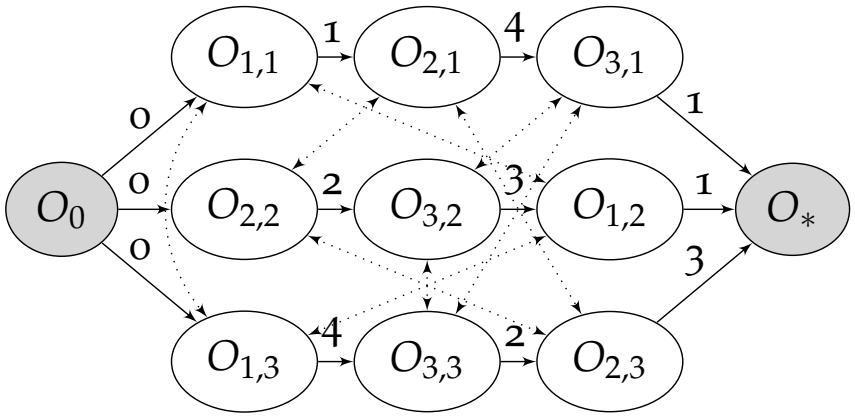



FIGURE 2.1 – *Graphe disjonctif.*

Une fois que les dates de début/fin sont définies, la solution pourra être représentée par un *diagramme de Gantt* (Gantt 1919) tel qu'illustré dans la figure 2.4.

En utilisant cette représentation de Gantt, on distingue trois types d'ordonnancements admissibles :

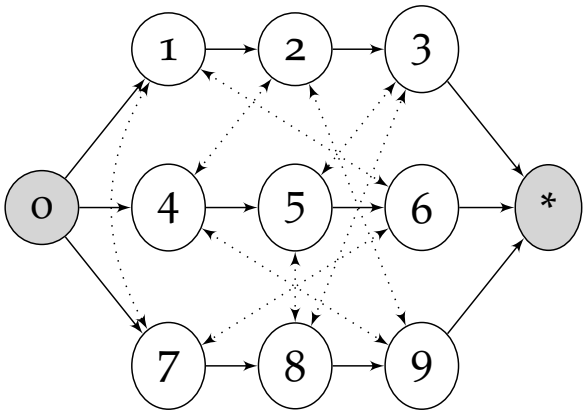



FIGURE 2.2 – *Simplification du graphe 2.1 .*

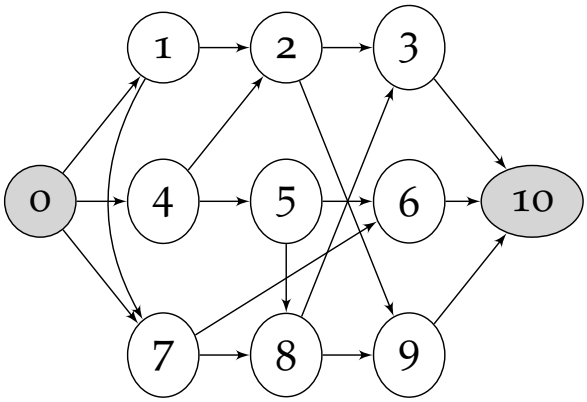



FIGURE 2.3 – *Graphe conjonctif.*

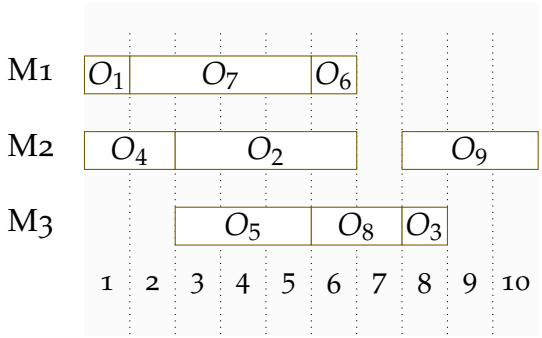



FIGURE 2.4 – *Représentation par un diagramme de Gantt.*

— Les ordonnancements semi-actifs : un ordonnancement semi-actif est un ordonnancement où toutes les opérations sont calées à gauche de telle sorte que chaque opération commence à sa date de début au plus tôt.
— Les ordonnancements actifs : un ordonnancement admissible est actif si aucune opération ne pourra commencer plus tôt sans changer le séquencement des autres opérations dans la même machine.
— Les ordonnancements sans délais : un ordonnancement admissible est sans délais si aucune machine n'est laissée inactive lorsqu'elle peut exécuter une opération en attente.

Un ordonnancement sans délais est également un ordonnancement actif. Un ordonnancement actif est également un ordonnancement semi-actif. Une solution d'ordonnancement optimale $S^*$ pour un problème de job shop appartient à la classe des ordonnancements actifs.

## 2.3 Méthodes de résolution d'un problème de Job Shop

Résoudre un problème d'ordonnancement de job shop consiste à *décider* de l'ordre d'exécution des opérations appartenant aux mêmes cliques $C$ (partageant les mêmes ressources) et par conséquent à chercher la meilleure solution qualifiée d'optimale. Ce problème d'optimisation combinatoire est démontré *NP-Difficile* ; en théorie de la complexité, *NP* (non-deterministic polynomial) représente la classe des problèmes qui ne peuvent pas être résolus en temps polynomial sur une machine de Turing dite déterministe. Un problème d'optimisation est *NP-Difficile* s'il est plus difficile que n'importe quel problème qui appartient à la classe *NP*.

Les méthodes et les algorithmes de résolutions de job shop datent des années soixante-dix avec la résolution de quelques problèmes particuliers de deux et trois machines. Dès que le nombre de machines est supérieur à 2, la complexité du problème devient NP-difficile. Ceci est remarqué dans certaines instances de la littérature comme le problème de 10 jobs et 10 machines de Fisher et Thomson (1963) qui a été ouvert pendant 25 ans (Carlier et Pinson 1989, Brucker et al. 1994a). Le tableau 2.2 résume la complexité de quelques classes de job shop.

En général, on distingue deux grandes catégories de méthodes de résolution d'un problème de job shop (Figure 2.5) : les méthodes exactes et les méthodes approchées. Les méthodes exactes contrairement aux autres méthodes garantissent de trouver une solution optimale du problème, mais dans un temps non borné. Par contre, les méthodes approchées ne garantissent pas l'optimalité de la solution, mais sont très performantes pour trouver une solution admissible généralement de bonne qualité dans un temps acceptable. Le but de cette partie n' est pas de donner une description précise de l' ensemble des méthodes de résolution d'un job shop, mais de passer sur quelques méthodes populaires, notamment la méthode exacte de séparation et d'évaluation que nous adoptons dans la deuxième partie de ce manuscrit pour résoudre notre problématique.

TABLE 2.2 – *Complexité de quelque problèmes de job shop*

| Problème Job shop | Complexité | Référence |
|---|---|---|
| $J1/r_{k,j}/C_{\max}$ | Polynomial | Lawler (1973) |
| $J1/r_{k,j}/L_{\max}$ | NP-difficile | Lenstra et al. (1977) |
| $J1/r_{k,j}, d_{k,j}/L_{\max}$ | NP-difficile | Garey et Johnson (1979) |
| $J1/r_{k,j}/\sum C_j$ | NP-difficile | Lenstra et al. (1977) |
| $J2/\rho_{k,j} = 1/C_{\max}$ | Polynomial | Timkovsky (1998) |
| $J2/\rho_{k,j} = 1, r_{k,j}/C_{\max}$ | Polynomial | Timkovsky (1998) |
| $J2/\rho_{k,j} = 1/\sum C_j$ | Polynomial | Kubiak et Timkovsky (1996) |
| $J2/\rho_{k,j} = 1/\sum U_{k,j}$ | Polynomial | Kravchenko (1999) |
| $J2/\rho_{k,j} = 1/\sum w_{k,j}U_{k,j}$ | NP-difficile | Kravchenko (1999) |
| $J2/\rho_{k,j} = 1/\sum w_{k,j}T_{k,j}$ | NP-difficile | Kravchenko (1999) |
| $J2/\rho_{k,j} = 1, r_{k,j}/\sum C_j$ | NP-difficile | Timkovsky (1998) |
| $J2/\rho_{k,j} = 1, r_{k,j}/\sum w_{k,j}C_{k,j}$ | NP-difficile | Timkovsky (1998) |
| $J2//C_{\max}$ | NP-difficile | Lenstra et Rinnooy Kan (1979) |
| $J2/pmtn/C_{\max}$ | NP-difficile | Lenstra et Rinnooy Kan (1979) |
| $J2/n = 3, pmtn/C_{\max}$ | NP-difficile | Brucker et al. (1999b) |
| $J2/n = 3, pmtn/\sum C_{k,j}$ | NP-difficile | Brucker et al. (1999b) |
| $J2//\sum C_j$ | NP-difficile | Garey et al. (1976) |
| $J3/n = 3/C_{\max}$ | NP-difficile | Sotskov et Shakhlevich (1995) |
| $J3/n = 3/\sum C_j$ | NP-difficile | Sotskov et Shakhlevich (1995) |
| $J3/\rho_{k,j} = 1/C_{\max}$ | NP-difficile | Lenstra et Rinnooy Kan (1979) |
| $J/prec, r_{k,j}, n = 2, pmtn/\sum w_{i,j}U_{i,j}$ | Polynomial | Sotskov (1991) |
| $J/prec, r_{k,j}, n = 2, pmtn/\sum w_{k,j}T_{k,j}$ | Polynomial | Sotskov (1991) |

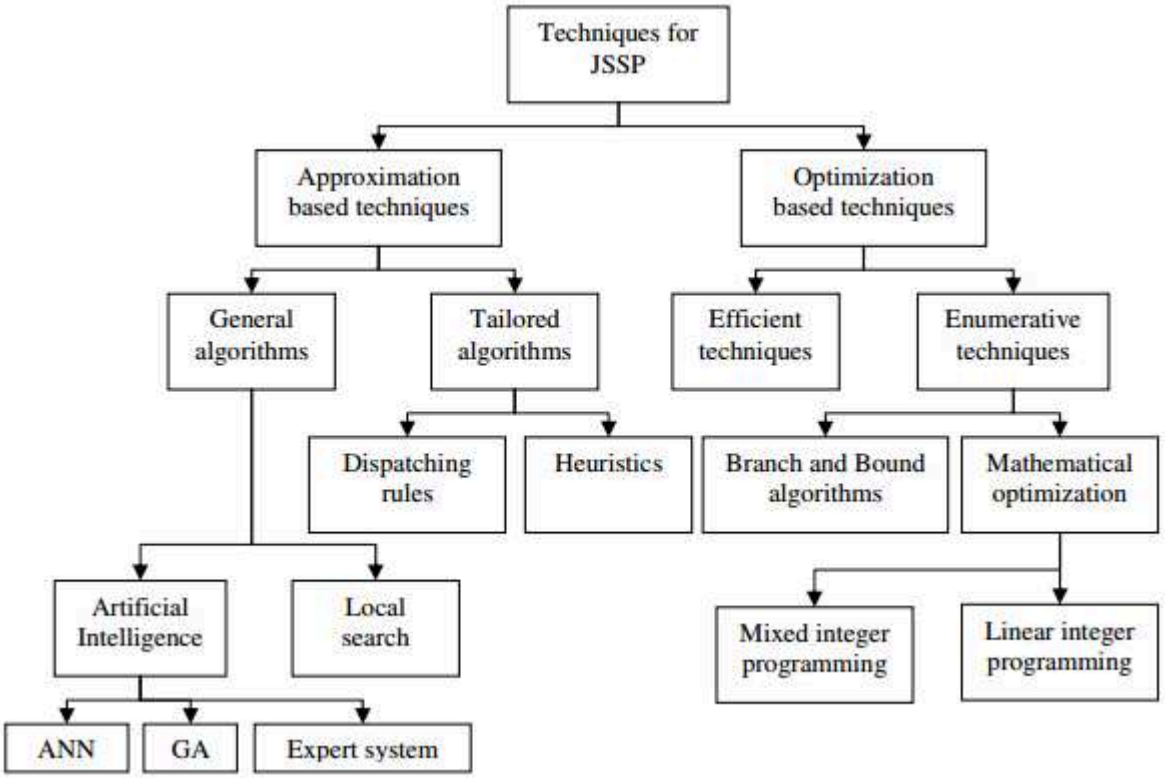



FIGURE 2.5 – *Méthodes de résolution d'un problème job shop (Jain et Meeran 1998)*

### 2.3.1 Méthodes exactes

Les méthodes exactes ont reçu une attention considérable pour la résolution du problème de job shop. Dans ce type de méthodes, on distingue les techniques dites efficaces et les techniques d'énumération de toutes les solutions possibles. Le premier type de techniques est généralement adapté à certains problèmes spécifiques qui peuvent être résolus par un algorithme polynomial. Les méthodes d'énumération complètes sont basées sur l'énumération exhaustive de toutes les solutions possibles.

La programmation dynamique est l'une des méthodes exactes les plus performantes et les plus robustes pour les problèmes d'énumération, mais cette approche est généralement limitée par sa capacité de stockage (Brusco et Stahl 2005). Pour les problèmes plus larges, la méthode de séparation et d'évaluation (Branch and Bound) est l'une des méthodes d'énumération la plus connue dans la littérature.

Cette méthode a été développée par BELLMAN dans les années cinquante et repose d'abord sur la séparation (Branch) de l'ensemble des solutions en sous-ensembles plus petits. L'exploration de ces solutions utilise ensuite une évaluation optimiste pour majorer (Bound) les sous-ensembles, ce qui permet de ne plus considérer que ceux susceptibles de contenir une solution potentiellement meilleure que la solution courante.

Un algorithme de séparation et d'évaluation construit progressivement un arbre dont chaque nœud représente un ensemble de solutions. Le premier nœud de l'arbre d'exploration désigne le problème $P$ et représente toutes les solutions admissibles du problème. Les solutions $S^s$ étant stockées dans les feuilles de l'arbre. Une implémen-

tation doit donc fournir une représentation de l'arbre et de ses nœuds ainsi qu'un algorithme de construction (séparation).

A chaque nœud de l'arbre, deux bornes d'évaluation sont calculées : une borne inférieure et une borne supérieure. Pour chaque nœud représentant une solution $S^p$, le calcul de bornes associées à ce nœud est relatif à la fonction $f(S^p)$ où $f$ représente la fonction objective de l'ordonnancement.

Dans un problème de minimisation par exemple, la borne inférieure représente une valeur de performance inférieure ou égale de la solution considérée. La borne supérieure représente la meilleure solution obtenue à l'instant courant. Cette borne supérieure est généralement initialisée à la tête de l'arbre en utilisant une simple heuristique. Un nœud dont la borne inférieure est inférieure ou égale à sa borne supérieure est supprimé de l'arbre ainsi que tout les nœuds qui découlent de se dernier. Cela signifie que les solutions dérivées de ce nœud ne peuvent pas être de meilleure qualité que la solution obtenue.

La solution optimale correspond à la meilleure solution admissible obtenue. Cependant, l'efficacité de la méthode de séparation et d'évaluation repose principalement sur la qualité des bornes utilisées dans la phase d'évaluation ainsi que de la méthode de parcours des nœuds. Plus proche est trouvée la solution optimale, plus de solutions sont éliminées de l'espace d'exploration (figure 2.6). Dans la plupart des travaux de la littérature, des méthodes approximatives telles que les algorithmes révolutionnaires sont utilisés pour le calcul des bornes supérieures.

### 2.3.2 Méthodes approximatives

Les méthodes approchées représentent une alternative très utile pour trouver une solution proche de l'optimale dans un laps de temps raisonnable. Ces méthodes sont caractérisées par leur facilité d'implémentation sur différentes problématiques. Elles sont classifiées en deux sous catégories :

— La première catégorie regroupe les règles de priorité et les heuristiques. Dans ce type d'algorithmes, une priorité est générée et donnée aux opérations qui doivent être exécutées dans la même machine. L'opération avec la plus petite/grande valeur de priorité est exécutée en premier.

— La deuxième classe concerne les techniques générales comme l'algorithme génétique. Elles tentent d'apprendre les techniques d'un problème pour converger vers une solution proche de l'optimale. Ce type d'algorithmes est généralement plus performant que les heuristiques et les règles de priorité et peuvent être adaptées à une large classe de problèmes différents, mais nécessitent un paramétrage très soigné pour être plus efficaces.

Les méthodes de résolution présentées dans ce chapitre considèrent que l'environnement d'ordonnancement est totalement déterministe où toutes les données du problème sont connues *a priori*, alors qu'en pratique, il existe plusieurs aléas indui-

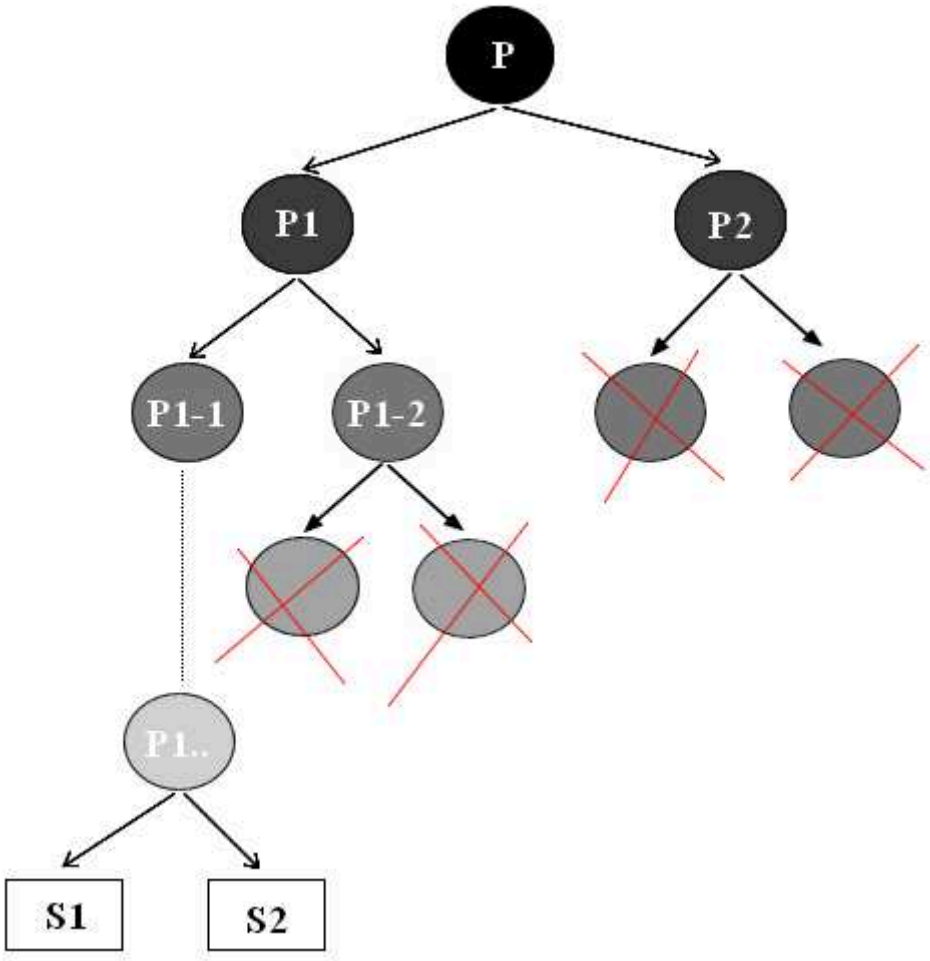



FIGURE 2.6 – *représentation de la méthode de séparation et d'évaluation*

sant un changement des données du problème, comme l'arrivée d'un nouveau job, les pannes des machines, le retard de livraison, etc.

# L'ordonnancement sous incertitudes

Ce chapitre met en évidence le fait qu'en pratique l'ordonnancement n'est pas déterministe. Les ateliers de production subissent des aléas et perturbations qui rendent le problème d'ordonnancement plus compliqué en terme de calcul. Nous commençons par donner des définitions relatives à l'ordonnancement sous incertitudes pour présenter ensuite un résumé sur les travaux de la littérature concernant ce problème. La dernière partie de ce chapitre justifie la méthode choisie et adoptée pour la suite de ce manuscrit.

## 3.1 Définitions

### 3.1.1 Phases de l'ordonnancement

L'ordonnancement d'atelier se compose de deux phases :

— Une phase *hors-ligne* appelée aussi *statique*. Dans cette phase, un planning prévisionnel satisfaisant toutes les contraintes du problème est proposé. Cet ordonnancement prévisionnel est appelé *ordonnancement de référence* sur lequel se référence la personne responsable de l'ordonnancement.

— Une phase *en-ligne*, appelée aussi la phase *réactive*. Dans cette phase, l'ordonnancement de référence est implémenté en temps réel dans l'atelier de production. La solution mise en place dans cette phase est appelée *ordonnancement réalisé* ($S^r$) et diffère de l'ordonnancement de référence ; selon le niveau de perturbation de l'atelier. Contrairement à la phase précédente, la prise en compte de la contrainte de temps est impérative dans cette phase.

Durant la phase réactive de l'ordonnancement, différentes perturbations peuvent se produire. Ces perturbations remettent en cause l'ordonnancement de référence généré lors de la phase statique.

Dans ce contexte, la littérature scientifique a connu ces dernières années une émergence de cette problématique d'ordonnancement sous incertitudes (Artigues 2004, Esswein 2003, La 2005, Billaut et al. 2008, Pinot 2008, Chaari 2010, Ourari 2011, Mebarki 2012, Artigues et al. 2016). À l'issue de ces travaux, différentes méthodes de résolutions ont été proposées.

Dans la suite de ce chapitre et avant de citer les différentes méthodes de résolution d'un problème d'ordonnancement sous incertitudes, nous présentons les définitions

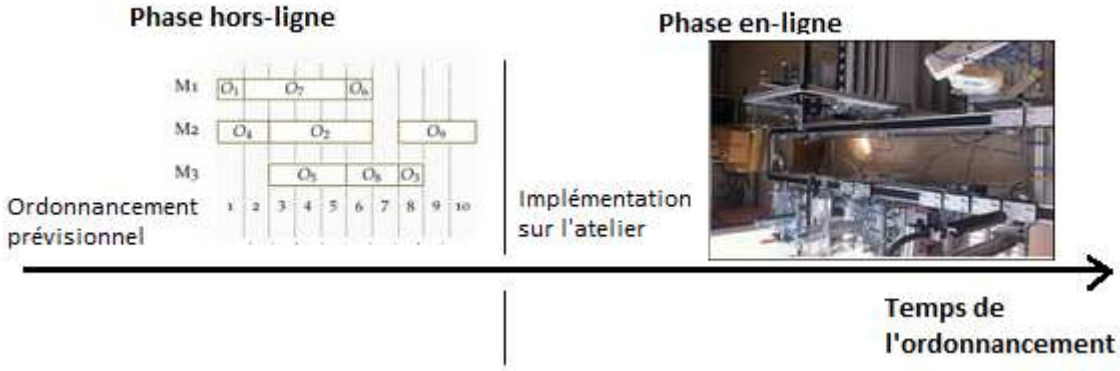



FIGURE 3.1 – *Phases de l'ordonnancement*

utilisées dans cette problématique, comme "robustesse", "flexibilité" et "stabilité". La majorité de cette partie est inspirée des travaux de Pinot (2008), Billaut et al. (2008), Sabuncuoglu et Goren (2009), Ourari (2011), Mebarki (2012).

### 3.1.2 Incertitudes

Esswein (2003) définit l'incertitude comme une modification potentielle des données d'un problème qui peut intervenir entre le calcul d'un ordonnancement de référence et la fin de sa mise en œuvre réelle dans l'atelier.

Sanlaville (2005) distingue trois types de changements, l'*incertitude*, l'*aléa*, la *variation*. Une donnée est incertaine si sa valeur exacte ne peut être déterminée au moment précis où cette donnée intervient. Dans ce cas, l'incertitude peut être modélisée par une variable aléatoire ou par un ensemble d'intervalles. L'aléa intervient lorsqu'une donnée dont la valeur connue a priori est susceptible d'être modifiée par un événement (panne, retard de livraison, ordre urgent, etc) dont l'occurrence si elle est connue, peut être représentée par une variable aléatoire. Une donnée est soumise à variation si sa valeur change durant l'exécution (capacité ou rendement d'une ressource par exemple).

Mebarki (2012) distingue l'incertitude et la perturbation. L'incertitude est associée soit à l'erreur soit au risque (psychologie, économie, gestion), soit à l'indétermination (physique). La perturbation est définie comme l'apparition d'un événement imprévu dont les modalités peuvent le plus souvent être modélisées.

Dans Davenport et Beck (2000), Kouvelis et Yu (1997) différents types d'incertitudes ont été définis. Parmi ces types d'incertitudes, nous reprenons celles distinguées par Billaut et al. (2008) :

— L'incertitude liée aux durées des opérations qui peuvent varier relativement par rapport à celles estimées.
— L'incertitude liée à la date de disponibilité d'une opération donnée.
— L'incertitude liée à la modification du problème initial, comme l'ajout ou la sup-

pression d'une opération ou la modification des caractéristiques structurelles la concernant.

— L'incertitude liée au coût associé à une opération.

D'une manière générale et pour faire simple, nous parlons d'incertitudes ou d'aléas lorsqu'on est dans la phase statique de l'ordonnancement où les données sont incertaines. Un aléa (contrairement aux incertitudes) est relatif à la structure du problème (panne machine) et non pas au changement des données. Nous appelons perturbation tout changement structurel ou non structurel de l'ordonnancement de référence durant la phase réactive.

### 3.1.3 Scénarios

Les incertitudes présentées dans la section précédente sont modélisées par différents scénarios $SC_{sc}$ tel que $sc$ correspond à un nouveau paramétrage des données causé par une possible perturbation et $SC_0$ correspond au scénario où aucune perturbation n'est présente. Le problème $P$ alors devient $P_{sc}$ qui désigne le problème du scénario $sc$. Une solution $S$ pour le problème $P_{sc}$ devient $S^p_{sc}$ et correspond à un ordonnancement du problème $P$ avec le scénario $sc$.

Chercher la solution optimale $S^*_{sc}$ dans un seul scénario est un problème $NP - difficile$. En réalité, le nombre de scénarios qui peuvent se présenter peut être très grand, ce qui rend le problème plus compliqué. Pour construire un ordonnancement qui prend en considération ces différentes perturbations, il est primordial de construire un ordonnancement de référence robuste. Dans la suite de ce mémoire, nous supposons que nous avons un seul scénario dont le but de simplifier le problème.

### 3.1.4 Robustesse

Il est perceptible que les définitions d'un ordonnancement robuste et la mesure de robustesse dans la littérature divergent dès qu'un domaine particulier est considéré. Dans la suite, nous citons quelques définitions restreintes à l'ordonnancement d'atelier :

Le Pape (1991) défini un ordonnancement robuste comme un ordonnancement pour lequel la violation des hypothèses sur lesquelles il est basé n'a pas ou que peu d'importance.

Dans Leon et al. (1994), un ordonnancement est dit robuste quand sa performance se dégrade très peu face à des perturbations.

Le groupe Gotha (2002) propose la définition suivante : "*La robustesse caractérise les performances d'un algorithme ou plutôt d'un processus complet de construction d'un ordonnancement ; en présence d'aléas*"

Dans Esswein (2003), un ordonnancement robuste est un ordonnancement dont les performances sont acceptables en présence d'incertitudes.

"*La robustesse de qualité est une propriété d'une solution dont la qualité, mesurée par la valeur de la fonction objective, ne change pas trop par rapport à l'optimalité lorsqu'il y a de petits changements sur les données du problème. La robustesse de solution peut être décrite comme la robustesse dans l'espace de solutions. La robustesse de solution apparaît quand une solution ne change pas beaucoup lorsqu'il y a de petits changements sur les données du problème, c'est-à-dire la méthode de résolution est capable de trouver une solution proche de la solution actuellement utilisée.*" (Sevaux et Sörensen 2004).

Billaut et al. (2008) définit l'ordonnancement robuste comme un ordonnancement peu sensible aux incertitudes des données.

Nous constatons que la robustesse est généralement mesurée soit par des critères de performance en terme de la valeur de la fonction objective de l'ordonnancement. Dans ce contexte, on parle généralement de la différence entre la pire performance qu'un ordonnancement réalisé pourra atteindre et la performance anticipée de l'ordonnancement de référence. Dans ce cas, la variable appelée *regret* représente la différence entre la performance de l'ordonnancement réalisé et l'ordonnancement optimal ($f(S^r) - f(S^*)$) (Bidot et al. 2008).

La deuxième mesure d'un ordonnancement robuste concerne la stabilité de ce dernier. La stabilité d'un ordonnancement représente le degré de changement entre un ordonnancement de référence et l'ordonnancement réalisé dans l'atelier ; cette stabilité est mesurée soit par le nombre d'opérations perturbées, soit par le nombre de changements survenu sur les dates de début/fin des opérations de l'ordonnancement. Dans ce dernier cas, un ordonnancement est dit 100% stable, lorsque les dates de début/fin des opérations de l'ordonnancement réalisé correspondent parfaitement aux dates de début/fin des opérations de l'ordonnancement de référence. Pour voir les différentes formules de stabilité d'un ordonnancement, le lecteur pourra se référer au travail de Sabuncuoglu et Goren (2009).

En parlant de la robustesse d'un ordonnancement, il est impératif de définir la relation entre robustesse et flexibilité, car afin de construire un ordonnancement robuste, il est important d'ajouter de la flexibilité dans l'ordonnancement de référence.

### 3.1.5 Flexibilité

La flexibilité d'un ordonnancement est la capacité à gérer la variété de ce dernier. Elle reflète le degré de liberté durant l'exécution d'un ordonnancement.

Cette flexibilité a pour but de palier les perturbations qui peuvent se produire durant l'exécution d'un ordonnancement de référence en proposant d'autres alternatives et solutions admissibles. Elle peut aussi être associée à un voisinage de solutions pouvant être examinées lors de l'exécution ou à une famille d'ordonnancements sans privilégier un ordonnancement en particulier (La 2005).

Dans Gotha (2002), les auteurs indiquent que la flexibilité peut être exprimée comme l'existence de modifications possibles dans un ordonnancement de référence, calculé hors-ligne, entraînant une perte de performance acceptable.

On distingue quatre types de flexibilité en général :

— Flexibilité d'affectation, cela permettra à une opération d'être exécutée sur d'autres machines que celles affectées initialement.
— Flexibilité temporelle : dans cette flexibilité, les temps de début et de fin des opérations peuvent être décalées dans un intervalle défini.
— Flexibilité séquentielle qui concerne l'ordre d'exécution des opérations sur une machine. Cet ordre flexible offrira implicitement une flexibilité temporelle entre les opérations.
— Flexibilité sur les modes : cela permettra de pouvoir changer le mode d'exécution d'une opération : la préemption, chevauchement, prise en compte ou non du temps d'installation, changement de ressources d'exécution d'une opération, etc. Cette flexibilité peut être introduite selon le contexte et la difficulté de la situation.

## 3.2 Approches de résolution

Le but de cette section n'est pas de donner un état de l'art exhaustif sur les approches de résolutions. Un travail considérable a été déjà présenté dans la littérature et cette thématique a été traité par plusieurs chercheurs d'une variété de domaines comme l'intelligence artificielle, la recherche opérationnelle, la coopération homme-machine, etc. En recherche opérationnel, Davenport et Beck (2000) a présenté une classification des approches de résolution d'un problème d'ordonnancement sous incertitudes. Cette classification est basée sur la façon dont les perturbations sont prises en compte.

Sabuncuoglu et Goren (2009) ont proposé une autre classification plus générale basée sur les mesures de robustesse et de stabilité d'un ordonnancement. Les méthodes étudiées sont classées selon deux questions décisionnelles : quand ordonnancer et comment ordonnancer ? La première question concerne le temps des décisions prises dans un ordonnancement et détermine la réactivité du système face aux événements imprévus de l'atelier. Tandis que la deuxième question répond à la façon dont l'ordonnancement est généré et modifié. Les réponses à ces deux questions ont été faite de manière à comprendre comment la robustesse et la stabilité sont utilisées pour palier les incertitudes d'un ordonnancement.

Dans la suite de cette section, nous présentons un résumé de quelques méthodes développées dans la littérature afin de passer ensuite à la méthode que nous allons adopter pour la suite de ce mémoire. Nous reprenons la classification de Davenport et Beck (2000) et Herroelen et Leus (2005) qui caractérise d'une manière générale trois classes principales d'approches de résolution.

### 3.2.1 Approches proactives

Le principe de ces approches est de construire durant la phase hors-ligne un ordonnancement de référence robuste en se basant *a priori* sur l'estimation des perturbations (Fang et al. 2015). Ces approches prennent en compte les incertitudes uniquement durant la phase prédictive de l'ordonnancement. Pour palier aux perturbations durant la phase réactive qui est supposée être triviale dans ce type d'approches, des techniques simples sont utilisées comme l'algorithme de décalage à droite proposé par Sadeh et al. (1993), Smith (1994). On trouve dans la littérature trois types d'approches proactives :

#### Les approches basées sur la redondance

Ce type d'approches construit un ordonnancement prédictif en affectant aux opérations des redondances temporelles (temps mort) et des ressources supplémentaires afin d'amortir l'impact des pannes éventuelles pendant l'exécution. Cette technique permettra la ré-exécution des tâches en cas d'échec.

Dans ce cas, l'introduction des temps morts et allocation de ressources supplémentaires dans un ordonnancement doit se faire très soigneusement afin de ne pas détériorer les performances attendues et estimés préalablement : respect des délais, coût, etc.

Une des méthodes proactives les plus connues dans l'ordonnancement de projet, est la méthode de la chaîne critique de Goldratt (1997). Cette méthode consiste à insérer, durant la phase hors-ligne de l'ordonnancement, un temps mort entre les opérations de la chaîne critique aux endroits adéquats du projet afin d'anticiper les retards qui peuvent se produire suite à une perturbation.

Différentes approches similaires ont été proposées dans la littérature pour l'ordonnancement d'atelier, par exemple, l'approche de Leon et al. (1994) suppose que les machines tombent en panne et pour palier à ses conséquences, les opérations de ces machines sont décalées à droite. De même, l'approche de Van de Vonder (2006) repose sur l'insertion volontaire de temps morts avant les successeurs des opérations. Le but est de pouvoir supporter sans trop de conséquences le retard des opérations.

#### Les approches stochastiques

Dans ce type d'approches, pour faire face aux incertitudes de l'atelier, la plupart des travaux de la littérature supposent que les durées des opérations sont des variables stochastiques avec une distribution de probabilité connue.

Dans ces approches probabilistes, plutôt que d'insérer des laps de temps supplémentaires dans l'ordonnancement, on cherche à construire un ordonnancement dont les performances sont acceptables dans un ensemble de situations (scénarios) données. Ces performances sont généralement mesurées par la pire déviation par rapport à la fonction objective.

Parmi les travaux les plus référencés dans la littérature est le travail de Brucker et al. (1999a) ainsi que celui de Herroelen et Leus (2005) qui considèrent l'ordonnancement de projet avec contraintes de ressources. La forme générale des approches proposées dans ces travaux est un problème de programmation stochastique à plusieurs niveaux, avec l'objectif de trouver une règle d'ordonnancement qui minimise le temps maximum des travaux.

Un exemple très connu des techniques probabiliste est l'approche *beta* de Daniels et Carrillo (1997). Dans ce travail, le problème à une machine est traité avec des durées opératoires aléatoires. Les auteurs présentent un algorithme de séparation et d'évaluation et une heuristique pour la construction d'un ordonnancement robuste avec un degré de robustesse maximum nommé $\beta$ ; garantissant que la somme des dates de fin des jobs ne dépasse pas un seuil donné.

Une approche similaire a été proposée par Beck et Wilson (2007). Les auteurs s'intéressent au problème du job shop avec des durées opératoires variables. L'objectif est de trouver une solution proactive robuste avec une grande probabilité d'avoir une bonne performance du makespan noté $\alpha$-makespan. Des heuristiques et une méthode de séparation et d'évaluation ont été proposées pour résoudre ce problème. Ces algorithmes sont basés sur le calcul des bornes inférieures et bornes supérieures du problème. Le calcul des bornes inférieures repose sur la relaxation des durées opératoires qui sont supposées être stochastiques. La simulation de Monte Carlo a été implémentée pour le calcul des bornes supérieures. La méthode de séparation et d'évaluation a montré ces limites faces aux problèmes de moyennes et grandes tailles. Pour palier à ce problème, les auteurs proposent des heuristiques alternatives.

Une autre approche est présentée dans Yang et Yu (2002). Les auteurs s'intéressent à la résolution d'un problème de minimisation des dates de fin moyennes sur une seule machine lorsque les durées opératoires sont supposées être aléatoires. Cette incertitude est modélisée par un ensemble de scénarios finis. Les auteurs ont montré que le problème est NP-difficile à résoudre particulièrement lorsque le nombre de *scénarii* considérés est grand. Les auteurs décrivent un algorithme exact de programmation dynamique ainsi que deux heuristiques pour la résolution de ce problème.

Lamas et Demeulemeester (2015) ont développé une méthode proactive pour le problème d'ordonnancement de projets avec contraintes de ressources. Les auteurs ont implémenté une méthode de branch-and-cut pour la génération d'un ordonnancement stable et robuste sans la prise en considération de la politique d'ordonnancement pour la phase réactive de l'ordonnancement.

#### Les approches basées sur plusieurs ordonnancements

Contrairement aux deux approches précédentes, ces approches ne génèrent pas un seul ordonnancement, mais proposent explicitement un ensemble d'ordonnancements adaptés à un ensemble de scénarios possibles. Ces méthodes sont intéressantes car

elles utilisent explicitement de la flexibilité séquentielle et non plus uniquement la flexibilité temporelle comme c'est généralement le cas.

La technique la plus connue dans la littérature est la technique just-in-case de Drummond et al. (1994). Cette technique est présentée dans le cadre d'une application réelle dans le domaine de l'observation astronomique. C'est un environnement à une machine qui est considéré et il s'agit d'ordonnancer les travaux dont les durées opératoires sont incertaines. Ces informations sont obtenues par des études statistiques. Cette méthode identifie les opérations les plus susceptibles d'échouer sur la base des informations disponibles sur les incertitudes possibles. L'objectif est de maximiser la probabilité d'exécuter toutes ces tâches en proposant pour chaque cas un ordonnancement alternatif. La limite de ces méthodes se situe dans les problèmes avec plusieurs ressources où l'explosion combinatoire rend le problème difficile et intraitable.

Toutefois, beaucoup d'effort ont été investi pour la construction des méthodes proactives prenant en compte les incertitudes de l'atelier durant la phase hors-ligne de l'ordonnancement. Ces approches généralement supposent que les incertitudes de l'atelier sont précises et connues *a priori*. Cependant, les variables utilisées durant l'exécution de l'ordonnancement ne sont pas toujours des informations prévisionnelles. Le point négatif de ces méthodes est qu'elles ne sont pas complétées par des algorithmes réactifs afin de faire face aux perturbations en temps réel.

### 3.2.2 Approches réactives

Dans ce type d'approches et contrairement aux approches précédentes, les perturbations ne sont pas prises en compte durant la phase hors-ligne de l'ordonnancement. Herroelen et Leus (2005) présente un état de l'art détaillé sur les approches réactives dans un problème d'ordonnancement de projet. Généralement, on distingue deux types d'approches :

#### Approches purement réactives

Appelées aussi approches *temps-réel* ou *dynamiques*. Dans ce type d'approches, l'ordonnancement est construit pas à pas en temps-réel dans l'atelier où aucun ordonnancement prédictif n'est calculé au préalable (Chan et al. 2002). Ce type d'approches est généralement utilisé lorsque le niveau de perturbations est trop important ou bien lorsque les variables du problème ne peuvent pas être définies préalablement d'où l'impossibilité de construire un ordonnancement de référence durant la phase hors-ligne.

Les décisions de l'ordonnancement des opérations sont prises durant l'exécution de ces derniers. Chaque fois qu'un conflit survient dans le choix de la prochaine opération à effectuer sur une machine qui devient libre, une décision doit être prise en se basant sur des techniques et stratégies réactives (comme une simple règle de

priorité). Ces techniques et stratégies sont en général basées sur l'observation et la connaissance de la distribution des opérations ainsi que sur les caractéristiques des machines. La figure 3.2 représente un schéma récapitulatif inspiré de Sabuncuoglu et Goren (2009) sur la réaction de ces approches face aux perturbations comparées aux approches proactives.

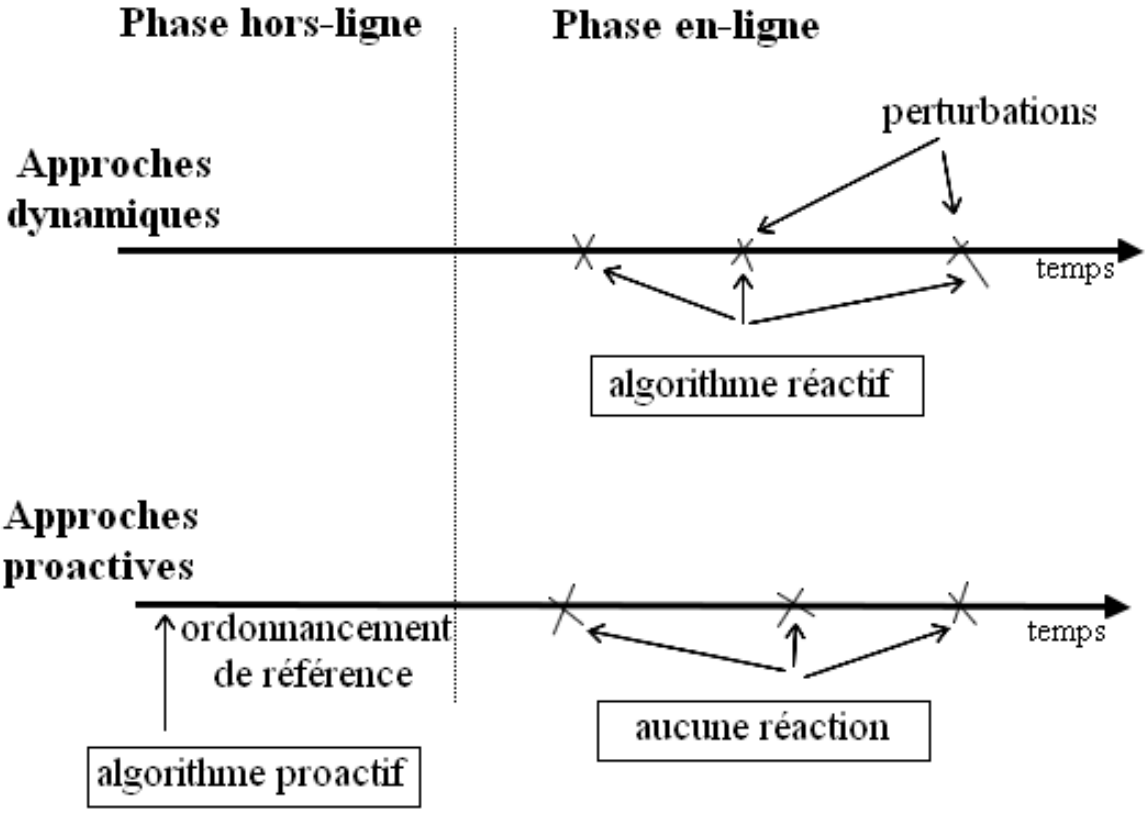



FIGURE 3.2 – *stratégies réactives entre une approche dynamique et une approche proactive*

On remarque que ce type d'approches apparaissent généralement dans les problèmes "juste-à-temps" (just-in-time) (T'Kindt et Billaut 2006) où l'objectif est de minimiser le stock et les coûts induits par ces stocks.

**Approches prédictives-réactives**

Contrairement aux approches temps-réel qui sont totalement réactives. Ces approches construisent un ordonnancement optimal ou proche de l'optimal durant la phase statique d'un ordonnancement sans l'anticipation des futures perturbations. Lorsque l'ordonnancement est exécuté dans l'atelier, ce dernier pourra être mis à jour par une technique de révision une fois qu'un événement inattendu survient sur l'atelier.

Une technique de révision est définie comme un algorithme en mesure de modifier les décisions d'un ordonnancement prédictif une fois implémenté dans l'atelier. En raison des contraintes de temps durant la phase d'exécution d'un ordonnancement,

il est courant d'utiliser des règles de priorités simples pour prendre des décisions en temps-réel et de manières réactives.

Le travail de van de Vonder et al. (2007) présente quatre algorithmes réactifs de réparation d'un ordonnancement. L'objectif du travail est de minimiser la déviation des dates de début/fin des opérations par rapport à l'ordonnancement prédictif lorsque ces opérations sont perturbées. L'ordonnancement prédictif est généré par l'algorithme de croisement combiné développé par Debels et Vanhoucke (2006). Les expérimentations menées sur les quatre algorithmes ont montré qu'une liste de priorités qui trie les opérations dans un ordre non-décroisant des dates de début avait les meilleurs résultats. Mais, cette liste doit être modifiée à chaque décision. Au final, les auteurs suggèrent de combiner ces algorithmes réactifs avec des méthodes proactives afin d'obtenir plus de stabilité et de robustesse.

Malgré que ces méthodes prennent en compte l'état réel de l'atelier contrairement aux approches précédentes, leur principal inconvénient est qu'elles ne peuvent pas garantir une performance minimale de l'ordonnancement d'où leur rendement généralement faible comparé aux approches proactives-réactives.

### 3.2.3 Approches proactives-réactives

Ces approches combinent certains avantages des approches précédentes en prenant en compte les deux phases de l'ordonnancement (Figure 3.3). L'idée principale est de construire un ordonnancement flexible durant la phase hors-ligne. Cet ordonnancement flexible caractérise plusieurs solutions et non pas une seule permettant d'absorber les perturbations qui peuvent se produire durant la phase en-ligne de l'ordonnancement.

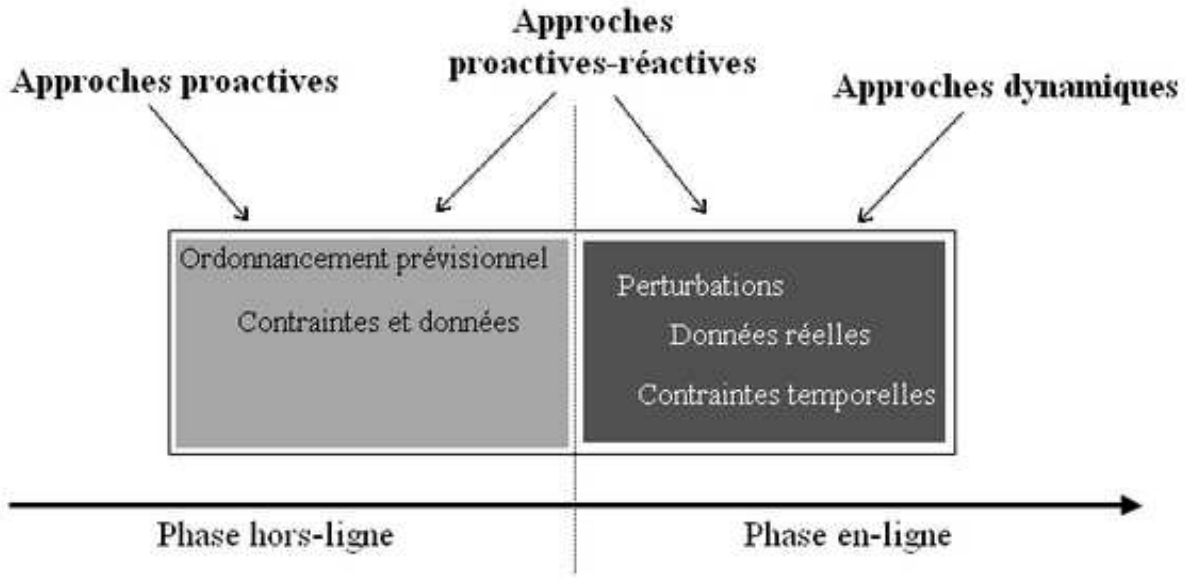



FIGURE 3.3 – *Approches de résolution d'un problème d'ordonnancement sous incertitudes*

Différentes méthodes proactives-réactives ont été proposées dans la littérature ces vingt dernières années. Par exemple, dans le travail de Van de Vonder et al. (2007), une étude comparative entre les approches prédictives/proactives-réactives a été proposée. Dans cette étude, deux méthodes prédictives et une méthode proactive ont été combinées avec quatre méthodes réactives. L'étude comparative de ces trois approches a été expérimentée sur la variation de trois paramètres : le niveau d'incertitude (élevé ou faible), la stabilité de l'ordonnancement et la déviation du makespan final par rapport à la solution de référence. Les résultats obtenus favorisent l'utilisation des méthodes proactives-réactives avec un maximum de robustesse afin de mieux réagir face aux perturbations durant la phase réactive de l'ordonnancement.

Dans Lambrechts et al. (2008), les auteurs ont traité le problème d'ordonnancement de projet avec ressources limitées dans un environnement instable caractérisé par des pannes de ressources. Les auteurs ont proposé huit stratégies pour la construction d'un ordonnancement proactif robuste et stable. La génération de cet ordonnancement est basée sur le calcul historique et probabiliste de la disponibilité moyenne des ressources. Une flexibilité temporelle et additionnelle est ajoutée à cet ordonnancement proactif par le biais des temps morts afin de renforcer la stabilité de l'ordonnancement. L'introduction des temps morts entre les opérations est basée sur l'estimation des impacts attendus sur les durées opératoires en raison de pannes de ressources. Les résultats expérimentaux ont montré l'intérêt d'utiliser une méthode réactive de type recherche tabou afin d'amortir l'impact des perturbations sur la faisabilité de l'ordonnancement proactif.

Une autre étude plus approfondie en relation avec les approches proactives-réactives se trouve dans Bidot et al. (2008). Dans cette étude, les auteurs considèrent aussi un autre type d'approches dites progressives dont l'idée est de relier les deux phases d'ordonnancement en résolvant le problème fragment par fragment. Les auteurs ont proposé un modèle conceptuel d'un système unique combinant les approches proactives, réactives et progressives. Ce système est basé sur deux modèles principaux : un modèle de génération qui permet la révision d'un ordonnancement et un modèle d'exécution qui exécute réellement l'ordonnancement. Ces deux modèles permettent de répondre aux "comment" et "quand" réviser un ordonnancement. Une perturbation causée par l'arrivée d'une nouvelle activité par exemple est contrôlée par des conditions réactives permettant d'enclencher le modèle de génération afin d'adapter le système au nouveau changement. Ces modèles généraux et paramétrables sont conçus pour servir d'une manière plus générale à d'autres problèmes diversifiés.

## 3.3 Quelle méthode proactive-réactive ?

Il est à noter que dans ce type de méthodes, l'approche proactive exigera toujours une technique réactive permettant de protéger la faisabilité de l'ordonnancement de référence face aux différentes perturbations. Le nombre d'interventions de la technique réactive est inversement proportionnelle à la robustesse de l'ordonnancent de

référence. La plupart des méthodes efficaces de la littérature se focalise sur la génération d'une solution hors-ligne flexible caractérisant plusieurs ordonnancements. Cette solution est représentée par un ordre partiel définit comme suit :

### 3.3.1 Ordre partiel

D'une manière générale, un ensemble $A$ représente un ordre partiel si la relation entre les éléments de $A$ est une relation réflexive, antisymétrique et transitive. Dans un problème d'ordonnancement, $A$ représente l'ensemble des opérations dans une même machine et la relation entre ces opérations est représentée par exemple par les contraintes de conjonction et disjonction entre ces opérations. Dans ce cas, un ensemble de nœuds d'ordre partiel doit être acyclique et doit contenir des arcs conjonctifs et des arcs disjonctifs. Si un graphe est d'ordre partiel et entre chaque paire de nœuds est défini un arc conjonctif, le graphe pourra alors être vu comme une chaîne de nœuds et il est dit d'ordre total comme représenté dans la figure 3.4. D'une manière générale, contrairement au graphe d'ordre total qui représente une relation linéaire entre les opérations, un graphe d'ordre partiel permet de représenter plusieurs solutions et non pas une seule.

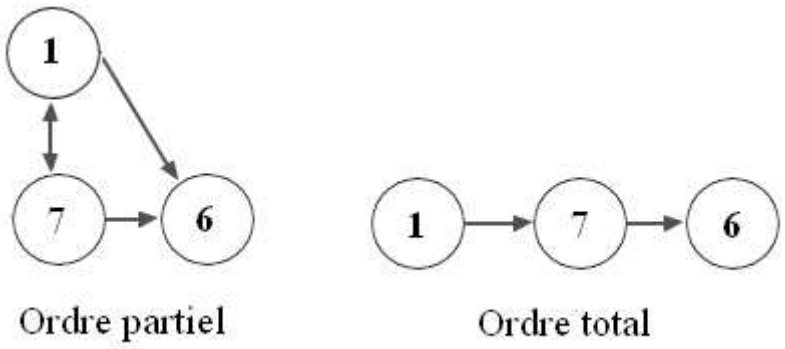



FIGURE 3.4 – *Ordre total VS Ordre partiel*

**Ordre partiel avec flexibilité temporelle**

Dans ce contexte, Policella et al. (2004) traitent le problème d'ordonnancement de projet avec contraintes de ressources. Dans ce travail, deux heuristiques ont été proposées pour la génération d'un ordonnancement partiel robuste et stable. La génération de cet ordonnancement partiel est faite sur deux étapes ; en premier lieu, un ordonnancement avec ordre total et des dates de début/fin fixes est généré. Ensuite, une flexibilité temporelle est introduite dans cet ordonnancement en utilisant un algorithme de chaînage. Le principe de cet algorithme consiste à lier les activités exécutées sur les mêmes ressources par des contraintes de chaînage avec des laps de temps mort au lieu de simples contraintes de précédence. Les deux heuristiques proposées

sont basées sur une procédure itérative de recherche d'échantillonnage afin d'explorer l'espace des ordonnancements partiels possibles et par conséquence construire un ordonnancement flexible et robuste.

En se basant sur la procédure proposée par Policella et al. (2004), les auteurs dans Fu et al. (2010) ont proposé une méthode de recherche locale pour le problème d'ordonnancement de projet avec contraintes de latences minimales/maximales du temps. Dans l'algorithme, la recherche locale est effectuée sur les listes des activités. Une liste d'activités est considérée comme la séquence de toutes les activités satisfaisant les contraintes de latences minimales et maximales. Par la suite, chaque activité est prévue de manière séquentielle en fonction de sa position dans la liste des activités et l'ordonnancement admissible partiel est généré. Au final, l'ordonnancement partiel choisi est celui qui a la valeur de fitness la plus intéressante. Cette mesure de fitness est liée à la distribution incertaine des durées opératoires des activités.

Une autre approche proactive-réactive nommée "TPOPEXEC" pour la planification et l'ordonnancement d'une manière générale a été proposée dans Muise et al. (2013). Cette approche est basée sur deux modules complémentaires. Le premier module "COMPILER" construit un ordonnancement hors-ligne robuste exposé comme une représentation générale de l'ordonnancement. Cette représentation constitue un ordre partiel entre les activités liées par des contraintes temporelles flexibles. Cet ordonnancement partiel est ensuite exécuté par le module "EXECUTOR" qui consiste à sélectionner parmi un nombre important de solutions, un ordonnancement admissible satisfaisant toutes les contraintes temporelles des activités et cohérent avec les nouvelles données du problème. L'efficacité de TPOPEXEC est démontrée par le biais d'un prototype expérimental afin de tester la robustesse et la flexibilité de cette méthode qui a été comparée avec une simple approche temporelle. La particularité de l'approche proposée est démontrée par sa capacité de réduire la nécessité de re-planification due au changement des données.

Une autre technique proactive-réactive est présentée dans Wua et al. (2009). Dans ce travail, les auteurs considèrent le problème d'ordonnancement d'atelier d'une seule machine avec minimisation du flowtime (temps d'écoulement). Les durées opératoires du problème sont définies par une variable aléatoire suivant une loi de distribution normale. Les auteurs ont proposé une approche $\beta$-robust. Cette approche, introduite par Daniels et Carrillo (1997), permet de maximiser la probabilité d'atteindre un niveau de performance donné. Les auteurs ont proposé un modèle de contraintes initiales pour résoudre ce problème ($\beta$-*robust*) en introduisant de la flexibilité séquentielle dans l'ordonnancement de référence. Cette flexibilité pourra ensuite être exploitée par un algorithme réactif, une fois que l'ordonnancement sera mis en place.

**Ordre partiel avec flexibilité séquentielle**

La plupart des approches précédentes se focalise sur la flexibilité temporelle de l'ordonnancement de référence. D'autres approches se focalisent sur la flexibilité sé-

quentielle entre les opérations. Dans ce type d'approches, la flexibilité temporelle est introduite implicitement avec la flexibilité séquentielle et d'autres mesures de robustesse sont proposées. La notion de groupes d'opérations permutables est généralement utilisée dans ce type d'approches (Billaut et al. 2008, Artigues et al. 2016). Le principe est d'introduire de la flexibilité séquentielle entre certaines opérations qui doivent être exécutées sur la même machine. Ces opérations appartiennent à des groupes qui représentent un ordonnancement d'ordre partiel. Dans ce type d'approches, et bien qu'elles permettent de représenter un ensemble de solutions admissibles sans les énumérer, deux avantages principaux sont généralement présents : elles garantissent une qualité minimale de l'ordonnancement durant son exécution, cette qualité est représentée par la pire performance et peut être calculée en temps polynomial (Artigues 2004). Le deuxième avantage principal est qu'elles favorisent la coopération homme-machine durant la phase d'exécution de l'ordonnancement et permettent de réagir intelligemment face aux perturbations.

Dans ce contexte, La (2005) a proposé une méthode robuste pour le problème à une seule machine avec contraintes de disponibilités. Cette méthode est basée sur la construction d'un ordonnancement de groupes d'opérations permutables en se basant sur la notion de structure d'intervalle. La caractéristique de cette méthode par rapport aux autres algorithmes de construction d'un ordonnancement de groupes est qu'elle génère des séquences de solutions dominantes de telle sorte que le meilleur et le pire ordonnancement possible sont connues *a priori*. Afin de réduire l'espace de solutions dominantes, une procédure de séparation et d'évaluation réduisant l'intervalle des opérations a été proposée.

Dans la suite de ce mémoire, nous nous focalisons sur deux aspects principaux : le meilleur des cas d'un ordonnancement de groupes ainsi que la coopération homme-machine. Dans la partie suivante, nous formalisons la problématique du meilleur des cas d'un ordonnancement de groupes et nous définissons l'intérêt et les méthodes de résolution proposées.

# Deuxième partie

# Ordonnancement de groupes

# Groupes d'opérations permutables

Ce chapitre présente la méthode de groupes d'opérations permutables. Nous commençons par l'introduction des différentes formules et évaluations relatives à cette méthode. Ensuite, nous mettons l'accent sur certains algorithmes de construction des groupes. Nous choisissons d'utiliser l'algorithme de construction EBJG pour les expérimentations réalisées dans ce travail.

## 4.1 Définition

L'ordonnancement de groupes également nommé *groupes d'opérations permutables* est une des méthodes proactives-réactives les plus étudiées dans la littérature comme mentionné dans le chapitre précédent. Elle a été introduite au LAAS/CNRS de Toulouse par Jacques Erschler, François Roubellat et Véronique Thomas la fin des années 80 (Erschler 1989).

Cette méthode consiste à proposer comme solution du problème d'ordonnancement, non pas des séquences totales d'opérations sur les machines considérées, mais un ordre partiel des séquences de groupes d'opérations de telle sorte que toute permutation des opérations dans chaque groupe représente un ordonnancement admissible. Cette approche a donné lieu à plusieurs thèses Thomas (1980), Billaut (1993), Artigues (1997), Esswein (2003), Le Gall (1989), Pinot (2008) ainsi que de nombreux articles (Artigues et al. 2005, Aloulou et Artigues 2010, Cardin et al. 2013) pour ne citer que les plus récents.

L'ordonnancement de groupes consiste à construire un ***ordonnancement flexible*** caractérisant un ensemble de solutions sans les énumérer. Cette représentation permet de choisir la solution la mieux adaptée à l'état réel de l'atelier lorsque l'ordonnancement de référence sera exécuté dans l'atelier. Cette méthode est utilisée selon trois étapes :

— Étape 0 : consiste à résoudre explicitement le problème combinatoire d'ordonnancement en utilisant les outils de la recherche opérationnelle telle que la méthode exacte de séparation et d'évaluation décrite dans la section 2.3.1. Dans cette étape, l'ordonnancement de référence est construit hors-ligne.

— Étape 1 (Étape de construction des groupes) : une certaine flexibilité séquentielle est introduite dans l'ordonnancement de référence de telle sorte que la solution flexible obtenue représente un ordonnancement d'ordre partiel au

problème d'ordonnancement disjonctif et caractérise une famille d'ordonnancements sans les énumérer. La solution flexible contient des groupes d'opérations totalement permutables et elle est construite hors-ligne en utilisant généralement un algorithme de regroupement.

— Étape 2 (Étape de décision) : une fois que l'ordonnancement de référence flexible est mis en place dans l'atelier, des décisions doivent être prises dans les groupes ; chaque décision consiste à choisir la prochaine opération à exécuter dans un groupe d'opérations permutables. Ces décisions sont prises en temps réel par une heuristique ou un humain (opérateur). Lorsqu'on parle de temps réel dans ce manuscrit, on se réfère à des délais imposés de l'ordonnancement, mesurés selon l'unité de temps utilisée dans ce dernier. Dans Monfared et Yang (2007), les auteurs considèrent trois phases et définissent pour chaque phase une mesure de temps : La première phase concerne la planification à long et moyen terme où la période de mesure du temps est généralement *semaines*. La deuxième phase est la phase d'ordonnancement à capacité finie centrée sur l'optimisation avec *heures* comme unité de temps. La troisième phase appelée supervision (contrôle) est destinée à la prise de décision en temps réel avec une unité de temps en secondes. Nous supposons dans ce manuscrit que, une à cinq secondes sont des délais acceptables pour prendre une décision durant cette phase en-ligne de l'ordonnancement de groupes.

### 4.1.1 Formulation

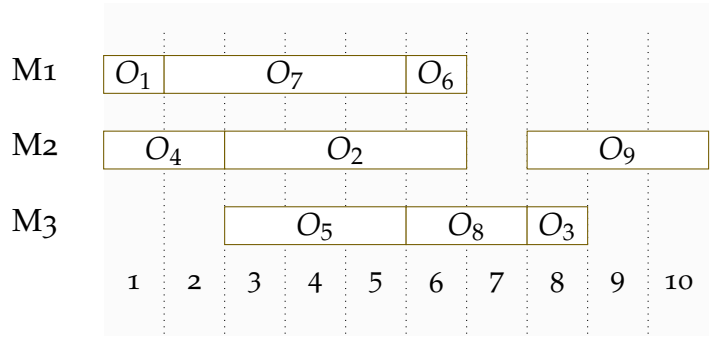
Un ordonnancement de groupes est défini par un ensemble de séquences de groupes à exécuter sur les machines. Un élément de cet ensemble est appelé *groupe d'opérations permutables*.

Un ordonnancement de groupes $(G_i^k)_{i=1,\dots,\nu_k}^{k=1,\dots,m}$ avec $\cap_{i=1}^{\nu_k} G_i^k = \emptyset$ et $\cup_{i=1}^{\nu_k} G_i^k = \mathcal{J}$ (tel que $\mathcal{J}$ représente l'ensemble des jobs du problème) est un ordonnancement partiel représenté par un graphe disjonctif spécifiant pour chaque machine $M_k$ une séquence de $\nu_k$ groupes d'opérations permutables, tel que $G_i^k$ représente le $i^{eme}$ groupe sur la machine $M_k$ et peut contenir au moins une opération, et au plus autant d'opérations à réaliser sur cette machine. $g_j^k$ représente la position du groupe contenant l'opération $O_{k,j}$ dans la machine $k$. $G^k_{g_j^k-1}$ (resp. $G^k_{g_j^k+1}$) est le groupe prédécesseur (resp. successeur) sur la machine $M_k$.

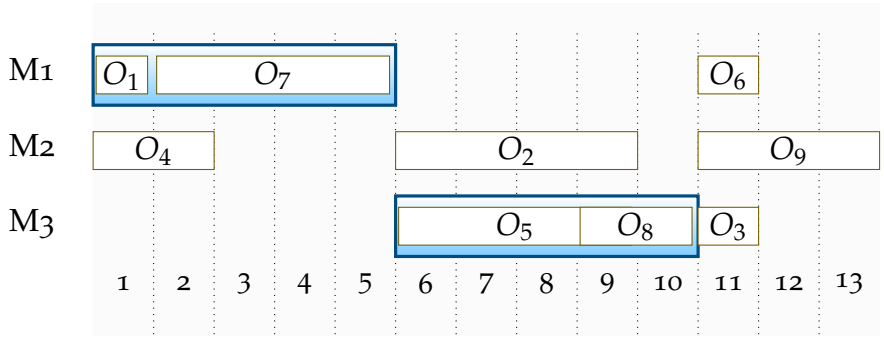
Un ordonnancement de groupes est *admissible* si chaque permutation des opérations d'un même groupe donne un ordonnancement admissible qui satisfait toutes les contraintes du problème. Les critères de performance d'un ordonnancement de groupes sont les mêmes que pour un ordonnancement classique. Cependant, la qualité d'un ordonnancement de groupes est mesurée par la qualité dans le pire des cas, ce qui correspond au pire ordonnancement semi-actif parmi tous les ordonnancements possibles (Artigues et al. 2016). Nous reviendrons sur la qualité d'un ordonnancement de groupes dans la section 4.1.3.

### 4.1.2 Exemple

Pour illustrer cette définition, nous reprenons l'exemple décrit dans la section 2.2. La figure 4.1 présente l'étape 0 et l'étape 1 de l'ordonnancement de groupes solution du problème. Cet ordonnancement de groupes (ordonnancement de référence flexible) est composé de sept groupes : deux groupes de deux opérations $G_1^1$ et $G_1^3$ et cinq groupes d'une seule opération.




(a) *Étape 0*




(b) *Étape 1*

FIGURE 4.1 – *Ordonnancement de groupes*

Cet ordonnancement de groupes pourra être également représenté par un graphe disjonctif comme illustré dans la figure 4.2. Dans cette représentation, deux cliques sont présentes : $(O_1, O_7)$ et $(O_5, O_8)$.

Durant la phase de décision (étape 2), les groupes d'opérations permutables sont exécutés en-ligne. Cette exécution consiste à sélectionner le meilleur ordonnancement adapté à l'état réel de l'atelier. Dans notre exemple, nous avons deux décisions à prendre : la première décision concerne le groupe $G_1^1$ et la deuxième décision concerne le groupe $G_1^3$. Selon les décisions prises, l'ordonnancement admissible est un des quatre ordonnancements semi-actifs représentés dans la figure 4.3. Il est à noter que la valeur de l'objectif de l'ordonnancement de chacune des solutions possibles est différente. Dans cet exemple et pour le $C_{max}$, nous avons un ordonnancement optimal égal à 10 et deux ordonnancements avec la pire valeur égale à 12.

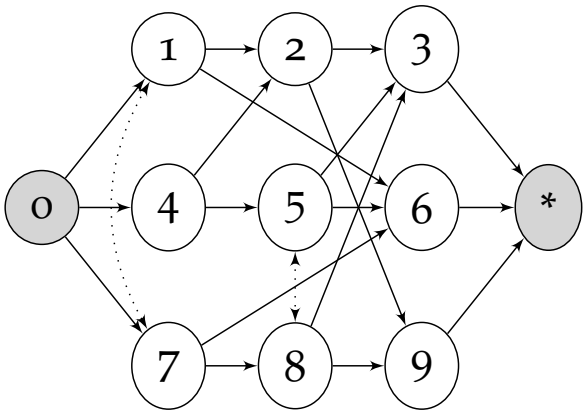



FIGURE 4.2 – *Graphe disjonctif de l'ordonnancement de groupes.*

### 4.1.3 Qualité d'un ordonnancement de groupes

Le pire des cas est utilisé pour évaluer la qualité d'un ordonnancement de groupes. Ce pire des cas représente le pire ordonnancement semi-actif caractérisé par l'ordonnancement de groupes et permet d'assurer une performance minimale en cas de perturbations durant l'étape 2 de l'ordonnancement de groupes. Cette mesure est très utile en pratique car le décideur préfère souvent se prémunir contre des situations délicates plutôt que de chercher à trouver la solution optimale. Ce paramètre peut aussi jouer un rôle informationnel important dans le processus de construction d'un ordonnancement de groupes (section 4.1.5).

Pour calculer le pire des cas dans un graphe disjonctif, nous avons besoin de calculer les dates de fin maximales des opérations présentes dans l'ensemble des ordonnancements semi-actifs décrits par un ordonnancement de groupes. On distingue deux types d'ordonnancements semi-actifs dans le pire des cas : l'ordonnancement au plus tôt et l'ordonnancement au plus tard.

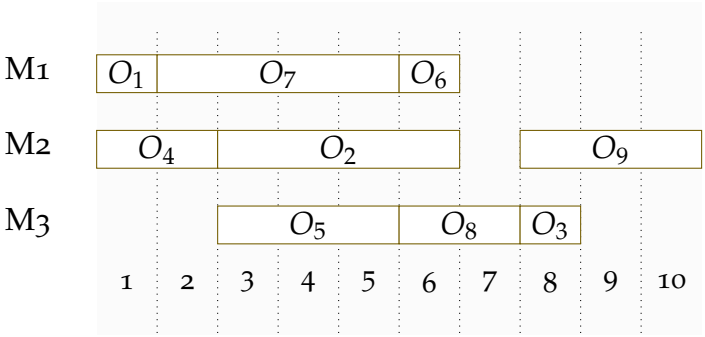
#### Le pire ordonnancement au plus tôt

Pour calculer le pire ordonnancement au plus tôt, nous avons besoin de calculer la date de fin au plus tôt $\underline{C}_{k,j}$ et par conséquent la date de début au plus tôt $\underline{\tau}_{k,j}$ de toutes les opérations.

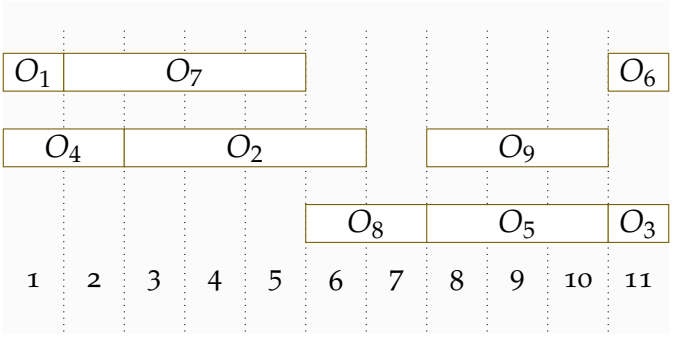
Par ailleurs, le calcul de $\underline{\tau}_{k,j}$ dans le groupe $G_i^k$ correspond à l'exécution de cette opération dans la première position de son groupe, sous l'hypothèse que les opérations prédécesseurs finissent le plus tard possible. La formulation de cette évaluation est présentée comme suit :

$$\underline{\tau}_{k,j} = \max(r_{k,j}, \underline{C}_{\pi_j^k,j}, \max_{O_{k,l} \in G_{i-1}^k} \underline{C}_{k,l}) \tag{4.1}$$

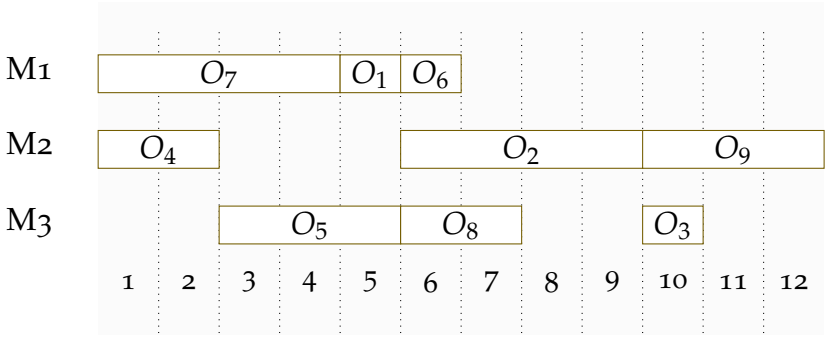
Pour calculer $\underline{C}_{k,j}$ du groupe $G_i^k$, soit la date de fin dans le pire des cas de l'opération $O_{k,j}$ ne dépend pas d'une autre opération du même groupe, dans ce cas, le premier terme de la formule 4.2 est utilisée, sinon, dans le pire des cas, l'opération se



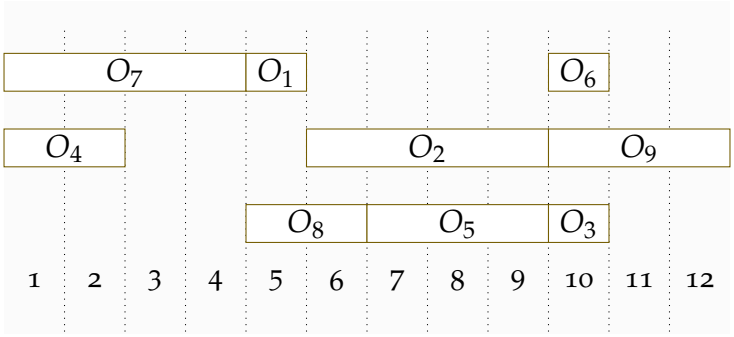

(a) $C^*_{max} = 10$




(b) $C_{max} = 11$




(c) $C_{max} = 12$




(d) $C_{max} = 12$

FIGURE 4.3 – *Ordonnancements admissibles semi-actifs*

termine à la dernière position de son groupe et le second terme est utilisé. La figure 4.4 illustre les deux cas possibles.

$$\underline{C}_{k,j} = \max(\underline{\tau}_{k,j} + p_{k,j}, \max_{O_{k,l} \in G_i^k, l \neq j} \underline{\tau}_{k,l} + \sum_{O_{k,l} \in G_i^k} p_{k,l}) \tag{4.2}$$

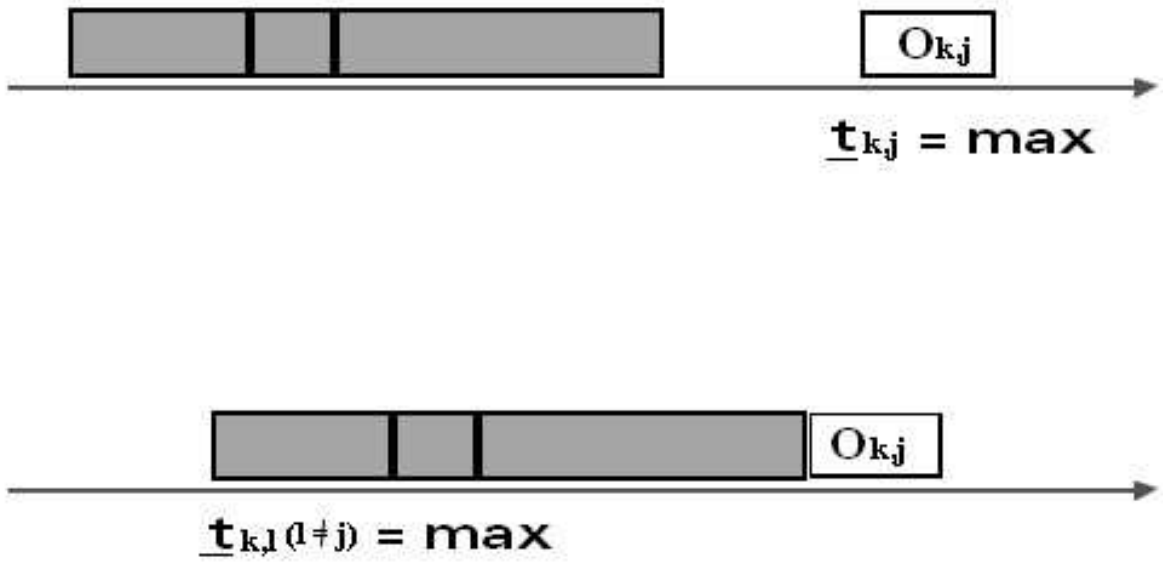



FIGURE 4.4 – *La date de fin au plus tôt dans le pire des cas*

Finalement, quel que soit l'ordre des opérations dans les groupes la formule $C_{k,j} \geq \underline{C}_{k,j}$ est garantie. Pour un objectif régulier de type *MinMax* comme le $C_{max}$ par exemple, $\max_{\forall J_i \in \mathcal{J}, \forall M_k \in \mathcal{M}}(\underline{C}_{k,j})$ représente le pire ordonnancement au plus tôt (le pire des cas).

**Le pire ordonnancement au plus tard**

Le calcul du pire ordonnancement au plus tard pour un objectif régulier est similaire au calcul précédent avec des dates d'échéances en plus. L'objectif est de calculer une borne supérieure (date d'échéance) de la date de fin au plus tard de chaque opération afin d'une part de ne pas retarder davantage les jobs déjà en retard et d'autre part de terminer les jobs qui ne sont pas en retard avant leurs dates d'échéances.

Pour cela, il faut calculer la date de fin au plus tard $\overline{C}_{k,j}$ et par conséquent la date de début au plus tard $\overline{\tau}_{k,j}$ de chaque opération $O_{k,j}$. Une version symétrique des équations (4.1) et (4.2) peut être utilisée. Plutôt que de commencer le calcul de la première opération du graphe disjonctif, le calcul commence à partir de la fin en utilisant les dates d'échéances des jobs comme expliqué dans Thomas (1980), Pinot (2008).

$\overline{\tau}_j^k$ et $\overline{C}_j^k$ peuvent être exprimées comme suit :

$$\overline{C}_{k,j} = \min(d_{k,j}, \overline{\tau}_{\Omega_j^k,j}, \min_{O_{k,l} \in G_{i+1}^k} \overline{\tau}_{k,l})$$

$$\overline{\tau}_{k,j} = \min(\overline{C}_{k,j} - p_{k,j}, \min_{\forall O_{k,l} \in G_i^k, l \neq j} \overline{C}_{k,l} - \sum_{\forall O_{k,l} \in G_i^k} p_{k,l}) \quad (4.3)$$

La qualité d'un ordonnancement de groupes est représentée par le pire des cas. Pour un critère régulier, ce pire des cas représente la pire date de fin $C_{k,j}$ qu'une opération peut prendre dans l'ensemble des ordonnancements semi-actifs décrit par l'ordonnancement de groupes. Cette valeur est calculée en temps polynomial pour les critères réguliers de type *MinMax* comme le $C_{\max}$ (Esswein 2003, Aloulou et al. 2004, Artigues et al. 2005). Cependant, cette évaluation représente seulement une borne supérieure pour les critères réguliers de type *MinSum*.

De manière similaire à cette évaluation, on peut également chercher à évaluer la qualité dans le meilleur des cas. Une telle évaluation n'est pas triviale, elle est même NP-difficile (Esswein 2003, Pinot 2008). Le chapitre suivant est dédié à cette évaluation.

### 4.1.4 Flexibilité d'un ordonnancement de groupes

Différentes mesures de flexibilité sont proposées dans la littérature pour l'ordonnancement de groupes. On distingue deux catégories : des mesures de flexibilité temporelle et des mesures de flexibilité séquentielle.

#### Flexibilité temporelle

Pour ce premier type de flexibilité, nous citons l'approche proactive-réactive *ORABAID* (Ordonnancement d'Atelier Basé sur l'AIde à la Décision) développée au LAAS-CNRS Lopez et Roubellat (2008). Cette approche utilise la notion de la marge libre séquentielle. Cette marge calcule pour une opération, le retard maximum qui garantit qu'aucun ordonnancement semi-actif caractérisé par l'ordonnancement de groupes ne sera en retard.

La marge libre séquentielle d'une opération dans un groupe est dénotée $m_{\text{seq}}(O_{k,j})$ et a deux éléments :

— La marge nette $m_{\text{sn}}(O_{k,j})$, qui est liée à l'opération elle-même, indépendamment des autres opérations du groupe.
— La marge de l'opération dans le groupe $m_{\text{sg}}(O_{k,j})$, qui est liée aux autres opérations du même groupe.

Le calcul de la marge libre séquentielle d'une opération correspond à la différence entre la pire date de début au plus tard $\overline{\tau}_{k,j}$ (4.3) et la pire date de début au plus tôt $\underline{\tau}_{k,j}$ (4.1).

En utilisant les équations de la date de début au plus tôt et la date de début au plus tard dans le pire des cas, la marge libre séquentielle s'exprime comme suit :

$$\begin{cases} m_{\text{seq}}(O_{k,j}) = \min(m_{\text{sn}}(O_{k,j}), m_{\text{sg}}(O_{k,j})) \\ m_{\text{sn}}(O_{k,j}) = \overline{C}_{k,j} - p_{k,j} - \underline{\tau}_{k,j} \\ m_{\text{sg}}(O_{k,j}) = \min_{\forall O_{k,l} \in G_i^k, l \neq j} \overline{C}_{k,j} - \sum_{\forall O_{k,l} \in G_i^k} p_{k,l} - \underline{\tau}_{k,j} \end{cases} \tag{4.4}$$

Cette mesure de flexibilité représente peu d'intérêt pour l'étape 1, la construction des groupes, puisque celle-ci est basée sur la flexibilité séquentielle. Mais la marge libre séquentielle s'avère beaucoup plus utile lors de l'étape 2, le choix d'une opération dans un groupe d'opérations permutables, étape que nous discuterons dans la troisième partie de ce manuscrit.

**Mesures de flexibilité séquentielle**

Dans Artigues et al. (2005), une mesure de flexibilité représentant le nombre de groupes est utilisée et dénotée $\#Gps$. En effet, moins il y a de groupes, plus il y a d'opérations par groupe et plus le choix (la flexibilité) est important durant l'étape 2 de la prise de décision.

$$\#Gps = |\sum_{i=1,\ldots,\nu_k}^{k=1,\ldots,m} G_i^k|$$

Deux autres mesures de flexibilité intéressantes sont proposées dans Esswein (2003) et Artigues et al. (2005). La première mesure concerne le nombre d'ordonnancements semi-actifs décrits par l'ordonnancement de groupes dénotée $\#Seq$. Une fois que les groupes sont construits, il est facile de calculer le nombre d'ordonnancement semi-actifs caractérisé par cet ordonnancement de groupes :

$$\#Seq = \Pi_{i=1,\ldots,\nu_k}^{k=1,\ldots,m}(|G_i^k|!)$$

La deuxième mesure est relative au nombre de groupes et la taille du problème, et est la plus utilisée dans la littérature. Cette mesure est notée $\#\phi$ et est donnée par la formule suivante :

$$\#\phi = \frac{(nXm) - \#Gps}{(nXm) - m}$$

Dans ce contexte, une flexibilité minimale de 0 (ou 0%) correspond à autant de groupes que d'opérations. Une flexibilité maximale de 1 (ou 100%) (ou plutôt une borne maximale pour la flexibilité) a un seul groupe par machine, ce cas pourra être présent dans un problème de flow-shop. Pour notre exemple représenté dans la figure 4.1(b) $\#\phi = 1/3$.

### 4.1.5 Construction des groupes

Pour la construction d'un ordonnancement de groupes flexible, Billaut (1993), Artigues (1997) proposent une méthode en trois étapes : une étape de construction d'un ordonnancement de groupes admissible, une étape cherchant à respecter les contraintes et une dernière étape d'amélioration de la flexibilité introduite.

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 de recherche tabou. A la fin de cette étape, les dates de livraisons non respectées sont actualisées à la date de fin du pire des cas de la dernière opération 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 livraison.

Esswein (2003) a proposé trois algorithmes de construction de groupes d'opérations permutables pour le problème du job shop. L'idée de ces trois algorithmes est de construire, à partir d'un ordonnancement de référence, un ordonnancement de groupes le plus flexible possible en minimisant le $C_{max}$ ; cette flexibilité est mesurée par le nombre de groupes caractérisant la solution de l'ordonnancement de groupes ($\#Gps$) ; plus le nombre de groupes est faible et plus l'ordonnancement est flexible. Le premier algorithme proposé, nommé *OGAJ*, est basé sur la méthode du shifting bottleneck. L'idée d'*OGAJ* est de construire un ordonnancement de groupes avec autant de groupes que d'opérations en résolvant le problème machine par machine. Le deuxième algorithme proposé est un algorithme génétique. Les paramètres de cet algorithme sont définis dans Esswein et al. (2003). Le troisième algorithme, nommé *EBJG*, est un algorithme polynomial de construction progressive. La question laquelle on doit répondre dans cet algorithme glouton est : *sur la machine* $M_k$*, doit-on affecter l'opération en* $i^{ème}$ *position dans le même groupe que l'opération en* $(i+1)^{ème}$ *position ?* Les expérimentations menées sur les trois algorithmes ont montré que l'algorithme *OGAJ* malgré sa rapidité, est le moins performant. Par contre, l'algorithme génétique et *EBJG* ont donné des résultats similaires pour les problèmes avec moins de 200 opérations. Cependant, dès que le nombre d'opérations est supérieur à 200, *EBJG* devient plus performant.

Pinot (2008) a proposé trois améliorations pour l'algorithme *EBJG* : une simplification d'implémentation, une amélioration diminuant le temps d'exécution et une modification permettant d'obtenir un ensemble de groupes dominants plutôt qu'une solution unique.

Nous rappelons que la méthode d'ordonnancement de groupes est une méthode proactive-réactive dont l'étape 0 et l'étape 1 pour la construction des groupes sont incluses dans la phase proactive. Durant cette phase, il est sans doute préférable de ne regrouper d'abord que les opérations susceptibles d'être perturbés en se basant par exemple sur des statistiques préalables de l'atelier. Dans cette thèse, nous nous intéressons plus particulièrement à l'étape 2 de l'ordonnancement de groupes. Nous

considérons donc que l'ordonnancement de référence de l'étape 0 et l'étape 1 sont des données initiales de notre problématique, peu importe l'algorithme de regroupement. Cependant, pour toutes les expérimentations réalisées dans ce manuscrit, nous avons utilisé l'algorithme *EBJG* pour la construction des groupes. Cet algorithme est détaillé dans la suite de cette section.

**Algorithme *EBJG***

L'algorithme de construction progressive *EBJG* consiste à prendre un ensemble de décisions successives d'une manière gloutonne (sans remettre en cause les décisions déjà prises). Cet algorithme transforme un ordonnancement de référence où chaque groupe contient une seule opération (Fig. 4.1.a) en un ordonnancement de groupes plus flexible où les groupes contiennent plus d'opérations (Fig. 4.1.b).

Chaque décision consiste à choisir le meilleur regroupement admissible. Un regroupement admissible est un fusionnement entre deux groupes consécutifs dont aucune relation de précédence n'est présente entre les opérations de ces deux groupes et dont le plus long chemin ($C_{max}$) ne dépasse pas une certaine valeur donnée $\epsilon$. Le regroupement ayant la plus petite valeur du $C_{max}$ est sélectionné.

---

**Algorithme 1 :** L'algorithme de construction des groupes d'opérations permutables (Esswein 2003)

---

1. Soit $S$ une solution du problème de job shop
   Soit $G(S)$ l'ensemble des groupements (initialement vide) de l'ordonnancement de groupes $S$, c'est-à-dire l'ensemble des décisions prises pour définir $S$ ;
   encore := Vrai ;
2. **while** *encore* **do**
   Construire l'ensemble $\xi$ des solutions $S^{'}$ admissibles et telles que
   $G(S) \subset G(S^{'})$,
   $|G(S^{'})| = |G(S)| + 1$ et $C_{max}(S^{'}) \leq \epsilon$ ;
   **if** $\xi \neq 0$ **then**
      Parmi les solutions de $\xi$ de plus faible $C_{max}$, soit $S^{*}$ celle de plus petit nombre de groupes en contenant le plus d'opérations ;
      $S := S^{*}$ ;
   **else**
      encore := Faux ;

---

Ce processus de décision pourra être vu comme un algorithme de recherche locale tel que le voisinage de $S$ représenté par $S^{'}$ est l'ensemble des regroupements admissibles où le nombre de groupes de $S^{'}$ est égale au nombre de groupes de $S$ +1. En pratique, la complexité de l'algorithme *EBJG* est $(n-1)$ x $m$ itérations, ce qui est polynomial (Esswein 2003).

Une fois que l'ordonnancement de groupes est généré, il est important d'évaluer

ce dernier afin de profiter au maximum des avantages de cette approche, telle que l'aide à la décision. Nous avons décrit l'évaluation du pire des cas représenté par la pire permutation dans chaque groupe menant à la pire performance possible. Dans le chapitre suivant, nous allons nous intéresser à un autre type de performance : le meilleur des cas.

# Le meilleur des cas

Le meilleur des cas est un problème d'optimisation qui consiste à trouver **le meilleur ordonnancement semi-actif** décrit par l'ordonnancement de groupes. Cette évaluation pourra être très intéressante pour différentes raisons :

— La qualité de l'ordonnancement de groupes pourra être mesurée par un intervalle au lieu d'une seule mesure ; le pire des cas, comme souligné par Esswein (2003). Cet intervalle donne l'étendue des valeurs possibles de l'ordonnancement réalisé et est borné comme suit : [*meilleur des cas ... pire des cas*].

— Le meilleur des cas permet aussi de savoir, une fois que les groupes d'un ordonnancement de groupes sont générés, la meilleure performance atteignable en cas d'absence de perturbations sur l'atelier.

— La raison la plus importante c'est qu'avec le meilleur des cas, le décideur peut choisir, lors de l'exécution de l'ordonnancement de groupes (étape 2), l'opération optimale à exécuter en premier, dans un groupe d'opérations permutables. Cette information est très utile dans un système d'aide à la décision. Ce point est traité en détail dans la troisième partie de ce manuscrit.

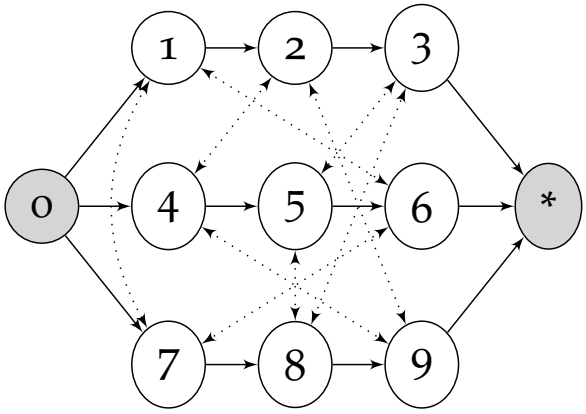
Le problème du meilleur des cas consiste à calculer les $C_{k,j}$ des opérations du meilleur ordonnancement possible, ce qui est similaire au calcul du plus long chemin du problème classique du job shop et est NP-difficile (Pinot 2008).

La différence entre un problème de meilleur des cas et un problème classique du job shop se résume dans le nombre d'arcs disjonctifs du graphe disjonctif ; dans un ordonnancement de groupes, certains des arcs disjonctifs du job shop sont remplacés par des arcs conjonctifs comme montré dans la figure 5.1. Dans cette figure, on voit bien que le graphe disjonctif de l'ordonnancement de groupes est un graphe partiel du graphe disjonctif du job shop ; Dans le problème de job shop initial (figure 5.1(a)), neuf décisions représentées par des arcs disjonctifs doivent être prises, par contre dans le graphe disjonctif de l'ordonnancement de groupes (figure 5.1(b)), sept de ces décisions sont déjà prises, ce qui simplifie le problème initial.

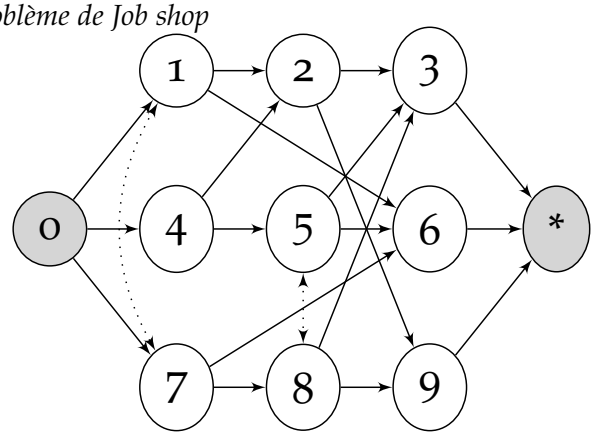
***Preuve que le meilleur des cas est un problème partiel du job shop***

— $\gamma_1 \neq \emptyset$ représente l'ensemble des arcs disjonctifs dans un problème de job shop $P$ de telle sorte qu'il n'existe aucun autre ensemble $\gamma_1^{'}$ tel que $\gamma_1 \subset \gamma_1^{'}$.

— $\gamma_2 = \emptyset$ représente l'ensemble des arcs disjonctifs pour un ordonnancement admissible du problème $P$. Cet ordonnancement représente l'ordonnancement de référence calculé dans l'étape 0 de l'ordonnancement de groupes.




(a) *Problème de Job shop*




(b) *Problème du meilleur des cas*

FIGURE 5.1 – *Problème du job shop VS Problème du meilleur des cas*

— Dans un problème d'ordonnancement de groupe $P^{'}$, les regroupements se font sur une même machine. L'ensemble des arcs disjonctifs notés $\gamma_3$ est inclu dans l'ensemble $\gamma_1$ puisque $\gamma_3$ est issue du problème $P$, donc le problème $P^{'}$ représenté par $\gamma_3$ est un sous problème du problème $P$ représenté par $\gamma_1$.

Dans ce type de problèmes (meilleur des cas), une méthode exacte comme la méthode de séparation et d'évaluation est généralement l'outil le plus utilisé dans la littérature pour trouver la solution optimale au problème. Cependant, dans un problème NP-difficile de grande taille, une méthode exacte s'avère très coûteuse en terme de temps, en particulier dans un système d'aide à la décision. Au lieu de cela, une borne inférieure efficace pourra être très utile pour donner une évaluation proche de l'optimale du meilleur des cas. Les mêmes techniques utilisées pour le calcul d'une borne inférieure pour un problème de job shop peuvent être utilisées pour un problème du meilleur des cas puisque le meilleur des cas est un sous-problème du job shop comme démontré précédemment.

Dans la section suivante, nous faisons une présentation succincte de certaines techniques efficaces pour le calcul d'une borne inférieure d'un problème de job shop. La majorité de ces techniques sont ensuite utilisées et adaptées à notre problématique du meilleur des cas.

## 5.1 Borne inférieure pour le problème de job shop

L'évaluation d'une borne inférieure pour le problème de job shop est généralement basée sur le calcul de deux valeurs :

— *La tête* $r^{'}_{k,j}$ (*Head*). Cette valeur représente une borne inférieure de la date de début au plus tôt d'une opération $O_{k,j}$ et est définie par le plus long chemin entre $O_0$ et $O_{k,j}$.

— *La latence* $q^{'}_{k,j}$ (*Tail*), qui représente une borne inférieure de la distance entre la date de fin de l'opération $O_{k,j}$ et la fin de l'ordonnancement ($O_*$).

Chaque opération ne peut commencer avant la date $r^{'}_{k,j}$, elle s'exécute pendant une durée $\rho_{k,j}$ sur la machine $k$ et l'ordonnancement se termine au moins avec une durée $q_{k,j}$ après la fin de l'exécution de l'opération. L'amélioration de $r^{'}_{k,j}$ et $q^{'}_{k,j}$ mène généralement à l'amélioration de la borne inférieure pour la fonction objective.

Le concept classique pour le calcul d'une borne inférieure d'un problème de job shop repose sur la relaxation du problème en $m$ différents problèmes à *une machine* reliés par les contraintes de précédences associées aux jobs afin de les résoudre séquentiellement et parallèlement. La valeur maximale/minimale de ces $m$ évaluations représente une borne inférieure du job shop.

Le JPS (*Jackson's Preemptive Schedule*) est un algorithme utilisé pour la résolution du problème à une machine avec préemption des jobs. La valeur obtenue par cet algorithme représente une borne inférieure pour le problème à une machine sans préemption. Cet algorithme utilise la règle MWR (Most Work Remaining) Jackson (1955) ; à chaque moment $t$ de l'ordonnancement, priorité est donnée au travail disponible qui

a la plus grande valeur de latence. Ce job est traité soit entièrement ou jusqu'à la disponibilité d'un autre job prioritaire. La complexité de cette règle est de $O(n \log n)$ et le makespan final est donné par l'équation :

$$JPS(\mathcal{J}) = \underbrace{Min}_{j\in\mathcal{J}} r_{k,j} + \underbrace{\sum_{j\in\mathcal{J}}}\rho_{k,j} + \underbrace{Min}_{j\in\mathcal{J}} q_{k,j} \tag{5.1}$$

La valeur maximale des évaluations $JPS(\mathcal{J})$ sur chaque machine $k$ représente une borne inférieure pour le problème de job shop tel qu'utilisé dans l'algorithme de séparation et d'évaluation de Morikawa et Takahashi (2009).

Une borne inférieure plus performante pourra être calculée à partir de l'algorithme JPS. Cette borne est basée sur la résolution optimale du problème $1|r_{k,j}|L_{\max}$ par la méthode de séparation et d'évaluation de Carlier (1982). Dans cette méthode, l'algorithme JPS est utilisé comme borne inférieure. L'algorithme de Schrage (1970) est utilisé comme solution initiale pour chaque nœud de l'arbre durant la procédure de séparation ; le job disponible à l'instant $t$ avec la plus grande durée de latence est ordonnancé en premier. En partant de cet algorithme, un chemin critique $Cr$ et une opération critique $Oc$ sont définis, l'algorithme de séparation est établi selon l'ordre de $Oc$ par rapport à $Cr$. La valeur maximale du makespan optimale pour chaque machine représente une borne inférieure pour le problème de job shop.

En utilisant la relaxation $m$ machines définie précédemment, Carlier et Pinson (1989) ont résolu pour la première fois le problème de job shop de 10 jobs et 10 machines, proposé initialement par Muth et Thompson. (1963). Les auteurs ont proposé un algorithme de séparation et d'évaluation sur la base de plusieurs propositions basées sur la représentation du problème par un graphe disjonctif. Ces propositions sont basées sur le concept de ***la sélection immédiate*** et permettent d'ajuster les têtes et les durées de latences des opérations liées par des contraintes disjonctives. Ce concept permet de définir des relations d'ordre entre les opérations appartenant à la même clique avant la mise en œuvre de la procédure de séparation de l'algorithme. Cette relation d'ordre pourra être fixée si l'ordre inverse de cette relation produit une solution qui n'améliore pas la meilleure solution trouvée à l'instant (la borne supérieure) ; Si $r_{k,j} + \rho_{k,j} + \rho_{k,i} + q_{k,i} \geq UB$ alors l'arc $O_{k,i} \rightarrow O_{k,j}$ est sélectionné de l'arc disjonctif $O_{k,i} \leftarrow\rightarrow O_{k,j}$.

Cette sélection immédiate permet d'ajuster les têtes et les durées de latences du problème de job shop. Dans une clique $C$, si la condition 5.2 (resp. 5.3) est vrai, alors l'opération $O_{k,o}$ (resp.$O_{k,e}$) est dite *sortie* (resp. *entrée*) de $C$. Cette opération doit être exécutée après (resp. avant) toute autre opération de la clique $C$. Cela permet d'ajuster les têtes et durées de latence des entrées/sorties de la clique $C$ comme montré dans Carlier et Pinson (1991).

$$\underbrace{Min}_{j\in C} r_{k,j} + \underbrace{\sum_{j\in C}}\rho_{k,j} + \underbrace{Min}_{j\in C/o} q_{k,j} \geq UB \tag{5.2}$$

$$\underbrace{Min}_{j \in C/e} r_{k,j} + \underbrace{\sum_{j \in C}} \rho_{k,j} + \underbrace{Min}_{j \in C} q_{k,j} \geq UB \tag{5.3}$$

Cette sélection immédiate a été étendue pour sélectionner les arcs disjonctifs entre les opérations triplées Dewess (1992). D'autres améliorations ont également été proposées par Brucker et al. (1994b) avec quelques modifications de l'algorithme initial réduisant le problème à des conditions plus faibles que celle proposées par Carlier et Pinson (1989), Dewess (1992). Cet algorithme a été généralisé pour permettre la sélection immédiate de certaines disjonctions dans une clique de $r$-opérations ($r \geq 2$), l'algorithme a une complexité de $O(n \log n, f)$, où $f$ représente le nombre de disjonctions à fixer dans une clique $C$. Cet algorithme a été utilisé dans une méthode de séparation et d'évaluation afin d'améliorer la résolution du problème 10 X 10 de Muth et Thompson. (1963).

D'autres améliorations sur le calcul des bornes inférieures basées sur le concept de la sélection immédiate ont été proposées dans Carlier et Pinson (1994). Des ajustements des têtes et durées de latences ont été améliorés d'une manière optimale en utilisant l'algorithme d'ordonnancement préemptif JPS sur une machine locale. Ensuite, des méthodes d'énumération ont été proposées en utilisant la sélection immédiate à la fois localement sur chaque machine et globalement pour toutes les machines. L'implémentation de l'algorithme est plus efficaces pour la réduction de l'espace de recherche mais avec beaucoup plus de consommation de temps par rapport à l'algorithme local.

Dans Brucker et Jurisch (1993), un autre type d'algorithme pour le calcul de la borne inférieure du job shop en utilisant les méthodes géométriques est proposé. Cet algorithme est basé sur une relaxation deux-Job avec préemption, avec une complexité de $O(n \log n)$ où $n$ représente le nombre d'opérations exécutées sur la machine. Cet algorithme est formulé comme un problème du plus court chemin dans un espace géométrique avec des objets rectangulaires comme obstacles. Les expérimentations menées sur des instances de benchmark ont montré que la relaxation deux-jobs domine les résultats de la relaxation sur une machine uniquement quand le rapport entre le nombre de machines et le nombre de jobs est supérieur à 1. Alors que, pour les autres instances lorsque le nombre de machines est inférieur ou égal au nombre de jobs, la relaxation sur une machine est nettement plus performante que la relaxation deux-jobs.

Un autre type de relaxation pour le calcul des bornes inférieures en se basant sur la programmation mathématique a été proposé récemment pour les objectifs de *min-sum* (Baptiste et al. 2008, Lancia et al. 2011, Tanaka et al. 2015). Dans Tanaka et al. (2015) par exemple, les auteurs ont proposé une relaxation lagrangienne combinée entre une relaxation au niveau des jobs et une relaxation au niveau des machines. Cette relaxation a permis de calculer des bornes inférieures et supérieures efficaces pour les objectifs *min-sum* dans un problème de job shop classique.

À notre connaissance, la plupart des bornes récentes sont dédiées à des problèmes particuliers ou étendus du problème classique de job shop ; comme le travail de Lobo

et al. (2013) et Brucker et al. (2012) pour ne citer que quelques uns. Dans Lobo et al. (2013) par exemple, les auteurs minimisent le retard maximum dans un problème de job shop avec une contrainte additionnelle. Cette contrainte consiste à affecter des travailleurs aux machines pour les faire fonctionner ; en pratique, le nombre de travailleurs n'est pas nécessairement égal au nombre de machines mais généralement inférieur, ce qui complique le problème classique de job shop. Ce problème est noté $N|M|L_{\max}$ avec contrainte d'allocation des travailleurs. Une solution à ce problème consiste à spécifier le groupe de machines auquel chaque travailleur est affecté ainsi que l'ordre d'exécution des jobs sur les machines. Les auteurs ont d'abord développé une borne inférieure pour un groupe de machines en utilisant la préemption des jobs. Ensuite, un algorithme de recherche optimale a été développé, cet algorithme consiste à chercher l'allocation optimale des travailleurs dans un groupe de machines. L'évaluation maximale des bornes sur les groupes de machines représente une borne inférieure pour le problème de job shop avec contrainte d'allocation des travailleurs.

## 5.2 Borne inférieure pour la date de début/fin au plus tôt des opérations dans le meilleur des cas

Le calcul d'une borne inférieure pour le meilleur des cas dans un ordonnancement de groupes n'est pas exactement similaire au calcul de la borne inférieure de la solution optimale du problème de job shop en raison des contraintes conjonctives entre les groupes sur les mêmes machines. En outre, le meilleur des cas n'est pas nécessairement l'ordonnancement optimal du problème initial de job shop.

Le calcul de ce meilleur des cas étant NP-difficile Pinot (2008) et afin d'éviter l'énumération exhaustive de tous les ordonnancements caractérisés par un ordonnancement de groupes, nous pouvons utiliser un calcul symétrique du pire des cas Artigues et al. (2005; 2016). Ce calcul est basé sur une borne inférieure de la date de début au plus tôt $r'_{k,j}$ et une borne inférieure de la date de fin au plus tôt dans le meilleur des cas $C'_{k,j}$ tel que :

$$C'_{k,j} = r'_{k,j} + \rho_{k,j}$$

.

Une première approche triviale pour calculer les $r'_{k,j}$, consiste pour les opérations d'un même groupe à relaxer les contraintes disjonctives en considérant que les ressources sont de capacité illimitée. Pour une opération $O_{k,j}$, la borne inférieure de sa date de début ($r'_{k,j}$), est calculée comme suit :

$$r'_{k,j} = \max(r_{k,j},\ C'_{\pi^k_j,j},\ \underbrace{\max}_{O_{k,i} \in G^k_{g^k_j - 1}}\ C'_{k,i})$$

Dans cette formulation, la date de début d'une opération $O_{k,j}$ dépend simplement soit de sa date de début au plus tôt (date de disponibilité $r_{k,j}$), soit de la date de fin au

plus tôt de son prédécesseur dans le job (dans ce cas, de la borne inférieure : $C'_{\pi^k_j,j}$), soit de la date de fin au plus tôt de toutes les opérations du groupe précédent sur la même machine (dans ce cas, de la borne inférieure : $\underbrace{\max}_{O_{k,i}\in G^k_{g^k_j-1}} C'_{k,i}$).

Une amélioration de cette formule a été proposée par Pinot (2008), cette amélioration repose sur la notion de groupes. Dans ce cas, la borne inférieure du meilleur des cas pour la date de début d'une opération $O_{k,j}$ est donnée comme le maximum de la borne inférieure de la date de fin de tous ses prédécesseurs ; pour une opération $O_{k,j}$, ses prédécesseurs incluent les prédécesseurs de cette opération ($C'_{\pi^k_j,j}$) ainsi que le groupe prédécesseur de cette opération sur la même machine ($G^k_{g^k_j-1}$). En effet, une opération d'un groupe donné ne peut pas être exécutée avant la fin de toutes les opérations du groupe prédécesseur sur la même machine. Par conséquent, une opération ne pourra commencer au plus tôt qu'après le makespan optimal du groupe prédécesseur $G^k_{g^k_j-1}$.

$$\begin{cases} r'_{k,j} = \max(r_{k,j}, C'_{\pi^k_j,j}, \gamma(G^k_{g^k_j-1})) \\ \gamma(G^k_i) = C_{\max}\, de\, 1|r_{k,i}|C_{\max}, \forall O_{k,j} \in G^k_i, r_{k,j} = r'_{k,j} \end{cases} \tag{5.4}$$

$\gamma(G^k_{g^k_j-1})$ est une borne inférieure de la date de fin du groupe prédécesseur de l'opération $O_{k,j}$. Cette borne inférieure est calculée en résolvant un problème de recherche du $C_{\max}$ à une machine avec contrainte de date de début au plus tôt, problème noté $1|r_{k,j}|C_{\max}$ avec $r_{k,j} = r'_{k,j}$. Ce problème d'une machine peut être résolu par un algorithme de complexité polynomiale en ordonnançant les opérations du groupe par ordre croissant de leurs dates de disponibilités (Lawler 1973, Brucker et Knust 2007).

Dans notre exemple de job shop, les prédécesseurs de l'opération $O_6$ qui est exécutée sur la machine $M_1$ comprennent l'opération $O_5$ (exécutée sur $M_3$) à cause de la contrainte de précédence entre les deux opérations, et aussi les deux opérations $O_1$ et $O_7$ du $G^1_1$, qui est le groupe prédécesseur de cette opération sur la même machine. Le tableau 5.1 illustre le calcul des bornes inférieures de tous les groupes et opérations de cet exemple.

TABLE 5.1 – *Bornes inférieures des dates de début/fin au plus tôt*

| $O_i$ | $O_1$ | $O_7$ | $O_4$ | $O_2$ | $O_5$ | $O_8$ | $O_3$ | $O_6$ | $O_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $r'_i$ | 0 | 0 | 0 | 2 | 2 | 4 | 7 | 5 | 6 |
| $C'_i$ | 1 | 4 | 2 | 6 | 5 | 6 | 8 | 6 | 9 |
| $\gamma(G_i)$ | 5 | | 2 | 6 | 7 | | 8 | 6 | 9 |

La figure 5.2 représente une formulation intuitive des $C'_{k,j}$. Nous pouvons remarquer que le makespan de cet ordonnancement ne représente pas le makepan optimal du problème comme représenté dans la figure 4.3(a).

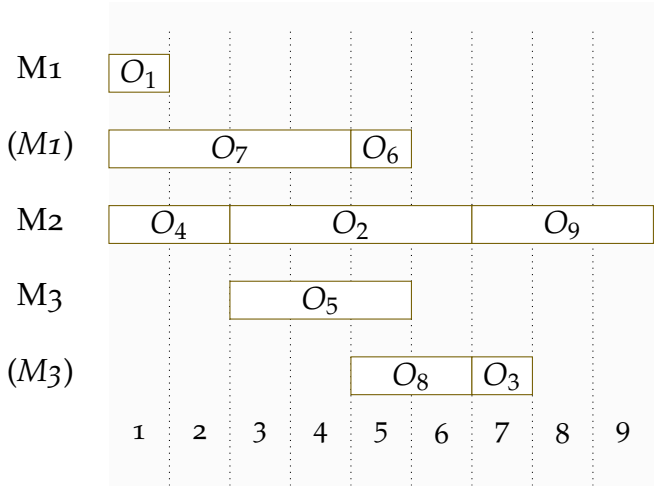



FIGURE 5.2 – *formulation intuitive des* $C'_{k,j}$

### 5.2.1 Amélioration des dates de début/fin au plus tôt des opérations

Dans cette section, nous présentons une amélioration pour le calcul des bornes inférieures des dates de début/fin au plus tôt des opérations en utilisant une technique d'ajustement basée sur la combinaison entre les contraintes de précédence et les contraintes entre les groupes exécutés sur une même machine.

Cette amélioration concerne tout ensemble de deux opérations $O_{k,j}$ et $O_{k,i}$ appartenant à deux groupes successeurs sur une même machine $G^k_{g^k_j}$ et $G^k_{g^k_i}$ ($G^k_{g^k_i-1} = G^k_{g^k_j}$). Si des prédécesseurs (directs ou indirects) de ces deux opérations respectivement $O^-_{k,j}$ et $O^-_{k,i}$ se trouvent dans un même groupe $G^{k'}_l$ alors la date de début de l'opération $O_{k,j}$ dépend de la décision prise dans le groupe $G^{k'}_l$. Cette propriété est illustrée dans les deux figures 5.3 et 5.4.

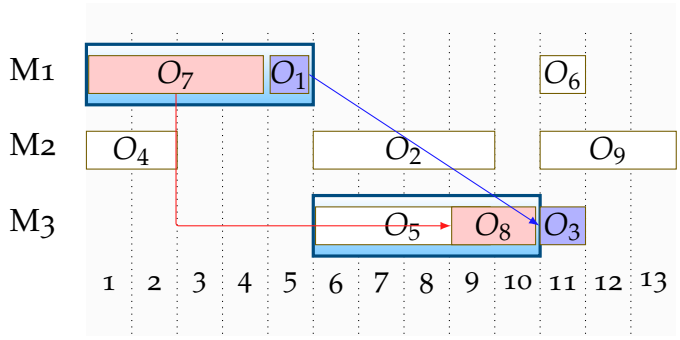



FIGURE 5.3 – *Contraintes de précédence entre* $O_8$ *et* $O_3$

Dans la figure 5.4 par exemple, nous avons sur la machine $M_2$ l'opération $O_9$ qui doit commencer après $O_8$ à cause des contraintes de précédence entre les opérations, elle doit aussi commencer après l'opération $O_2$ à cause des contraintes de précédence entre les groupes sur la même machine. Les deux prédécesseurs indirects de $O_9$ et $O_2$

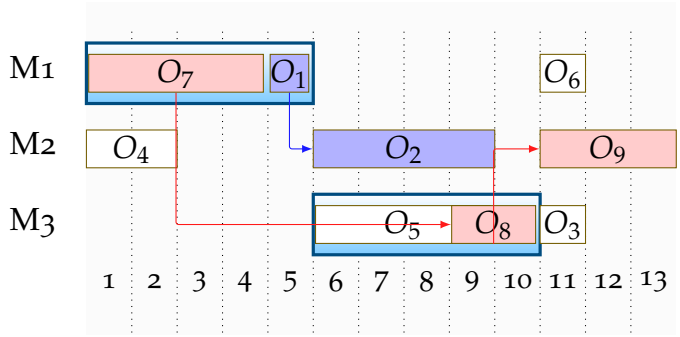



FIGURE 5.4 – *Contraintes de précédence entre $O_2$ et $O_9$*

(qui sont respectivement $O_7$ et $O_1$) se trouvent dans le même groupe, l'ordre d'exécution de ces prédécesseurs influencera donc la date de début de l'opération $O_9$.

Dans le calcul présenté dans le tableau 5.1, le calcul de la borne inférieure de la date de début au plus tôt de $O_9$ en utilisant l'équation 5.4 amène à placer $O_2$ et $O_8$ dans leurs meilleures dates de début au plus tôt, et par transitivité aussi $O_7$ et $O_1$ comme montré dans la figure 5.2, ce qui est impossible. Nous savons que $O_7$ et $O_1$ ne peuvent pas commencer en même temps.

Pour prendre en considération cette situation, nous proposons une technique d'ajustement pour la borne inférieure de la date de début au plus tôt de l'opération $O_9$. Cette technique nécessite d'envisager les deux possibilités de séquencement des deux prédécesseurs ($O_1$ $O_7$) et de calculer à chaque fois la date au plus tôt résultante de l'opération successeur ($O_9$), la plus petite des valeurs étant une borne inférieure valide comme montré dans le tableau 5.2.

TABLE 5.2 – *Amélioration des bornes inférieures des dates de début/fin au plus tôt*

| | $r_1 < r_7$ | | | $r_7 < r_1$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $O_i$ | $r'1_{k,j}$ | $C'1_{k,j}$ | $\gamma 1_{g_{l,k}}$ | $r'2_{k,j}$ | $C'2_{k,j}$ | $\gamma 2_{g_{l,k}}$ | min $(r'1_{k,j}, r'2_{k,j})$ | min $(C'1_{k,j}, C'2_{k,j})$ | min $(\gamma 1_{g_{l,k}}, \gamma 2_{g_{l,k}})$ |
| $O_1$ | 0 | 1 | 5 | 4 | 5 | 5 | 0 | 1 | 5 |
| $O_7$ | 1 | 5 | | 0 | 4 | | 0 | 4 | |
| $O_4$ | 0 | 2 | 2 | 0 | 2 | 2 | 0 | 2 | 2 |
| $O_2$ | 2 | 6 | 6 | 5 | 9 | 9 | 2 | 6 | 6 |
| $O_5$ | 2 | 5 | 7 | 2 | 5 | 7 | 2 | 5 | 7 |
| $O_8$ | 5 | 7 | | 4 | 6 | | 4 | 6 | |
| $O_3$ | 7 | 8 | 8 | 9 | 10 | 10 | 7 | 8 | 8 |
| $O_6$ | 5 | 6 | 6 | 5 | 6 | 6 | 5 | 6 | 6 |
| $O_9$ | 7 | 10 | 10 | 9 | 12 | 12 | 7 | 10 | 10 |

Le tableau 5.2 montre qu'avec cette propriété on obtiendra des bornes inférieures plus précises. La figure 5.5 montre que le makespan obtenu par la formulation intui-

tive des $C'_{k,j}$ donne une performance égale à 10, ce qui représente le makespan optimal du problème.

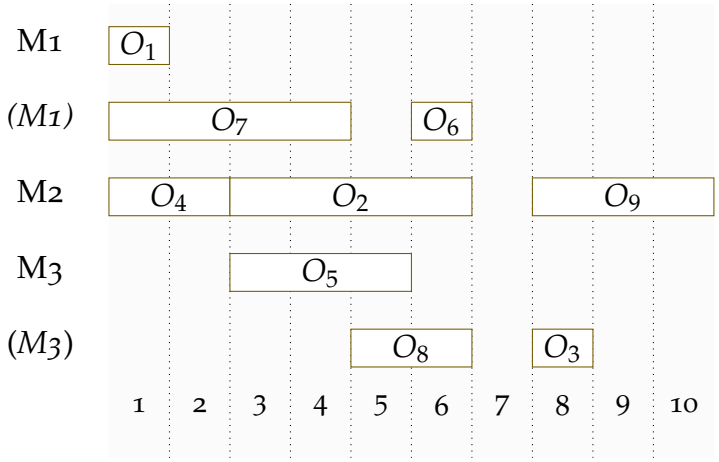



FIGURE 5.5 – *formulation intuitive des* $C'_{k,j}$

Le calcul de $C'_{k,j}$ en utilisant l'équation 5.4 est polynomial (Pinot 2008) et a une complexité de $O(n^2)$ pour les critères réguliers de type *MinMax*. Dans un problème d'ordonnancement de groupes, nous supposons dans la pire situation que pour chaque deux opérations sur deux groupes successifs, la condition illustrée ci-dessus est présente (ce qui est pratiquement impossible). Dans ce cas, cette technique d'ajustement serait appliquée $(n-1) \times m$ fois en théorie ($n > m$). Donc, cette complexité est égale à $O(n^3)$ dans la pire situation.

## 5.3 Borne inférieure pour le meilleur des cas

Une fois que les dates de début/fin au plus tôt sont calculées, la borne inférieure du meilleur des cas (ordonnancement optimal de l'ordonnancement de groupe), est calculée en fonction des dates de début/fin des opérations. Pour le makespan par exemple, nous pouvons calculer cette borne avec l'équation suivante :

$$C^*_{\max} = \max(\gamma(G^k_i)) \tag{5.5}$$

L'amélioration de cette borne repose sur l'amélioration des têtes et des durées de latence des opérations. En utilisant l'équation 5.4, $r'_{k,j}$ représente la tête de l'opération $O_{k,j}$. En raison de la symétrie entre les têtes et les durées de latence, ces dernières peuvent être calculées de la même manière que $r'_{k,j}$ en utilisant une version inversée de l'équation 5.4 (Pinot 2008) :

$$\begin{cases} q'_{k,j} = \max(q''_{\omega^k_j,j}, \gamma'(G^k_{g^k_j+1})) \\ q''_{k,j} = q'_{k,j} + \rho_{k,j} \\ \\ \gamma'(G^k_i) = C_{\max} of 1 | r_{k,j} | C_{\max}, \forall O_{k,j} \in G^k_i, r_{k,j} = q''_{k,j} \end{cases} \tag{5.6}$$

Avec $r'_{k,j}$ étant une tête valide et $q'_{k,j}$ étant une durée de latence valable pour l'opération $O_{k,j}$, une amélioration de la borne inférieure présentée dans l'équation 5.5 pourra être présentée comme suit :

$$C^*_{\max} = \max(\gamma(G^k_i) + \gamma'(G^k_i)) \tag{5.7}$$

Dans un problème d'ordonnancement de groupes, nous pouvons utiliser la relaxation d'une machine présentée dans la section précédente. Mais, nous pouvons aussi utiliser cette relaxation différemment dans le cas où chaque groupe est représenté par une machine. Puis pour toutes les opérations de chaque groupe, nous évaluons les têtes et les durées de latence en utilisant les équations 5.4 et 5.6. Une fois que ces évaluations sont faites, pour chaque groupe, le makespan optimal pour le meilleur des cas est calculé en utilisant l'algorithme exact de séparation et d'évaluation de Carlier (1982). Nous référençons cette borne inférieure calculée à partir de la machine (groupe) $G^k_i$ par $\lambda(G^k_i)$. L'évaluation maximale de toutes ces évaluations représente une borne inférieure du meilleur des cas d'un ordonnancement de groupes et elle est représentée dans l'équation suivante :

$$C^*_{\max} = \max(\lambda(G^k_i)) \tag{5.8}$$

### 5.3.1 Expérimentations et résultats

Afin de tester ces trois bornes inférieures, nous utilisons un ensemble d'instances de job shop très utilisé dans la littérature, nommé La01 à La40, dites instances de Lawrence (Lawrence 1984). Ce sont des instances de job shop classiques, avec $m$ opérations pour chaque job ($m$ le nombre de machines), chaque opération d'un job s'exécutant sur une machine différente. Cet ensemble est composé de 40 instances de tailles différentes (5 instances pour chaque taille). Pour chaque instance, nous générons un ordonnancement de groupes avec une solution optimale connue *a priori* et avec la plus grande flexibilité possible en utilisant l'algorithme *EBJG* ($\epsilon = \infty$).

Les résultats des bornes inférieures pour chaque instance sont exposés dans les trois tableaux (tableau 5.3,5.4 et 5.5) qui représentent respectivement le résultat de la borne inférieure calculée par l'équation 5.5 (noté *LB_best-case1*), l'équation 5.7 (noté *LB_best-case2*) et l'équation 5.8 (noté *LB_best-case3*).

Les quatre premières colonnes de chaque ligne du premier tableau représentent les informations sur l'instance ainsi que sur l'ordonnancement de groupes généré à partir de cette instance. Pour chaque instance, est ensuite donnée la solution optimale qui représente la valeur exacte du meilleur des cas pour le $C_{\max}$. Ensuite, les trois colonnes qui suivent représentent respectivement : la borne inférieure du meilleur des cas calculé, le pourcentage d'erreur par rapport à la valeur exacte et le temps consommé en milliseconde pour calculer cette valeur.

Les bornes présentées dans cette section ainsi que toutes les expérimentations de

ce manuscript ont été codées en JAVA et exécutées sur un Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz.

Sur la base des résultats présentés dans les trois tableaux, la borne inférieure *LB_best-case1* fournit une solution optimale pour 22 instances sur 40. Dans ces 22 instances, on trouve toutes les instances de 20 jobs X 5 machines ainsi que toutes les instances de 30 jobs X 10 machines et presque toutes les instances de 15 jobs X 5 machines sauf La07 (avec une erreur moyenne de 2.36%). L'erreur moyenne de la distance entre la solution optimale et le résultat de la borne inférieure pour les 40 instances est égale à 1.43%.

Les deux bornes inférieures *LB_best-case2* et *LB_best-case3* ont pu trouver la solution optimale d'une seule instance additionnelle (La07), par rapport à *LB_best-case1*. Cependant, l'erreur moyenne de toutes les instances représentées dans l'avant dernière ligne des deux tableaux est presque similaire pour les deux bornes et représente à peine 1% de la valeur optimale du makespan ; *LB_best-case2* présente un léger avantage par rapport à *LB_best-case3* sur l'instance La40 avec une différence d'une seule unité de makespan.

Il est notable que, dans les trois tableaux, les instances les plus difficiles avec des résultats non-optimaux sont les instances carrées de 10 et 15 machines. Le temps de CPU consommé pour *LB_best-case1* est presque nul ce qui rend cette borne la plus rapide malgré son efficacité inférieure par rapport aux deux autres bornes. Ce résultat était attendu puisque cette borne ne calcule pas les durées de latences des groupes d'opérations permutables. Cette rapidité de calcul est aussi remarquée dans *LB_best-case2* pour la plupart des instances qui ont pu être résolues en presque une seconde sauf La31 et La37. Cette borne a consommé en totalité 101 secondes de CPU sur toutes les instances (la moitié de *LB_best-case3*). Cela est dû au temps de calcul consommé par *LB_best-case3* sur chaque machine (groupe) pour calculer la solution optimale du problème d'une machine à l'aide de l'algorithme de séparation et d'évaluation de Carlier (1982).

La principale conclusion tirée de ces résultats est que ces trois bornes sont efficaces et peuvent être utiliser en temps réel dans un système d'aide à la décision. Dans le reste de ce manuscrit, nous avons pris la décision d'utiliser la deuxième borne proposée qui s'avère plus intéressante en terme de temps de calcul et $C_{max}$.

Toutefois, le calcul de la borne inférieure du meilleur des cas pourra être utilisé comme indicateur d'aide à la décision informant le décideur sur la meilleure prochaine opération à exécuter dans un groupe d'opérations permutables. Cette évaluation approchée du meilleur des cas pourra être améliorée par deux façons, soit en implémentant les bornes proposées dans une heuristique de recherche du meilleur des cas, comme proposé dans le travail de Pinot (2008), soit par l'utilisation d'une méthode de séparation et dévaluation afin de trouver la valeur exacte du meilleur des cas. Cependant, notre objectif est d'utiliser le meilleur des cas en temps réel dans un système d'aide à la décision où une évaluation rapide est nécessaire et primordiale. Dans la section suivante, nous montrons que l'évaluation exacte de cet indicateur

TABLE 5.3 – *Borne inférieure du meilleur des cas en utilisant l'équation 5.5*

| n X m | Instances | *Nbre de groupes* | *Nbre de décisions* | *Solution Optimale* | *Borne inférieure (LB_best-case1)* | *Erreur gap (%)* | temps (ms) |
|---|---|---|---|---|---|---|---|
| 10 X 5 | La01 | 15 | 35 | 666 | 666 | 0% | 0 |
| | La02 | 18 | 32 | 655 | 655 | 0% | 0 |
| | La03 | 15 | 35 | 597 | 588 | 1.51% | 0 |
| | La04 | 14 | 36 | 590 | 588 | 0.34% | 0 |
| | La05 | 15 | 35 | 593 | 593 | 0% | 0 |
| 15 X 5 | La06 | 19 | 56 | 926 | 926 | 0% | 0 |
| | La07 | 17 | 58 | 890 | 869 | 2.36% | 0 |
| | La08 | 16 | 59 | 863 | 863 | 0% | 0 |
| | La09 | 16 | 59 | 951 | 951 | 0% | 0 |
| | La10 | 19 | 56 | 958 | 958 | 0% | 0 |
| 20 X 5 | La11 | 20 | 80 | 1222 | 1222 | 0% | 10 |
| | La12 | 21 | 79 | 1039 | 1039 | 0% | 0 |
| | La13 | 18 | 82 | 1150 | 1150 | 0% | 0 |
| | La14 | 19 | 81 | 1292 | 1292 | 0% | 0 |
| | La15 | 22 | 78 | 1207 | 1207 | 0% | 0 |
| 10 X 10 | La16 | 43 | 57 | 945 | 924 | 2.22% | 0 |
| | La17 | 36 | 64 | 784 | 751 | 4.21% | 0 |
| | La18 | 40 | 60 | 848 | 787 | 7.19% | 0 |
| | La19 | 39 | 61 | 842 | 796 | 5.46% | 0 |
| | La20 | 39 | 61 | 902 | 806 | 10.64% | 0 |
| 15 X 10 | La21 | 54 | 96 | 1046 | 1019 | 2.58% | 0 |
| | La22 | 49 | 101 | 927 | 927 | 0% | 0 |
| | La23 | 54 | 96 | 1032 | 1032 | 0% | 0 |
| | La24 | 50 | 100 | 935 | 914 | 2.25% | 0 |
| | La25 | 50 | 100 | 977 | 954 | 2.35% | 0 |
| 20 X 10 | La26 | 53 | 147 | 1218 | 1218 | 0% | 0 |
| | La27 | 56 | 144 | 1252 | 1231 | 1.68 | 0 |
| | La28 | 63 | 137 | 1273 | 1273 | 0% | 0 |
| | La29 | 60 | 140 | 1202 | 1189 | 1.08% | 0 |
| | La30 | 58 | 142 | 1355 | 1355 | 0% | 0 |
| 30 X 10 | La31 | 69 | 231 | 1784 | 1784 | 0% | 0 |
| | La32 | 64 | 236 | 1850 | 1850 | 0 % | 0 |
| | La33 | 64 | 236 | 1719 | 1719 | 0 % | 0 |
| | La34 | 72 | 228 | 1721 | 1721 | 0 % | 0 |
| | La35 | 73 | 227 | 1888 | 1888 | 0 % | 0 |
| 15 X 15 | La36 | 88 | 137 | 1268 | 1240 | 2.21% | 0 |
| | La37 | 85 | 140 | 1397 | 1348 | 3.51% | 0 |
| | La38 | 88 | 137 | 1196 | 1131 | 5.43% | 0 |
| | La39 | 86 | 139 | 1233 | 1221 | 0.98% | 0 |
| | La40 | 86 | 139 | 1222 | 1205 | 1.39% | 0 |
| Moyenne | | | | | | 1.43% | |
| Somme | | | | | | | 10 |

TABLE 5.4 – *Borne inférieure du meilleur des cas en utilisant l'équation 5.7*

| Instances | *Solution Optimale* | *Borne inférieure (LB_best-case2)* | *Erreur gap(%)* | temps (ms) |
|---|---|---|---|---|
| La01 | 666 | 666 | 0% | 0 |
| La02 | 655 | 655 | 0% | 0 |
| La03 | 597 | 588 | 1.51% | 0 |
| La04 | 590 | 588 | 0.34% | 0 |
| La05 | 593 | 593 | 0% | 0 |
| La06 | 926 | 926 | 0% | 0 |
| La07 | 890 | 890 | 0% | 0 |
| La08 | 863 | 863 | 0% | 0 |
| La09 | 951 | 951 | 0% | 0 |
| La10 | 958 | 958 | 0% | 0 |
| La11 | 1222 | 1222 | 0% | 10 |
| La12 | 1039 | 1039 | 0% | 0 |
| La13 | 1150 | 1150 | 0% | 0 |
| La14 | 1292 | 1292 | 0% | 0 |
| La15 | 1207 | 1207 | 0% | 0 |
| La16 | 945 | 931 | 1.48% | 0 |
| La17 | 784 | 761 | 2.93% | 0 |
| La18 | 848 | 816 | 3.77% | 0 |
| La19 | 842 | 796 | 5.46% | 0 |
| La20 | 902 | 850 | 5.76% | 0 |
| La21 | 1046 | 1045 | 0.10% | 0 |
| La22 | 927 | 927 | 0% | 2 |
| La23 | 1032 | 1032 | 0% | 20 |
| La24 | 935 | 914 | 2.25% | 0 |
| La25 | 977 | 954 | 2.35% | 2 |
| La26 | 1218 | 1218 | 0% | 0 |
| La27 | 1252 | 1235 | 1.36% | 877 |
| La28 | 1273 | 1273 | 0% | 0 |
| La29 | 1202 | 1189 | 1.08% | 22 |
| La30 | 1355 | 1355 | 0% | 10 |
| La31 | 1784 | 1784 | 0% | 79384 |
| La32 | 1850 | 1850 | 0 % | 10 |
| La33 | 1719 | 1719 | 0 % | 10 |
| La34 | 1721 | 1721 | 0 % | 6 |
| La35 | 1888 | 1888 | 0 % | 10 |
| La36 | 1268 | 1244 | 1.89% | 14 |
| La37 | 1397 | 1392 | 0.36% | 19283 |
| La38 | 1196 | 1147 | 4.10% | 1 |
| La39 | 1233 | 1230 | 0.24% | 1086 |
| La40 | 1222 | 1208 | 1.15% | 32 |
| Moyenne | | | 0.903% | |
| Somme | | | | 100779 |

TABLE 5.5 – *Borne inférieure du meilleur des cas en utilisant l'équation 5.8*

| Instances | *Solution Optimale* | *Borne inférieure (LB_best-case3)* | *Erreur gap (%)* | temps (ms) |
|---|---|---|---|---|
| La01 | 666 | 666 | 0% | 40 |
| La02 | 655 | 655 | 0% | 0 |
| La03 | 597 | 588 | 1.51% | 0 |
| La04 | 590 | 588 | 0.34% | 8 |
| La05 | 593 | 593 | 0% | 0 |
| La06 | 926 | 926 | 0% | 10 |
| La07 | 890 | 890 | 0% | 0 |
| La08 | 863 | 863 | 0% | 20 |
| La09 | 951 | 951 | 0% | 20 |
| La10 | 958 | 958 | 0% | 10 |
| La11 | 1222 | 1222 | 0% | 60 |
| La12 | 1039 | 1039 | 0% | 21 |
| La13 | 1150 | 1150 | 0% | 4886 |
| La14 | 1292 | 1292 | 0% | 60 |
| La15 | 1207 | 1207 | 0% | 91 |
| La16 | 945 | 931 | 1.48% | 0 |
| La17 | 784 | 761 | 2.93% | 0 |
| La18 | 848 | 816 | 3.77% | 0 |
| La19 | 842 | 796 | 5.46% | 10 |
| La20 | 902 | 850 | 5.76% | 0 |
| La21 | 1046 | 1045 | 0.10% | 0 |
| La22 | 927 | 927 | 0% | 10 |
| La23 | 1032 | 1032 | 0% | 30 |
| La24 | 935 | 914 | 2.25% | 10 |
| La25 | 977 | 954 | 2.35% | 10 |
| La26 | 1218 | 1218 | 0% | 0 |
| La27 | 1252 | 1235 | 1.36% | 1681 |
| La28 | 1273 | 1273 | 0% | 10 |
| La29 | 1202 | 1189 | 1.08% | 40 |
| La30 | 1355 | 1355 | 0% | 21 |
| La31 | 1784 | 1784 | 0% | 179457 |
| La32 | 1850 | 1850 | 0 % | 161 |
| La33 | 1719 | 1719 | 0 % | 180 |
| La34 | 1721 | 1721 | 0 % | 56 |
| La35 | 1888 | 1888 | 0 % | 70 |
| La36 | 1268 | 1244 | 1.89% | 15 |
| La37 | 1397 | 1392 | 0.36% | 19346 |
| La38 | 1196 | 1147 | 4.10% | 8 |
| La39 | 1233 | 1230 | 0.24% | 1084 |
| La40 | 1222 | 1207 | 1.23% | 35 |
| Moyenne | | | 0.905% | |
| Somme | | | | 207460 |

pourra être calculée en un temps acceptable d'autant plus qu'en pratique le nombre de décisions à prendre dans un ordonnancement de groupes n'est pas aussi élevé que celui indiqué dans le tableau 5.3.

## 5.4 Méthode de séparation et d'évaluation pour le meilleur des cas

L'efficacité et la puissance d'une méthode de séparation et d'évaluation dépend des bornes utilisées dans la phase d'évaluation ainsi que de la méthode de parcours des nœuds. Le meilleur des cas d'un ordonnancement de groupes se trouve dans la classe des ordonnancements actifs. Dans un tel ordonnancement, il est impossible d'avancer une opération sans en retarder une autre. Afin d'éviter l'énumération des ordonnancements non actifs, la méthode de séparation (génération des nœuds) est présentée dans les étapes suivantes :

- Génération d'une liste $L(G)$ contenant tous les groupes de l'ordonnancement de groupes avec au moins deux opérations permutables.
- Les dates de début au plus tôt $r_{k,j} = r'_{k,j}$ de chaque opération du premier groupe dans la liste sont calculées.
- La plus petite date de fin de ces opérations est calculée $C_{\min} = \underbrace{\min}_{\forall O_{k,j} \in G_i^k} (r_{k,j} + \rho_{k,j})$.
- Toutes les opérations qui ont une date de début inférieure à $C_{\min}$ $(r_{k,j} < C_{\min})$ sont sélectionnées.
- Pour chacune de ces opérations, on génère un nouveau nœud.
- On supprime le groupe traité et on refait la même procédure pour le groupe suivant jusqu'à ce que la liste soit vide.

Cette procédure de séparation est appliquée à chaque nœud de l'arbre. Les nœuds de cet arbre sont les opérations définies dans la liste $L(G)$. L'ordre de séparation des nœuds dépend de l'ordre des groupes dans la liste $L(G)$. Dans le travail de Pinot (2008), les groupes de cette liste sont ordonnés suivant un ordre de précédence, cela veut dire que si le groupe $G_i^k$ contient une opération qui est prédécesseur d'une opération de $G_{i'}^{k'}$, alors le groupe $G_i^k$ est traité avant le groupe $G_{i'}^{k'}$.

Appliquons cette méthode à notre exemple de job shop représenté par l'ordonnancement de groupes décrit dans la figure 4.1(b) :

- La liste générée contient les deux groupes suivants $L(G) = \{(O_1, O_7), (O_5, O_8)\}$.
- On commence par le traitement des opérations du premier groupe : $r_1 = 0$, $C_1 = 1$, $r_7 = 0$, $C_7 = 4$, $C_{\min} = \min(1, 4) = 1$.
- Nous avons $r_1 \leq r_7 < C_{\min}$, donc les deux opérations sont sélectionnées et deux sous problèmes sont générés : le premier lorsque $O_1$ est ordonné avant $O_7$ et le deuxième pour l'inverse.
- La nouvelle liste devient $L(G) = \{(O_5, O_8)\}$.

— Pour le premier sous problème lorsque $O_1$ est placé avant $O_7$, $r_5 = 2$, $C_5 = 5$, $r_8 = 5$, $C_8 = 7$, $C_{\min} = 5$.
— Dans ce cas, seulement l'opération $O_5$ est sélectionnée car $r_8 = 5 = C_{\min}$, alors $O_5$ est ordonnée avant $O_8$ est la liste $L(G)$ devient vide.
— Pour le deuxième sous problème lorsque $O_7$ est placé avant $O_1$, $r_5 = 2$, $C_5 = 5$, $r_8 = 4$, $C_8 = 6$, $C_{\min} = 5$.
— Nous avons $r_5 < r_8 < C_{\min}$, donc les deux opérations sont sélectionnées et deux sous problèmes sont générés : le premier lorsque $O_5$ est ordonnée avant $O_7$ et le deuxième lorsque $O_8$ est placée avant $O_5$.
— La procédure de séparation est terminée car la liste $L(G)$ est vide pour tous les sous- problèmes. L'arbre des ordonnancements actifs est généré comme indiqué dans la figure 5.6.

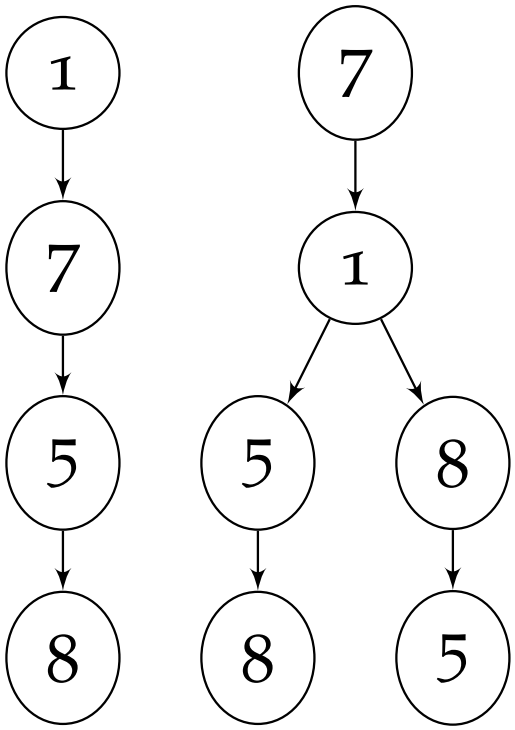



FIGURE 5.6 – *Procédure de séparation sur les ordonnancements actifs*

## 5.5 Amélioration de la procédure de séparation pour le meilleur des cas

Comme souligné précédemment, l'efficacité d'une procédure de séparation et d'évaluation dépend de l'ordre de parcours des nœuds. La procédure de séparation génère des nœuds, mais l'ordre selon lequel les nœuds sont explorés affecte les performances de l'algorithme : si la meilleure solution est trouvée plus tôt, la borne supérieure sera meilleure et plus de nœuds seront éliminés.

Dans l'exemple illustré précédemment, les groupes sont ordonnés selon la relation de précédence entre eux, si aucune relation de précédence n'est trouvée entre deux

groupes, priorité est donnée au groupe qui a la plus petite date de début, dans la suite cette approche est notée "*PredOrder*". Dans cet exemple, le groupe $G_1^1$ contenant $O_1$ et $O_7$ est traité avant le groupe $G_1^3$ contenant $O_5$ et $O_8$ à cause de la relation de précédence entre $O_7$ et $O_8$.

Afin de réduire l'espace de recherche dans la procédure de séparation et d'évaluation, nous proposons deux méthodes de séparations basées sur l'ordre de traitement des groupes.

Nous appelons les groupes de la liste $L(G)$ des voisins (Neighbors). Dans ces deux nouvelles approches, la priorité est donnée au groupe ayant le moins de relations avec les autres voisins. Le nombre de relations d'un groupe donné est calculé selon les relations de précédence (directe et indirecte) ou de succession (directe et indirecte) des opérations de ce groupe avec les opérations de ses voisins. Ces approches sont notées "*NeighborDirectRel*" et "*NeighborIndirectRel*". Dans l'approche *NeighborDirectRel*, une relation de voisinage existe entre deux groupes si seulement si une opération d'un de ces deux groupes est prédécesseur direct ($O_{\pi_j^k,j}$) d'une opération $O_{k,j}$ du deuxième groupe. Cependant, pour l'approche *NeighborIndirectRel*, cette relation de voisinage est présente lorsqu'une opération de ces groupes est prédécesseur direct ou indirect ($\in O_{k,j}^-$) d'une opération $O_{k,j}$ du deuxième groupe.

Afin d'illustrer ces deux procédures de séparation, nous définissons le problème représenté dans le tableau 5.6 et la figure 5.7.

TABLE 5.6 – *Exemple d'un flow shop*

| $J_j$ | $J_1$ | | | $J_2$ | | | $J_3$ | | | $J_4$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $O_i$ | $O_1$ | $O_2$ | $O_3$ | $O_4$ | $O_5$ | $O_6$ | $O_7$ | $O_8$ | $O_9$ | $O_{10}$ | $O_{11}$ | $O_{12}$ |
| $M_k$ | $M_1$ | $M_3$ | $M_2$ | $M_1$ | $M_3$ | $M_2$ | $M_1$ | $M_3$ | $M_2$ | $M_1$ | $M_3$ | $M_2$ |
| $\rho_{k,j}$ | 3 | 4 | 2 | 2 | 3 | 4 | 3 | 5 | 5 | 2 | 2 | 2 |

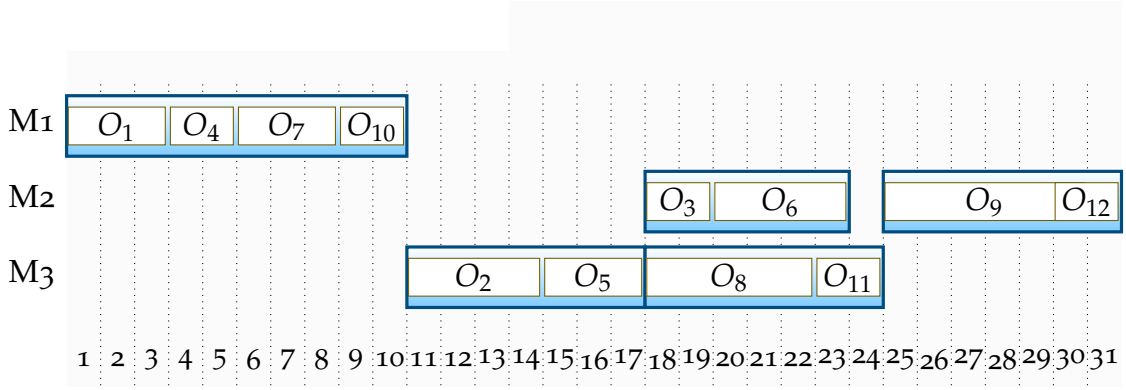



FIGURE 5.7 – *Ordonnancement de groupes*

La figure 5.7 représente un ordonnancement de groupes admissible pour le flow shop présenté dans le tableau 5.6. Cet ordonnancement de groupes est composé de cinq groupes d'opérations permutables : $G_1^1 = (O_1, O_4, O_7, O_{10})$, $G_1^3 = (O_2, O_5)$, $G_1^2 =$

$(O_3, O_6)$, $G_2^3 = (O_8, O_{11})$, $G_2^2 = (O_9, O_{12})$. L'ordre des groupes pour la procédure de séparation *NeighborDirectRel* est défini comme suit :

1. Générer $L(G)$ avec un ordre aléatoire :
   $L(G) = \{G_1^1, G_1^3, G_1^2, G_2^3, G_2^2\}$
2. Pour chaque groupe de la liste $L(G)$, les groupes voisins, dénotés par la variable $neighbors(G_i^k)$, sont générés sans répétition :

   — $neighbors(G_1^1) = \{G_1^3, G_2^3\}$ (parce que $O_2$,$O_5$ sont des successeurs de $O_1$,$O_4$ et $O_8$,$O_{11}$ sont des successeurs de $O_7$,$O_{10}$ )

   — $neighbors(G_1^3) = \{G_1^1, G_1^2\}$ (parce que $O_2$,$O_5$ sont des successeurs de $O_1$,$O_4$ et les prédécesseurs de $O_3$,$O_6$ )

   — $neighbors(G_1^2) = \{G_1^3\}$.

   — $neighbors(G_2^3) = \{G_1^1, G_2^2\}$.

   — $neighbors(G_2^2) = \{G_2^3\}$.
3. Choisir le groupe avec le moins de voisins ; Card($neighbors(G_1^2)$) = Card($neighbors(G_2^2)$) = 1, il s'agit de la valeur minimale, dans ce cas $G_1^2$ est choisi en premier à cause des contraintes de précédences entre $O_2$,$O_3$ et $O_5$,$O_6$.
4. Supprimer $G_1^2$ de la liste $L(G)$ :
   $L(G) = \{G_1^1, G_1^3, G_2^3, G_2^2\}$
5. Répéter le processus 2 à $G_1^1$,$G_1^3$,$G_2^3$ et $G_2^2$ :
   $neighbors(G_1) = \{G_1^3, G_2^3\}$
   $neighbors(G_1^3) = \{G_1^1\}$
   $neighbors(G_2^3) = \{G_1^1, G_2^2\}$
   $neighbors(G_2^2) = \{G_2^3\}$.
6. $G_1^3$ est choisi parce sa date de début est la plus petite.
7. Supprimer $G_1^3$ de $L(G)$ :
   $L(G) = \{G_1^1, G_2^3, G_2^2\}$
8. Répéter le processus 2 à $G_1^1$,$G_2^3$ et $G_2^2$ :
   $neighbors(G_1^1) = \{G_2^3\}$
   $neighbors(G_2^3) = \{G_1^1, G_2^2\}$
   $neighbors(G_2^2) = \{G_2^3\}$
9. $G_1^1$ est choisi à cause des contraintes de précédence indirectes entre $O_{10}$ et $O_{12}$ ($O_{10}$ avant $O_{11}$ et $O_{11}$ avant $O_{12}$ alors $O_{10}$ avant $O_{12}$ ).
10. Supprimer $G_1^1$ de $L(G)$ :
    $L(G) = \{G_2^3, G_2^2\}$

11. Répéter le processus 2 à $G_4$ and $G_5$ :
    $neighbors(G_2^3) = \{G_2^2\}$ .
    $neighbors(G_2^2) = \{G_2^3\}$ .
12. $G_2^3$ est choisi avant $G_2^2$ à cause des contraintes de précédence entre $O_8$,$O_9$ et $O_{11}$,$O_{12}$.
13. L'ordre de traitement des groupes est le suivant : $G_1^2$, $G_1^3$, $G_1^1$, $G_2^3$ then $G_2^2$.

Dans la méthode *NeighborIndirectRel*, l'espace de recherche des voisins est élargi en regardant non seulement les premiers successeurs et prédécesseurs directs ($O_{\pi_j^k,j}$) de l'opération en cours, mais toutes les opérations successeurs et prédécesseurs du même job ($O_{k,j}^-$). Par exemple, avec la procédure de séparation *NeighborDirectRel*, $G_2^2$ a seulement le voisin $G_2^3$ à cause des contraintes de précédence directes entre $O_9$, $O_{12}$ et $O_8$,$O_{11}$ respectivement, alors que dans la procédure de séparation *NeighborIndirectRel*, les voisins de $G_2^2$ sont $G_2^3$ et aussi $G_1^1$ parce que $O_7$ et $O_{10}$ sont dans le même job que $O_9$ et $O_{12}$ respectivement (prédécesseurs indirects).

Pour résumer, l'ordre de traitement des groupes dans l'algorithme de séparation et d'évaluation est :

— *PredOrder* : $\{G_1^1, G_1^3, G_1^2, G_2^3, G_2^2\}$
— *NeighborDirectRel* : $\{G_1^2, G_1^3, G_1^1, G_2^3, G_2^2\}$
— *NeighborIndirectRel* : $\{G_1^3, G_1^2, G_1^1, G_2^3, G_2^2\}$

Nous proposons une autre méthode de séparation nommée "*CriticalPath*". Cette méthode est basée sur les groupes appartenant aux chemins critiques possibles du graphe disjonctif représentant l'ordonnancement de groupes. Pour chaque groupe est calculé la date de début/fin au plus tôt et la date de début/fin au plus tard. Ces dates sont calculé à l'aide de $r'_{k,j}$ et $\gamma(G_i^k)$ représentés dans l'équation 5.4. La nouvelle liste $L(G)$ ne contient que les groupes critiques avec plus d'une opération ; un groupe est critique si sa date de début au plus tôt est égale à sa date de début au plus tard.

La figure 5.8 montre le calcul de ces dates. L'ensemble des deux valeurs en italique (resp. en gras) représente la date de début au plus tôt (resp. tard) et la date de fin au plus tôt (resp. tard) de chaque groupe. Nous constatons à partir de ces valeurs que le groupe $G_1^2$ n'appartient pas au chemin critique de la meilleure solution du nœud courant, car sa date de début au plus tôt est différente de sa date de début au plus tard ; donc le groupe est supprimé de la liste $L(G)$ et remis dans les prochaines listes des sous-nœuds générés. Une fois que les groupes non-critiques sont supprimés, les groupes restants sont ordonnancés selon la procédure *NeighborIndirectRel*.

### 5.5.1 Expérimentations et résultats

Dans cette section, nous présentons l'implémentation des quatre méthodes de séparation *PredOrder, NeighborDirectRe, NeighborIndirectRel, CriticalPath* dans une procédure de séparation et d'évaluation. Les quatre méthodes de séparation sont testées sur les 40 instances de Lawrence (déjà citées) et le critère à minimiser est le makespan.

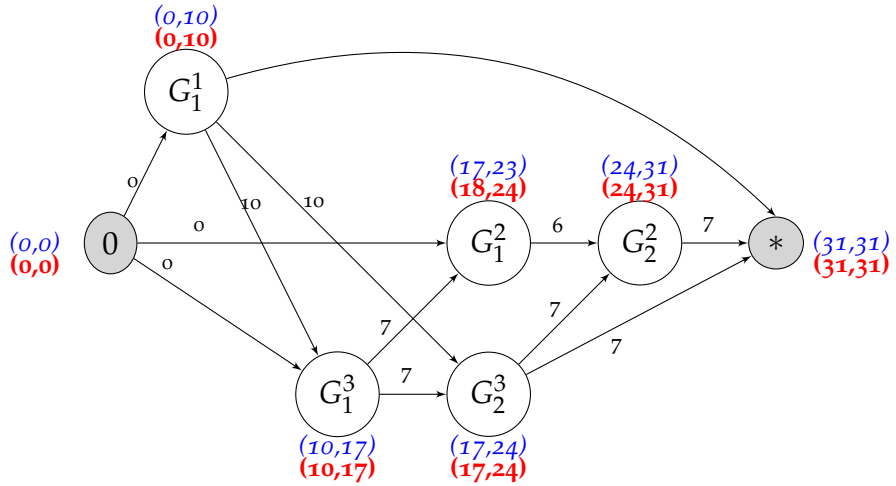



FIGURE 5.8 – *Calcul des groupes critiques*

L'objectif de ces expérimentations est de comparer l'efficacité des quatre méthodes de séparations proposées.

Pour construire les ordonnancements de groupes sur les instances de Lawrence, nous utilisons toujours l'algorithme EBJG pour le regroupement des opérations. Les solutions initiales de référence (avant regroupement) des instances sont des solutions connues *a priori* et sont des solutions optimales pour le job shop. Dans ce cas, le meilleur des cas de l'ordonnancement de groupes construit à partir de ces solutions est une solution initiale (optimale) utilisée. Le paramètre d'arrêt de l'algorithme EBJG utilisé pour le regroupement ($\epsilon$) est égale à 1.10 du makespan initial (avant regroupement). Cette valeur pourra être obtenue de l'équation de Esswein (2003) (avec $\Delta = 10\%$ dans notre cas) :

$$\epsilon = \underbrace{C_{\max}}_{ordo.\,de\,ref.} X\,(1+\Delta) \tag{5.9}$$

Par exemple, pour la sixième instance La06, nous avons construit une solution initiale qui est la solution optimale de cette instance obtenue à partir de Brucker et Jurisch (1993), Pinot (2008), Morikawa et Takahashi (2009) avec un makespan égal à 926. L'algorithme de regroupement fusionne les opérations et les groupes successifs sur une même machine jusqu'à ce que le makespan, après regroupement, dépasse la valeur 1019. Le makespan du meilleur des cas doit être égal au makespan de la solution optimale 926.

Pour chacune des quatre procédures de séparation, si une opération d'entrée est trouvée par la condition 5.3, cette opération est sélectionnée comme premier nœud de l'arbre/sous-arbre courant. Après la génération des nœuds, la technique de recherche en profondeur est utilisée. Dans ce mode, le traitement des nœuds générés se fait selon un ordre croissant des bornes inférieures. Traiter l'arbre en profondeur d'abord permet d'obtenir rapidement une solution mais avec le risque que les explorations n'aboutissent pas à de bonnes solutions. On peut choisir de traiter l'arbre en largeur

et s'intéresser d'abord aux nœuds possédant la meilleure borne inférieure. On traite ainsi moins de nœuds, qu'il faut cependant conserver, ce qui prend beaucoup de mémoire et nous impose de limiter le nombre de nœuds stockés.

La borne inférieure de l'algorithme de séparation et d'évaluation est calculée avec l'équation 5.7. La borne supérieure est initialisée à l'infini. Si dans un nœud de l'arbre, la borne inférieure est supérieure ou égale à la borne supérieure, ce nœud est supprimé de l'exploration ainsi que tous les nœuds qui découlent de ce dernier.

Les résultats des expérimentations sont illustrés dans les tableaux suivants, les variables des colonnes sont définies comme suit :

— N1 : Nombre total de groupes générés à partir de l'instance donnée.
— N2 : Nombre de décisions de l'ordonnancement de groupes.
— N3 : Meilleur des cas (solution optimale).
— N4 : Nombre de nœuds parcourus pour trouver le meilleur des cas.
— N5 : Nombre total de nœuds parcourus pour terminer l'algorithme (N4 + Le nombre de nœuds parcourus pour prouver l'optimalité du meilleur des cas).
— N6 : Temps total en milliseconde pour terminer l'algorithme.

Les quatre méthodes trouvent le meilleur des cas dans un temps très court ; en moins d'une seconde dans 82.5% des instances. La31 représente l'instance la plus gourmande en CPU avec un temps de plus de neuf secondes pour *CriticalPath*. En comparant le CPU total des quatre méthodes, *NeighborIndirectRel* est la méthode de séparation qui a consommé le moins de temps (24 secondes) et *PredOrder* celle qui a consommé le plus de temps (31 secondes).

Pour toutes les instances, au moins une des trois méthodes proposées (*NeighborDirectRel*, *NeighborIndirectRel* et *CriticalPath*) domine les résultats pour N4 et N6. Le nombre de nœuds parcourus pour trouver le meilleur des cas (N4) est très faible dans les quatre méthodes sur la plupart des instances à cause de la borne inférieure qui est très précise (sur la plupart des instances). Ainsi, la plus grande valeur de N4 est 106 sur La32 qui a 158 groupes et 142 décisions à prendre. Cette valeur (N4) représente généralement la profondeur de l'arbre de recherche (la longueur entre la racine et la feuille) puisque la technique de recherche en profondeur est utilisée et les nœuds de gauche sont traités en premier.

L'écart entre N4 et N6 est généralement nul, ce qui signifie que lorsque les algorithmes trouvent le meilleur des cas, pratiquement tous les nœuds restants sont éliminés à cause de leurs bornes inférieures qui sont généralement supérieures ou égales à la valeur du meilleur des cas (Upper Bound).

La méthode *CriticalPath* est plus performante sauf pour les instances La35 et La39. Dans cette dernière instance, l'algorithme a parcouru presque le double de nœuds parcourus par *NeighborDirectRel* et 14 fois celui de *NeighborIndirectRel*, ce qui explique la différence entre la somme des nœuds parcourus pour ces trois algorithmes (représenté dans la dernière ligne des tableaux 5.8, 5.9 et 5.10).

Par rapport à la méthode *PredOrder*, les méthodes *NeighborIndirectRel* et *NeighborDirectRel* ont eu de meilleurs résultats sur trois instances et ont dégradé les résultats

| | N1 | N2 | N3 | N4 | N5 | N6 |
|---|---|---|---|---|---|---|
| La01 | 33 | 17 | 650 | 15 | 15 | 13 |
| La02 | 40 | 40 | 655 | 9 | 9 | 5 |
| La03 | 35 | 15 | 597 | 13 | 13 | 20 |
| La04 | 35 | 15 | 590 | 13 | 13 | 20 |
| La05 | 29 | 21 | 593 | 17 | 17 | 21 |
| La06 | 39 | 36 | 926 | 26 | 26 | 56 |
| La07 | 45 | 30 | 890 | 24 | 24 | 56 |
| La08 | 43 | 32 | 863 | 25 | 25 | 109 |
| La09 | 41 | 34 | 951 | 26 | 26 | 90 |
| La10 | 37 | 38 | 958 | 28 | 28 | 166 |
| La11 | 41 | 59 | 1222 | 34 | 34 | 414 |
| La12 | 47 | 53 | 1039 | 35 | 35 | 193 |
| La13 | 47 | 53 | 1150 | 32 | 32 | 493 |
| La14 | 36 | 64 | 1292 | 30 | 30 | 458 |
| La15 | 51 | 49 | 1207 | 35 | 35 | 131 |
| La16 | 80 | 20 | 945 | 20 | 20 | 33 |
| La17 | 80 | 20 | 784 | 18 | 273 | 579 |
| La18 | 81 | 19 | 848 | 19 | 19 | 40 |
| La19 | 85 | 15 | 842 | 15 | 15 | 27 |
| La20 | 85 | 15 | 902 | 30 | 30 | 40 |
| La21 | 117 | 33 | 1046 | 31 | 31 | 88 |
| La22 | 118 | 32 | 927 | 28 | 28 | 83 |
| La23 | 120 | 30 | 1032 | 29 | 29 | 73 |
| La24 | 120 | 30 | 935 | 27 | 27 | 85 |
| La25 | 118 | 32 | 977 | 31 | 31 | 141 |
| La26 | 142 | 58 | 1218 | 45 | 45 | 314 |
| La27 | 149 | 51 | 1252 | 47 | 47 | 598 |
| La28 | 141 | 59 | 1273 | 54 | 54 | 248 |
| La29 | 146 | 54 | 1202 | 52 | 60 | 1100 |
| La30 | 147 | 53 | 1355 | 44 | 44 | 345 |
| La31 | 165 | 135 | 1784 | 97 | 97 | 9942 |
| La32 | 158 | 142 | 1850 | 106 | 106 | 2845 |
| La33 | 174 | 126 | 1719 | 97 | 97 | 2171 |
| La34 | 177 | 123 | 1721 | 97 | 97 | 4048 |
| La35 | 221 | 79 | 1888 | 68 | 68 | 1022 |
| La36 | 186 | 39 | 1268 | 38 | 949 | 4772 |
| La37 | 187 | 38 | 1397 | 35 | 35 | 994 |
| La38 | 189 | 36 | 1196 | 34 | 34 | 187 |
| La39 | 191 | 34 | 1233 | 34 | 183 | 773 |
| La40 | 194 | 31 | 1222 | 31 | 31 | 158 |
| Somme | | | | | 2812 | 32951 |

TABLE 5.7 – *PredOrder*

| | N4 | N5 | N6 |
|---|---|---|---|
| La01 | 15 | 15 | 19 |
| La02 | 9 | 9 | 3 |
| La03 | 13 | 13 | 16 |
| La04 | 13 | 13 | 21 |
| La05 | 17 | 17 | 23 |
| La06 | 26 | 26 | 76 |
| La07 | 24 | 24 | 59 |
| La08 | 25 | 25 | 82 |
| La09 | 26 | 26 | 86 |
| La10 | 28 | 28 | 191 |
| La11 | 34 | 34 | 379 |
| La12 | 35 | 35 | 179 |
| La13 | 32 | 32 | 312 |
| La14 | 30 | 30 | 378 |
| La15 | 35 | 35 | 128 |
| La16 | 20 | 20 | 32 |
| La17 | 18 | 371 | 512 |
| La18 | 19 | 19 | 35 |
| La19 | 15 | 15 | 28 |
| La20 | 15 | 15 | 24 |
| La21 | 31 | 31 | 78 |
| La22 | 28 | 28 | 86 |
| La23 | 29 | 29 | 76 |
| La24 | 27 | 27 | 67 |
| La25 | 31 | 31 | 115 |
| La26 | 45 | 45 | 292 |
| La27 | 47 | 47 | 198 |
| La28 | 54 | 54 | 245 |
| La29 | 52 | 52 | 458 |
| La30 | 44 | 44 | 184 |
| La31 | 97 | 97 | 7074 |
| La32 | 106 | 106 | 2749 |
| La33 | 97 | 97 | 2107 |
| La34 | 97 | 97 | 4093 |
| La35 | 68 | 68 | 742 |
| La36 | 38 | 49 | 2051 |
| La37 | 35 | 35 | 1029 |
| La38 | 34 | 34 | 136 |
| La39 | 34 | 328 | 1200 |
| La40 | 31 | 31 | 90 |
| Somme | | 2132 | 25653 |

TABLE 5.8 – *NeighborDirectRel*

| | N4 | N5 | N6 |
|---|---|---|---|
| La01 | 15 | 15 | 18 |
| La02 | 9 | 9 | 6 |
| La03 | 13 | 13 | 21 |
| La04 | 13 | 13 | 20 |
| La05 | 17 | 17 | 22 |
| La06 | 26 | 26 | 63 |
| La07 | 24 | 24 | 47 |
| La08 | 25 | 25 | 68 |
| La09 | 26 | 26 | 68 |
| La10 | 28 | 28 | 118 |
| La11 | 34 | 34 | 268 |
| La12 | 35 | 35 | 152 |
| La13 | 32 | 32 | 251 |
| La14 | 30 | 30 | 410 |
| La15 | 35 | 35 | 123 |
| La16 | 20 | 20 | 26 |
| La17 | 18 | 73 | 180 |
| La18 | 19 | 19 | 34 |
| La19 | 15 | 15 | 13 |
| La20 | 15 | 55 | 60 |
| La21 | 31 | 31 | 66 |
| La22 | 28 | 28 | 74 |
| La23 | 29 | 29 | 44 |
| La24 | 27 | 27 | 50 |
| La25 | 31 | 31 | 68 |
| La26 | 45 | 45 | 254 |
| La27 | 47 | 47 | 164 |
| La28 | 54 | 54 | 214 |
| La29 | 52 | 165 | 2246 |
| La30 | 44 | 44 | 199 |
| La31 | 97 | 97 | 5331 |
| La32 | 106 | 106 | 2097 |
| La33 | 97 | 97 | 1531 |
| La34 | 97 | 97 | 4566 |
| La35 | 68 | 68 | 632 |
| La36 | 38 | 43 | 1991 |
| La37 | 35 | 35 | 2301 |
| La38 | 34 | 34 | 124 |
| La39 | 34 | 42 | 185 |
| La40 | 31 | 31 | 101 |
| Somme | | 1695 | 24206 |

TABLE 5.9 – *NeighborIndirectRel*

| | N4 | N5 | N6 |
|---|---|---|---|
| La01 | 10 | 10 | 15 |
| La02 | 6 | 6 | 6 |
| La03 | 9 | 9 | 20 |
| La04 | 12 | 12 | 21 |
| La05 | 9 | 9 | 17 |
| La06 | 16 | 16 | 40 |
| La07 | 20 | 20 | 36 |
| La08 | 18 | 18 | 94 |
| La09 | 17 | 17 | 44 |
| La10 | 20 | 20 | 154 |
| La11 | 19 | 19 | 321 |
| La12 | 19 | 19 | 98 |
| La13 | 22 | 22 | 351 |
| La14 | 15 | 15 | 251 |
| La15 | 20 | 20 | 41 |
| La16 | 12 | 12 | 21 |
| La17 | 14 | 69 | 179 |
| La18 | 14 | 14 | 35 |
| La19 | 11 | 11 | 26 |
| La20 | 8 | 13 | 27 |
| La21 | 25 | 25 | 81 |
| La22 | 22 | 22 | 73 |
| La23 | 22 | 22 | 60 |
| La24 | 22 | 22 | 63 |
| La25 | 22 | 22 | 94 |
| La26 | 37 | 37 | 276 |
| La27 | 37 | 37 | 182 |
| La28 | 30 | 30 | 181 |
| La29 | 39 | 41 | 832 |
| La30 | 32 | 32 | 187 |
| La31 | 64 | 64 | 9218 |
| La32 | 59 | 59 | 1768 |
| La33 | 70 | 70 | 1949 |
| La34 | 61 | 61 | 2438 |
| La35 | 88 | 88 | 334 |
| La36 | 32 | 112 | 2374 |
| La37 | 32 | 32 | 1618 |
| La38 | 23 | 23 | 228 |
| La39 | 29 | 599 | 2212 |
| La40 | 25 | 25 | 188 |
| Somme | | 1774 | 26153 |

TABLE 5.10 – *CriticalPath*

de deux instances. Par exemple pour La36, les deux méthodes ont amélioré la valeur de N6, ce qui n'est pas le cas pour La17, La20, La29 et La39 où une des méthodes a été plus performante que l'autre.

Cette amélioration des résultats de ces deux méthodes par rapport à *PredOrder* est dû au fait que le nombre de nœuds à traiter pour prouver l'optimalité de la solution obtenue est plus faible si le nombre des premiers sous-nœuds de l'arbre de la méthode exacte est plus petit. En effet, aux premiers niveaux de l'arbre, la borne inférieure proposée n'est pas toujours très précise (surtout sur les quatre instances La17, La20, La29 et La39). Même si la solution optimale est trouvée plus tôt, les nœuds sur la droite de l'arbre doivent être traités afin de prouver l'optimalité de cette solution. Avec la méthode *NeighborIndirectRel*, les nœuds ayant le plus grand nombre de relations avec les autres groupes sont traités à la fin, tandis que ceux ayant le plus petit nombre de relations sont traités en premier. Cela réduit la largeur de l'arbre dans les premiers niveaux, où les bornes inférieures sont moins précises. Par conséquent, cela réduit l'espace de recherche pour prouver l'optimalité de la meilleure solution trouvée.

Les figures 5.9, 5.10 et 5.11 illustre cette propriété sur notre exemple de flowshop présenté dans le tableau 5.7 et la figure 5.7. Dans ces trois figures, chaque nœud représente une séquence d'opérations dans un groupe d'opérations permutables.

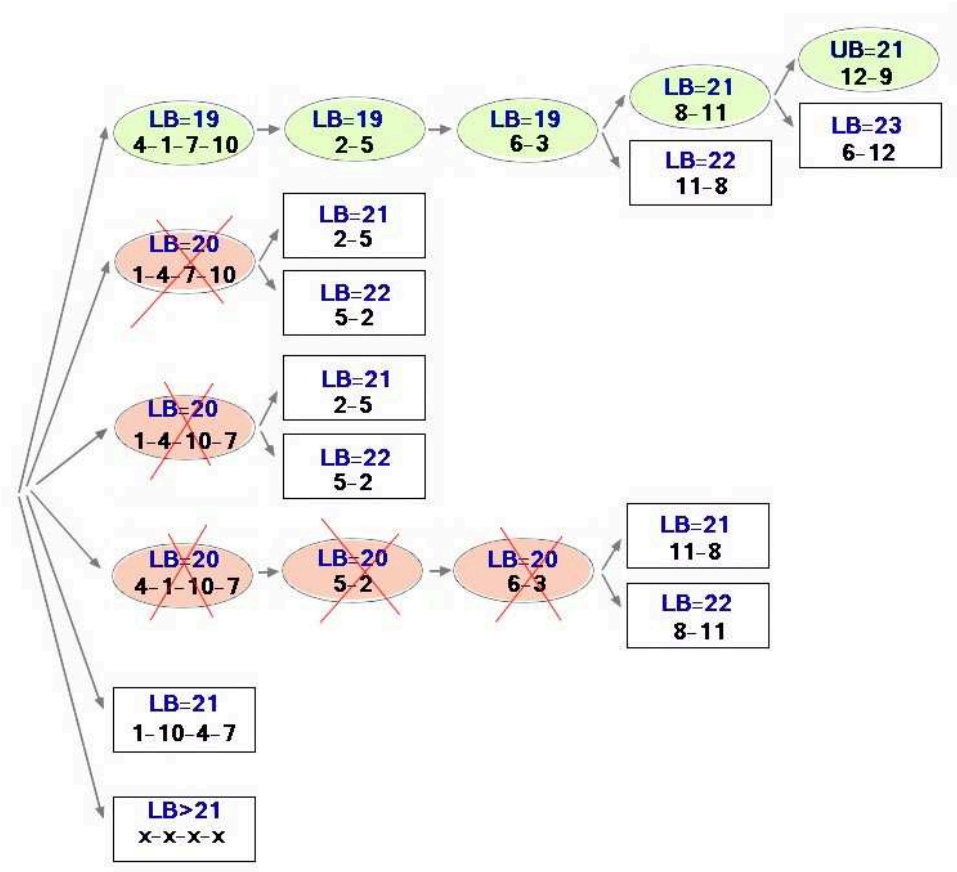



FIGURE 5.9 – *PredOrder*

Pour les trois méthodes, le meilleur des cas est trouvé après la visite de cinq

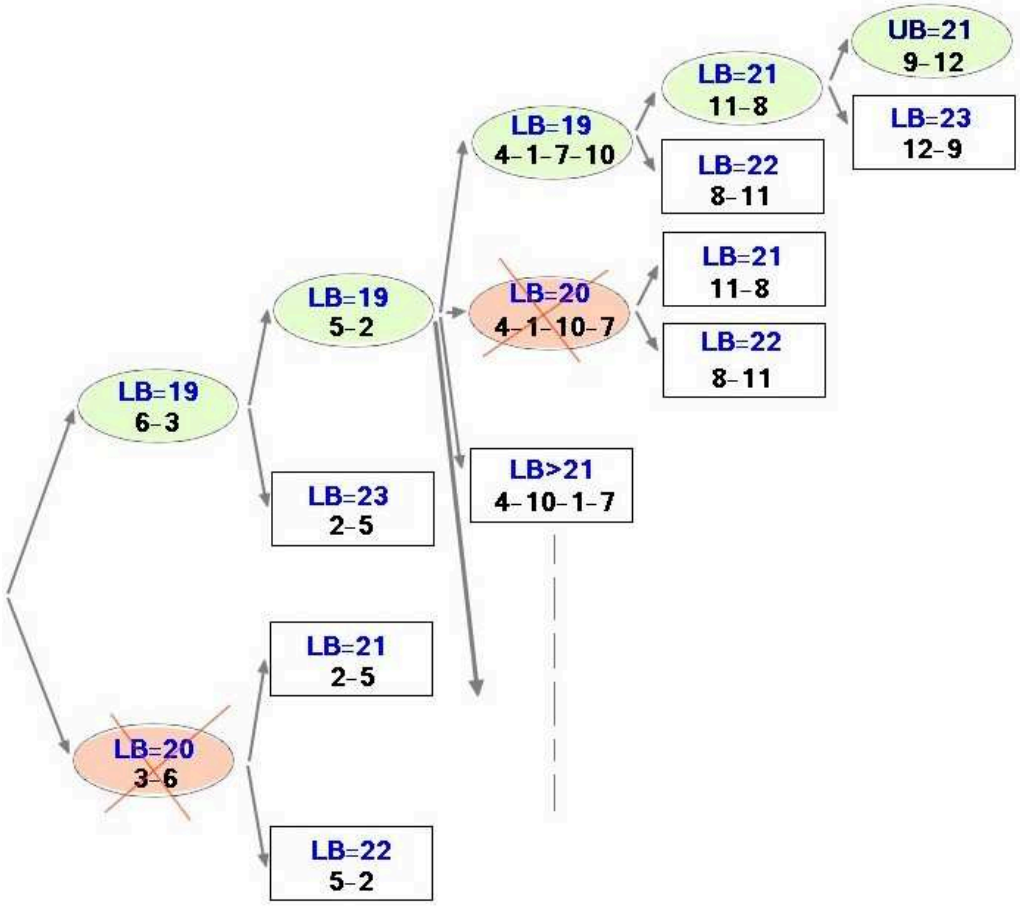



FIGURE 5.10 – *NeighborDirectRel*

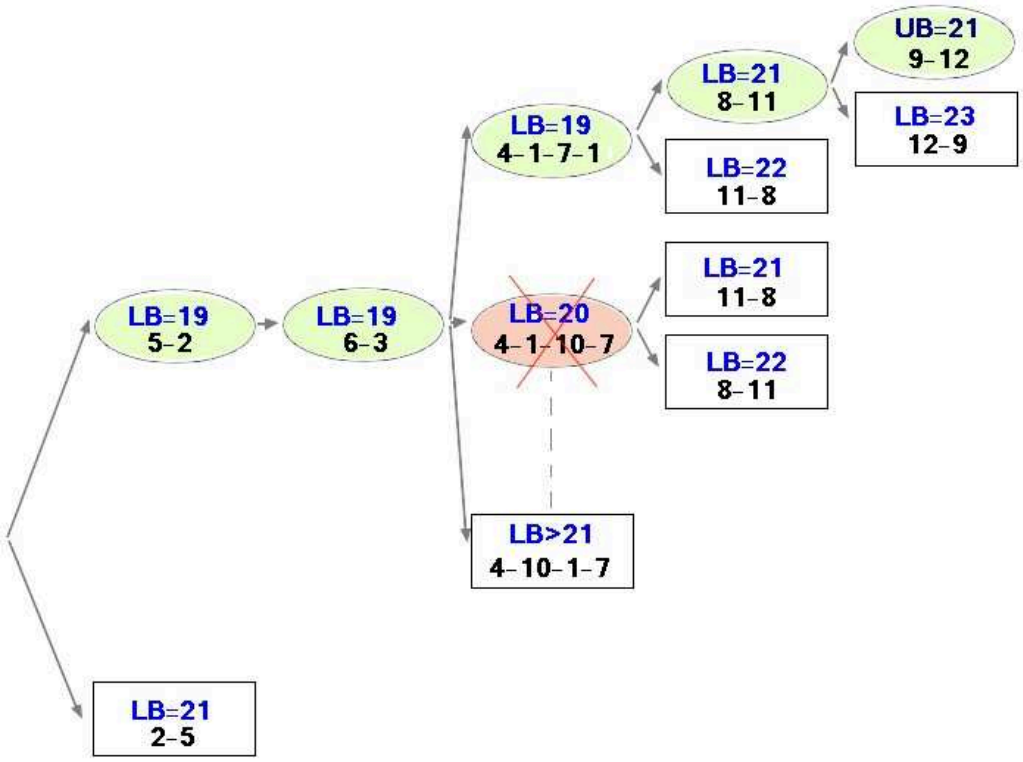



FIGURE 5.11 – *NeighborIndirectRel*

nœuds. Cependant, pour prouver son optimalité, *PredOrder* a visité cinq autres nœuds (en couleur rouge), par contre *NeighborDirectRel* et *NeighborIndirectRel* ont visité respectivement deux et un seul nœud. Avec la méthode de recherche en profondeur, la procédure de séparation *PredOrder* a généré vingt-quatre nœuds au début de l'algorithme de séparation et d'évaluation, cela mène à créer plus de sous-nœuds par la suite. Par conséquent, il faut plus de temps pour prouver l'optimalité de la meilleure solution trouvée. Par contre, *NeighborDirectRel* et *NeighborInDirectRel* commencent avec seulement deux nœuds, puisque le premier groupe de séparation est le groupe ayant le moins de relations avec les autres voisins. Le traitement de ce groupe en premier a permis d'améliorer la borne inférieure de telle sorte qu'elle soit plus précise sur les sous-nœuds générés, ce qui a permis de réduire l'espace de recherche (Figure 5.10 et figure 5.11).

Notre objectif dans la partie suivante est d'étudier l'intérêt du meilleur des cas dans un système d'aide à la décision, dont le but est d'utiliser plus efficacement la méthode d'ordonnancement de groupes.

# Troisième partie

# Coopération homme-machine et aide à la décision

# L'ordonnancement de groupes : une méthode qui favorise la coopération homme-machine

Comme mentionné dans le chapitre 4, l'ordonnancement de groupes est composé de trois étapes. La dernière étape qui représente la phase de décision consiste à sélectionner la séquence des opérations dans des groupes d'opérations permutables. Dans un contexte industriel, la sélection de chaque opération dans un groupe d'opérations permutables doit se faire en un temps très court. Afin de satisfaire cette contrainte de temps, des heuristiques d'ordonnancement sont utilisées sinon cette sélection est faite par **un opérateur humain**. Son rôle est de choisir en temps réel la prochaine opération à exécuter dans un groupe d'opérations permutables. Afin que l'opérateur ou l'heuristique de décision soit efficaces, des critères de mesure sont utilisés avant chaque décision. Ce chapitre est focalisé sur les critères de mesure de l'ordonnancement de groupes, dont le meilleur des cas fait partie. Dans la section suivante, nous présentons un récapitulatif sur certains travaux de la littérature sur l'intervention de l'humain dans l'ordonnancement.

## 6.1 Le facteur humain dans l'ordonnancement

L'intervention de l'humain dans l'ordonnancement et la planification d'une manière générale est une question étudiée et discutée depuis les années cinquante ; des premières observations expérimentales sur les modèles de prise de décisions de l'humain ont été formulées par Tversky et Kahneman (1974), Sanderson (1989) ; certains chercheurs affirment que cette prise de décision est rationnelle et algorithmique dans le sens où les décisions sont bien adaptées à l'environnement humain (Gigerenzer et al. 1999). L'idée générale de cet ouvrage ("*Simple heuristics that make us smart*") est la suivante : une prise de décision rapide à l'aide d'une heuristique simple pourra être aussi précise qu'une prise de décision à l'aide d'une système plus riche en informations et plus coûteux en temps. Ces heuristiques sont censées capturer la façon dont l'humain prend les décisions sous des contraintes de temps et connaissances limitées. Elles sont considérées comme des modèles reflétant les systèmes artificiels ou un comportement de l'humain.

Cependant, des tentatives ont été prises pour réduire ou remplacer l'humain dans l'ordonnancement et la planification, en automatisant le processus de décision (Crawford et al. 1999). Dans ce contexte, des systèmes experts ainsi que des agents intelligents ont été développés par des chercheurs sur l'intelligence artificielle. L'objectif est de modéliser le comportement de l'humain afin d'effectuer des tâches cognitives complexes (Ammons et al. 1986, Bui et Lee 1999, Trentesaux et al. 2000). L'argument principal de ces approches, est que l'intelligence artificielle peut inclure l'expertise humain dans le processus d'ordonnancement, ce qui n'est pas le cas pour les approches issues de la programmation mathématique, qui sont basées sur une structure rigide et statique incapable de représenter l'expertise et la connaissance de l'humain (Monfared et Yang 2007).

Aujourd'hui, les systèmes experts sont considérés comme moins appropriés en raison de la volatilité de l'environnement de production (Fox 1990) et l'humain est crucial pour la prise en compte des différentes contraintes d'adaptation aux changements (Crawford et al. 1999, Crawford et Wiers 2001, Jackson et al. 2004). En outre, les travaux de recherche d'aide à la décision ont révélé que le comportement de prise de décision de l'humain est un comportement non algorithmique et diffère quelque peu de la logique rationnelle, à cause de certains facteurs, qui se produisent en temps réel et ne peuvent pas être pris en compte à l'avance (Gasser et al. 2011).

Dans ce contexte, différentes études ont été faites en s'appuyant sur des analyses d'observations en situation réelle du comportement humain telle que l'étude de Jackson et al. (2004). Les auteurs ont proposé un modèle descriptif capturant le comportement du planificateur humain dans un contexte manufacturier afin d'assister au mieux le processus d'ordonnancement. L'étude proposée a été réalisée sur des opérateurs de sept entreprises qui étaient observées pendant une durée d'une semaine. Le résultat de cette étude montre que la performance de l'opérateur en ordonnancement est influencée non seulement par sa contribution cognitive individuelle mais aussi par son comportement social et organisationnel dans l'environnement manufacturier.

Une étude similaire confirmant le résultat de Jackson et al. (2004) est présentée dans Berglunda et Karltuna (2007). Cette nouvelle étude avait pour but d'étudier les facteurs liés à l'humain influençant le processus de planification de la production. Les auteurs ont observé le comportement des opérateurs de quatre entreprises suédoises de construction de bois. Trois facteurs fondamentaux ont été définis (figure 6.1) :

— Facteur humain, englobant ses compétences sociales, biologiques, cognitives, psychologique et son jugement intuitif.
— Facteur organisationnel tel que la gestion et l'organisation des différentes activités qui interfèrent son activité prioritaire.
— Facteur technologique, tel que les outils, interfaces et logiciels primaires et secondaires qui doivent permettre une certaine souplesse durant les prises de décisions.

Cependant, dans le contexte général de l'interaction homme-machine, des questions préliminaires sur le contexte organisationnel doivent être abordées. Ces ques-

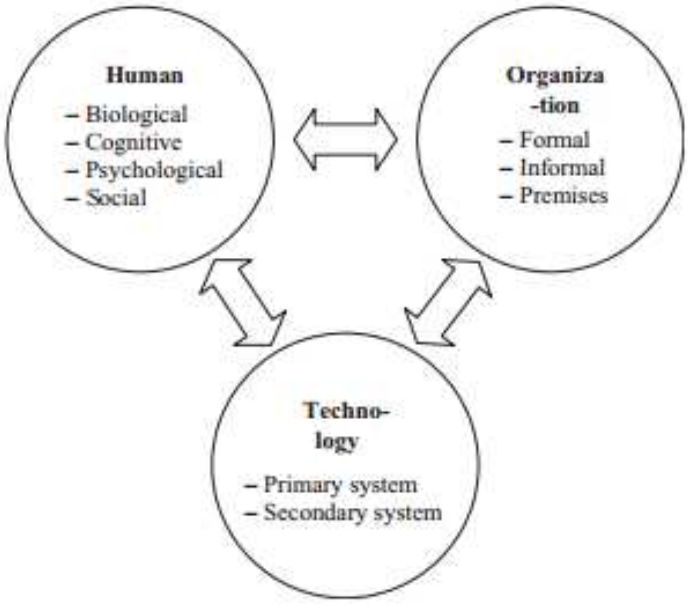



FIGURE 6.1 – *Impacts de l'humain sur l'ordonnancement (Berglunda et Karltuna 2007)*

tions portent principalement sur la liberté de contrôle d'ordonnancement assignée à chacune des deux entités : l'humain et la machine. Le travail de Wezel et al. (2011) caractérise cinq niveaux de contrôle :

— Contrôle manuel : dans ce mode de contrôle, toutes les décisions de l'ordonnancement sont prises par l'humain, la machine ne possède aucun contrôle de décision.
— Contrôle consultatif : ce mode est presque similaire au mode précédent où toutes les décisions sont prises par l'humain, mais la machine intervient pour vérifier la faisabilité des décisions prises par l'humain.
— Contrôle interactif : ce contrôle représente une solution intermédiaire où les décisions sont partagées entre l'humain et la machine.
— Contrôle de surveillance : dans ce mode, la machine joue le rôle de surveillance et de consultation sur les décisions, mais l'humain les valide.
— Contrôle automatique ou algorithmique : contrairement au premier mode de contrôle, toutes les décisions sont prises par la machine, généralement un algorithme qui remplace l'humain.

Ces modes de contrôle permettent de déterminer la tâche de décision de chaque entité. Dans un contexte d'incertitudes comme un atelier manufacturier, le but n'est pas d'exclure une entité de l'autre (mode manuel ou automatique) mais de favoriser la coopération entre les entités afin d'obtenir une efficience maximale.

L'identification du mode de contrôle conduit à la définition du problème d'interaction homme-machine (IHM) représentant l'ensemble des outils de contrôle et de communication entre l'humain et la machine. Cependant, la conception d'un modèle IHM peut suivre des principes théoriques généraux fondées sur les théories cognitives et l'ergonomie (Gilovich et al. 2002, Hardman et Macchi 2003, Plessner et al. 2008).

L'article de Hoc et al. (2012) traitant l'analyse de l'expertise en ordonnancement avec le cas de la problématique d'emploi du temps, montre le manque d'interfaces et de logiciels présentant des perspectives ergonomiques et écologiques adaptées à l'humain afin de tirer profit de la coopération homme-machine. Ces interfaces doivent être basées sur la description de l'environnement de l'opérateur et sont dites écologiques parce qu'elles cherchent à expliciter les contraintes de l'environnement pour l'humain (Mebarki 2012).

Les perspectives de recherches sur les interfaces hommes-machines présentent donc un intérêt tout particulier en ordonnancement. Les travaux de la littérature sur les interfaces homme-machines s'appuient sur des analyses d'observations en situation réelle ou en situation de simulation faisant appel à des scénarios réalistes.

Gacias et al. (2012) proposent une interface d'aide à la décision homme-machine pour la gestion d'optimisation du problème d'ordonnancement des transports. Un ensemble d'algorithmes favorisant la coopération homme-machine en prenant en compte l'aspect recherche opérationnelle et cognitif ont été proposés et incorporés dans un système d'aide à la décision. Ces algorithmes sont développés sur une approche 3-phases et utilisent les trois modes de contrôle : consultatif, interactif et surveillance. L'efficacité de l'interface d'aide à la décision proposée est illustrée sur des benchmark à tailles réduites.

Dans une autre étude récente (Dimopoulos et al. 2015) proposent la définition d'un nouveau modèle d'IHM industriel appelé i-DESME. Ce modèle permet la création de logiciels d'aide à la décision en ordonnancement dans les PME et TPE. L'avantage du modèle proposé est qu'il considère les caractéristiques multidisciplinaires de l'aide à la décision liées au facteur humain, tel que l'ergonomie et l'aspect cognitif. L'approche proposée a permis d'identifier des algorithmes de recherche qui peuvent apporter un soutien à l'humain pour la mise en place de l'ordonnancement dans un environnement industriel.

## 6.2 IHM pour l'ordonnancement de groupes

Dans cette section, nous nous intéressons particulièrement à la définition d'une IHM adaptée à l'ordonnancement de groupes en prenant en compte le facteur technologique et l'aspect ergonomique de la coopération homme-machine. Les systèmes technologiques utilisés sont des interfaces logicielles capables de fournir à l'opérateur humain une assistance simple, performante et efficace sur les données de l'ordonnancement ainsi que le contrôle de ce dernier.

La figure 6.2 illustre le modèle IHM proposé pour l'ordonnancement de groupes. Le concept opérationnel de ce modèle exige que l'humain interagit avec la machine à l'aide d'un système d'aide à la décision et ceci uniquement lors de la phase en-ligne de l'ordonnancement.

Ce système d'aide à la décision représente le cœur de l'IHM proposée et doit four-

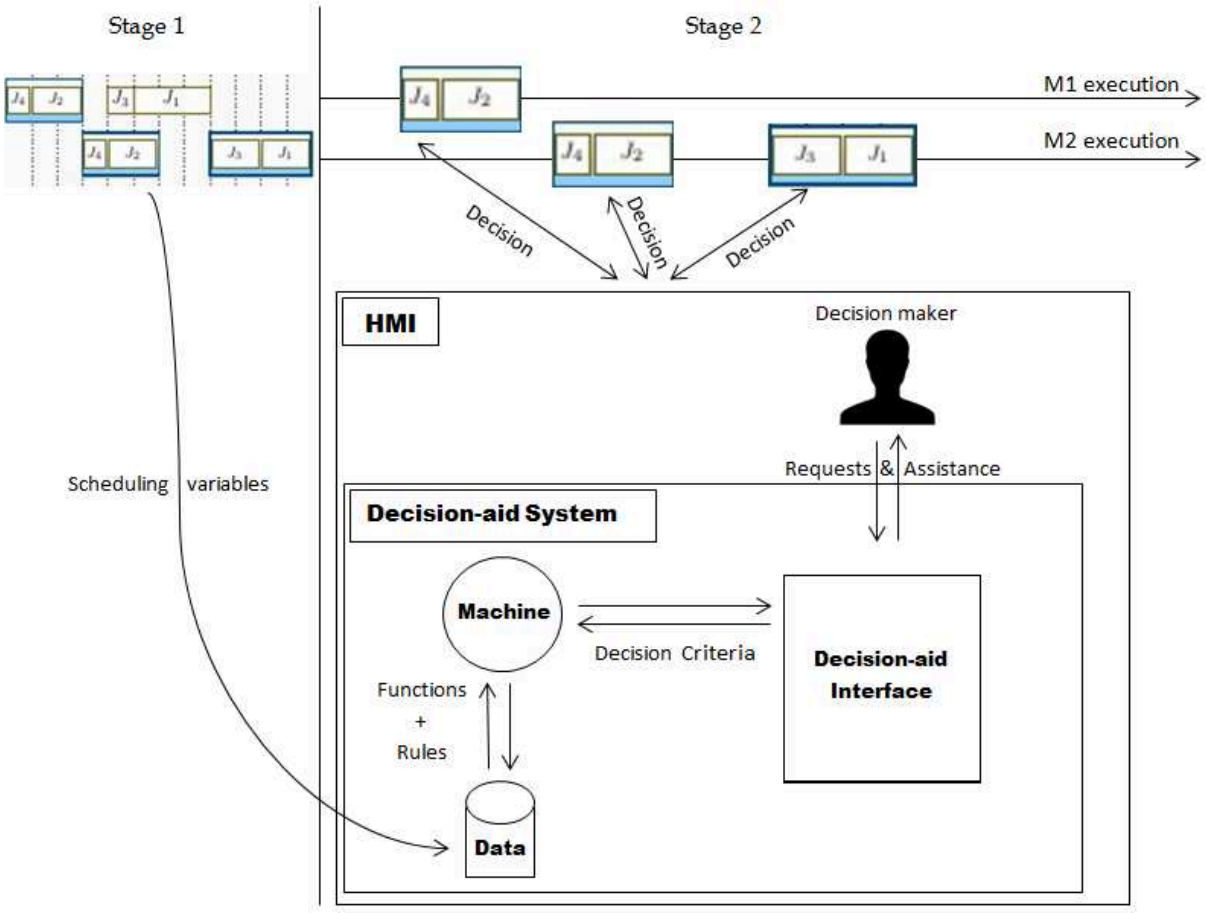



FIGURE 6.2 – *Architecture de l'IHM*

nir à l'opérateur toutes les informations dont il a besoin durant la phase de décision de l'ordonnancement de groupes. Ce système est composé de trois modules :

— *decision-aid interface*, qui représente le module permettant le dialogue entre l'opérateur humain et la machine. Ce module permet de répondre aux demandes de l'opérateur en lui fournissant les informations dont il a besoin. Il permet aussi de mettre en œuvre l'ordre d'exécution des opérations des jobs sur les machines en temps réel.

— *Data storage* : ce module permet d'enregistrer toutes les variables de l'ordonnancement calculées durant les deux premières étapes de l'ordonnancement de groupes.

— *Machine* : ce troisième module utilise les informations enregistrées dans le module précédent afin de calculer et répondre aux demandes du décideur.

Nous supposons que l'humain a le contrôle sur toutes les décisions et la machine prend le rôle d'assister, évaluer ainsi que valider ces décisions, car il est plus utile de faire calculer à la machine les conséquences d'une décision que de lui faire prendre la décision à la place de l'opérateur. En outre, l'humain reste assez performant pour prendre une décision, bien que non optimale à cause de la complexité et la difficulté du problème qui peut atteindre rapidement une explosion combinatoire. Mais, les machines, bien qu'elles soient conçues pour être optimales et efficaces en calculs complexes (que l'humain n'arrivera pas à faire), elles restent peu performantes dans des situations réalistes (Hoc et al. 2004). Toutefois, l'humain, dispose d'informations non informatisables auxquelles les machines ne peuvent pas avoir accès.

## 6.3 L'utilité du meilleur des cas dans un système d'aide à la décision

Dans la deuxième partie de ce manuscrit, nous avons proposé des méthodes de calcul du meilleur des cas d'un ordonnancement de groupes avec pour objectif l'utilisation de cette mesure en temps réel durant la phase de décision. Durant cette phase, le décideur sélectionne une opération d'un groupe d'opérations permutables. Pour chaque opération, est proposé un ensemble de critères, chacun apportant de l'information sur l'ordonnancement final (réalisé) si l'opération est choisie en première position de son groupe. Dans cette section, nous étudions l'intérêt de trois critères de mesure dont le meilleur des cas :

— *LB_best-case($O_{k,j}$)* représente la borne inférieure du meilleur des cas, i.e., le meilleur ordonnancement atteignable, si l'opération $O_{k,j}$ est exécutée en premier dans son groupe. Le calcul de cette borne est polynomial

— *worst-case($O_{k,j}$)* représente la pire performance atteignable si l'opération $O_{k,j}$ est exécutée en première position de son groupe. Le calcul de la pire performance est polynomial et est basé sur la pire permutation dans chaque groupe comme montré dans les équations 4.1 et 4.2.

— *free-seq-margin($O_{k,j}$)* représente la marge libre séquentielle de l'opération $O_{k,j}$ et est calculé à l'aide de l'équation 4.4 indiquée dans la section 4.1.4. Contrairement aux deux critères précédents, ce critère ne peut être utilisé que pour des objectifs liés aux retards des jobs.

Afin d'évaluer ces critères dans le système d'aide à la décision de l'IHM proposée, nous proposons dans un premier lieu un algorithme de prise de décision simulant l'opérateur humain et ce, pour chacun des trois critères. L'algorithme *RA_best-case* décrit ci-après est un algorithme réactif qui simule le processus de décision en se basant sur le critère *LB_best-case($O_{k,j}$)*.

**Algorithme 2 :** *RA_best-case*

$L(G)$ représente la liste des groupes d'opérations permutables avec plus d'une opération.
**for** *chaque groupe* $G_i^k$ *dans* $L(G)$ **do**
  **while** $|G_i^k| > 1$ **do**
    *LB_best-case := valeur maximale*;
    **for** *chaque opération* $O_{k,j}$ *dans* $G_i^k$ **do**
      - Placer $O_{k,j}$ en première position et Calculer *LB_best-case($O_{k,j}$)* ;
      - *LB_best-case* := $\min_{O_{k,j} \in G_i^k}$(*LB_best-case*, $LB\_best - case(O_{k,j})$);
    - L'opération ayant la plus petite valeur est choisie, s'il y a une égalité, l'opération ayant la plus petite borne $r'_{k,j}$ est exécutée;
    - Supprimer l'opération choisie de $G_i^k$;
  - supprimer $G_i^k$ de $L(G)$.;

Cet algorithme est illustré sur notre exemple de job shop représenté dans le tableau 2.1 et la figure 4.1. Nous supposons que l'objectif de l'ordonnancement est de minimiser le retard maximum $T_{\max}$. Nous commençons d'abord par l'énumération de tous les groupes d'opérations permutables avec plus d'une opération.

— La liste des groupes d'opération permutable $L(G) = \{G_1^1, G_1^3\}$. L'algorithme a seulement deux décisions :
— Pour la première décision concernant le premier groupe $G_1^1 = (O_1, O_7)$ nous avons : *LB_best-case* = min( *LB_best-case($O_1$)* , *LB_best-case($O_7$)* ) = $\min(0, 2)$.
— Donc l'algorithme exécute l'opération $O_1$ en premier et la supprime du groupe $G_1^1$.
— Pour le deuxième groupe : *LB_best-case* = min ( *LB_best-case($O_5$)* , *LB_best-case($O_8$)* ) = $\min(0, 3)$.
— L'algorithme exécute $O_5$ en premier.
— À la fin, l'algorithme aurait exécuté l'ordonnancement optimal représenté dans la figure 4.3(a).

De manière similaire à *RA_best-case*, nous proposons *RA_worst-case* et *RA_free-seq-margin*. Pour *RA_worst-case*, l'opération avec la plus petite valeur de *worst-case($O_{k,j}$)*

est exécutée en premier. En cas d'égalité, l'opération placée en première position dans le groupe est exécutée en premier.

Cependant, pour le troisième algorithme utilisant la marge libre séquentielle, nous avons trois différentes interprétations pour les marges libres séquentielles des opérations du même groupe :

— Si toutes les marges libres séquentielles des opérations dans un même groupe sont positives, cela veut dire que quelle que soit l'opération choisie, l'ordonnancement final ne sera pas en retard.

— Si une ou plusieurs opérations (mais pas toutes) du même groupe ont des marges négatives (ou nulles), dans ce cas, l'ordonnancement final pourra être en retard, spécialement si les opérations avec les marges libres séquentielles négatives sont exécutées en premier. Dans ce cas, il est recommandé d'exécuter les opérations ayant les plus grandes marges libres séquentielles car les marges des autres opérations peuvent ensuite augmenter et redevenir positives.

— Si toutes les marges des opérations du groupe sont négatives ou nulles, quel que soit le choix de l'opération, l'ordonnancement final sera en retard.

Avec un tel critère, il est recommandé de choisir l'opération ayant la plus grande marge libre séquentielle afin de préserver la flexibilité de l'ordonnancement et ainsi anticiper le retard.

Ce critère est particulièrement intéressant durant la phase de décision de l'ordonnancement de groupes (Slowinski et Weglarz 1989, Lopez et Roubellat 2008, Billaut et al. 2008). Il a d'ailleurs été implémenté dans un logiciel industriel ORDO. Ce logiciel est décrit dans Roubellat et al. (1995), d'autres références peuvent également être trouvées dans Lopez et Roubellat (2008).

Le tableau 6.1 représente le calcul des marges (équation 4.4) de toutes les opérations de notre exemple de job shop défini dans le tableau 2.1 et la figure 4.1.

TABLE 6.1 – *Calcul des marges libre séquentielles*

| $O_i$ | $O_1$ | $O_2$ | $O_3$ | $O_4$ | $O_5$ | $O_6$ | $O_7$ | $O_8$ | $O_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $\rho_i$ | 1 | 4 | 1 | 2 | 3 | 1 | 4 | 2 | 3 |
| $d_i$ | 5 | 7 | 10 | 2 | 8 | 9 | 5 | 8 | 10 |
| $m_{\text{sn}}(O_i)$ | 4 | -2 | -1 | 0 | 3 | -2 | 1 | 1 | -2 |
| $m_{\text{sg}}(O_i)$ | 0 | -2 | -1 | 1 | 1 | -2 | 0 | -2 | -2 |
| $m_{\text{seq}}(O_i)$ | 0 | -2 | -1 | 0 | 1 | -2 | 0 | -2 | -2 |

On peut constater que pour la deuxième décision ($G_3^1$), l'opération $O_8$ présente une marge libre séquentielle négative, ce qui signifie que dans certains cas, le séquencement de cette opération peut conduire à des retards. On constate effectivement qu'en fonction des décisions prises, l'opération $O_5$ termine en retard dans deux séquences sur quatre (figure 4.3). Dans l'algorithme *RA_free-seq-margin* nous exécutons l'opération avec la plus grande marge libre séquentielle en premier.

### 6.3.1 Expérimentations et résultats

Ces trois algorithmes simulant les prises de décision ont été testés sur les quarante instances de Lawrence (Beasley 2004) avec les mêmes paramètres de regroupement que lors de la première expérimentation ($\epsilon = \infty$), mais cette fois, nous mesurons le retard maximum $T_{\max}$ au lieu du makespan. À partir de la solution de référence définie dans l'étape 0 de l'ordonnancement de groupes, nous supposons que les dates d'échéances des opérations sont égales à leurs dates de fin : $d_{k,j} = C_{k,j}$ ; dans ce cas, le retard maximum $T_{max}$ de l'ordonnancement de référence est nul.

Pour chaque instance, nous comparons le $T_{\max}$ de l'algorithme réalisé obtenu pour chacun des trois algorithmes de décision avec la meilleure solution possible (solution de référence initiale avec un retard maximum nul). Les résultats de ces expérimentations sont exposés dans le tableau 6.2.

Les valeurs obtenues montrent que l'algorithme *RA_best-case* a dominé les résultats sur presque toutes les instances sauf La17 et La18. Dans ces instances l'algorithme *RA_best-case* avait un retard moyen de 31.5 unités par rapport à *RA_free-seq-margin*. L'algorithme *RA_free-seq-margin* a dominé lui aussi l'algorithme *RA_worst-case* sur plus de 90% des instances avec un retard moyen meilleur de 50% par rapport au retard moyen de *RA_worst-case*.

Sur toutes les instances, *RA_worst-case* n'a trouvé aucun ordonnancement optimal. Cependant, *RA_free-seq-margin* a trouvé l'ordonnancement optimal de seulement trois instances (La10, La17 et La18) et *RA_best-case* a trouvé l'ordonnancement optimal de 13 instances dont la plupart sont des instances de cinq machines ; ceci est dû à la qualité de la borne inférieure qui est très précise sur ces instances (cf. tableau 5.4).

Par ailleurs, on peut constater la performance de chacun des critères de mesure à partir des résultats exposés sur les deux dernières lignes du tableau 6.2. Comme prévu, le pire des cas présente la pire performance ; la moyenne du retard maximum de l'algorithme de décision utilisant ce critère représente le double de celui de la marge libre séquentielle et douze fois plus de retard (maximum) comparé à l'algorithme utilisant le meilleur des cas. Le pire des cas est supposé aider le décideur à éviter les pires décisions excédant un certain seuil de performance, mais ne permet pas de choisir la meilleure décision parmi les bonnes décisions comme le fait le meilleur des cas.

Avec cette expérimentation, le meilleur des cas surclasse les autres critères et pourrait être choisi comme seul critère de décision. Cependant, un système d'aide à la décision mono-critère ne manque pas d'inconvénient ; par exemple, un système d'aide à la décision basé seulement sur la marge libre séquentielle, comme c'est le cas du logiciel d'ordonnancement ORDO, ne permet pas de connaître la prochaine opération optimale dans le groupe si toutes les opérations de ce dernier ont des marges négatives. De plus, la marge libre séquentielle ne peut être utilisée que pour les objectifs de retard et ne pourra pas être utilisée pour le makespan par exemple, ce qui n'est pas le cas pour le meilleur des cas. Cependant, même le meilleur des cas utilisé seul a

TABLE 6.2 – *Ordonnancement réalisé de l'algorithme de prise de décisions*

| Instances | *Nbre de décisions* | *RA_best-case* | *RA_worst-case* | *RA_free-seq-margin* |
|---|---|---|---|---|
| La01 | 35 | 0 | 230 | 140 |
| La02 | 32 | 33 | 237 | 48 |
| La03 | 35 | 0 | 200 | 119 |
| La04 | 36 | 65 | 145 | 268 |
| La05 | 35 | 0 | 175 | 76 |
| La06 | 56 | 0 | 183 | 62 |
| La07 | 58 | 32 | 260 | 36 |
| La08 | 59 | 0 | 298 | 286 |
| La09 | 59 | 0 | 363 | 387 |
| La10 | 56 | 0 | 206 | 0 |
| La11 | 80 | 0 | 147 | 76 |
| La12 | 79 | 0 | 314 | 171 |
| La13 | 82 | 0 | 351 | 225 |
| La14 | 81 | 0 | 217 | 158 |
| La15 | 78 | 114 | 400 | 157 |
| La16 | 57 | 5 | 239 | 23 |
| La17 | 64 | 31 | 129 | 0 |
| La18 | 60 | 32 | 446 | 0 |
| La19 | 61 | 45 | 574 | 63 |
| La20 | 61 | 44 | 336 | 76 |
| La21 | 96 | 24 | 219 | 393 |
| La22 | 101 | 24 | 884 | 427 |
| La23 | 96 | 52 | 561 | 308 |
| La24 | 100 | 82 | 250 | 209 |
| La25 | 100 | 64 | 335 | 321 |
| La26 | 147 | 77 | 753 | 327 |
| La27 | 144 | 32 | 538 | 381 |
| La28 | 137 | 11 | 388 | 195 |
| La29 | 140 | 40 | 590 | 354 |
| La30 | 142 | 60 | 551 | 128 |
| La31 | 231 | 0 | 477 | 114 |
| La32 | 236 | 0 | 711 | 248 |
| La33 | 236 | 59 | 859 | 210 |
| La34 | 228 | 24 | 412 | 302 |
| La35 | 227 | 43 | 606 | 314 |
| La36 | 137 | 61 | 647 | 237 |
| La37 | 140 | 82 | 859 | 271 |
| La38 | 137 | 59 | 463 | 454 |
| La39 | 139 | 111 | 574 | 143 |
| La40 | 139 | 36 | 705 | 475 |
| Moyenne | | 33.55 | 420.8 | 204.55 |
| Somme | | 1342 | 16832 | 8182 |

montré ses limites ; puisque l'ordonnancement optimal sans aucun retard n'a pas été atteint dans 67% des instances.

Pour ces raisons, il semble plus intéressant de privilégier un système d'aide à la décision multicritères. Pour évaluer la pertinence d'un tel système nous proposons de modifier une condition de l'algorithme 2 permettant de résoudre les conflits entre deux opérations d'un groupe ayant la même valeur du meilleur des cas, puisque ce critère surclasse les deux autres. Nous refaisons les expérimentations précédentes avec l'algorithme $RA\_best - case$. Mais, cette fois en cas d'égalité de valeurs du meilleur des cas pour des opérations appartenant au même groupe, l'opération est choisie soit selon le critère du pire des cas, soit selon la marge libre séquentielle, dénotés respectivement $RA\_best - case_{WC}$ et $RA\_best - case_{FSM}$.

Les figures 6.3 et 6.4 représentent respectivement l'écart d'amélioration ou de dégradation de $RA\_best - case_{WC}$ et $RA\_best - case_{FSM}$ par rapport à $RA\_best - case$ (les instances sans aucune modification des résultats ne sont pas représentées dans les deux figures).

L'algorithme $RA\_best - case_{WC}$ n'a amélioré aucune instance et a dégradé le résultat de 37 instances. Le retard moyen de cet algorithme sur toutes les instances a été dégradé de plus de 28 unités de temps par rapport au retard moyen obtenu par l'algorithme $RA\_best - case$. Par contre, l'algorithme $RA\_best - case_{FSM}$ a dégradé le retard de 13 instances et a amélioré le retard de 18 instances. L'algorithme multi-critères $RA\_best - case_{FSM}$ a pu trouver la solution optimale de 17 instances (contre 13 instances avec $RA\_best - case$). Le retard moyen obtenu avec cet algorithme multi-critères est 25.15 et représente une amélioration de 25% du retard moyen de l'algorithme mono-critère $RA\_best - case$.

Cependant, l'environnement de cette expérimentation est déterministe et les variables du problème sont statiques tout au long du processus de décision. Dans la section suivante, nous allons évaluer la performance de $RA\_best - case$ dans un environnement de simulation perturbé.

**Performance du meilleur des cas en présence de perturbations**

Le but de cette section est de mesurer la dégradation de la performance des résultats de l'algorithme de prise de décisions en cas de *mauvaises décisions* due à une présence de perturbation qui transforme une décision optimale en décision non-optimale. Une décision sera considérée "mauvaise", si dans un groupe d'opérations permutables, l'opération avec la pire valeur de *LB_best-case*($O_{k,j}$) est choisie en premier.

Par exemple, dans le premier groupe d'opérations permutables du job shop représenté dans le tableau 2.1 et la figure 4.1, la "mauvaise" décision consiste à exécuter l'opération $O_7$ avant $O_1$, car *LB_best-case*($O_1$)=0 et *LB_best-case*($O_7$)=2 (pour le $T_{\max}$). Cependant, dans un groupe de plus de deux opérations, une mauvaise décision consiste à exécuter en premier l'opération qui dégrade le plus la fonction objective (ceci représente le pire scénario qui pourra arriver). En cas d'égalité de valeurs

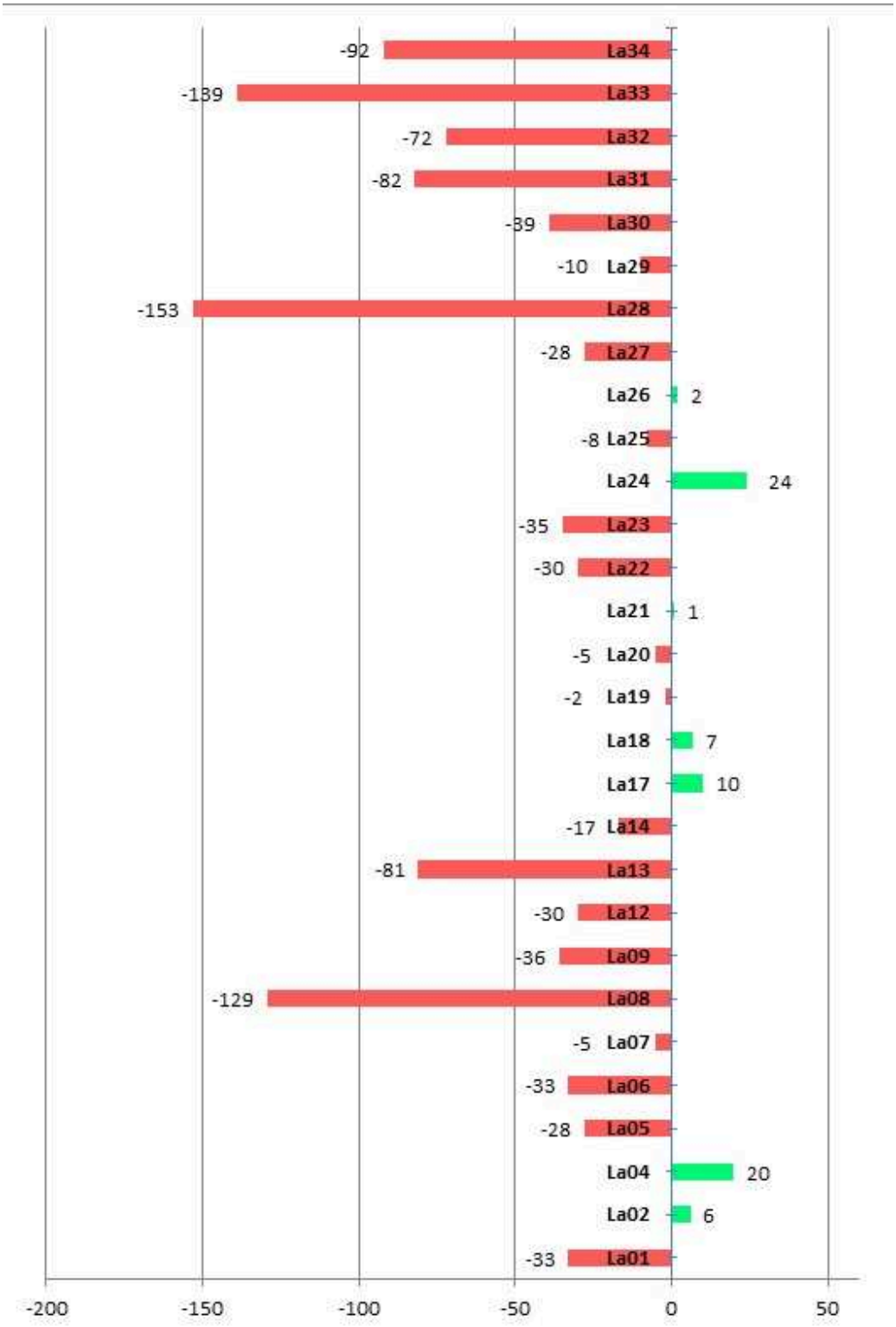



FIGURE 6.3 – $RA\_best - case_{WC}$

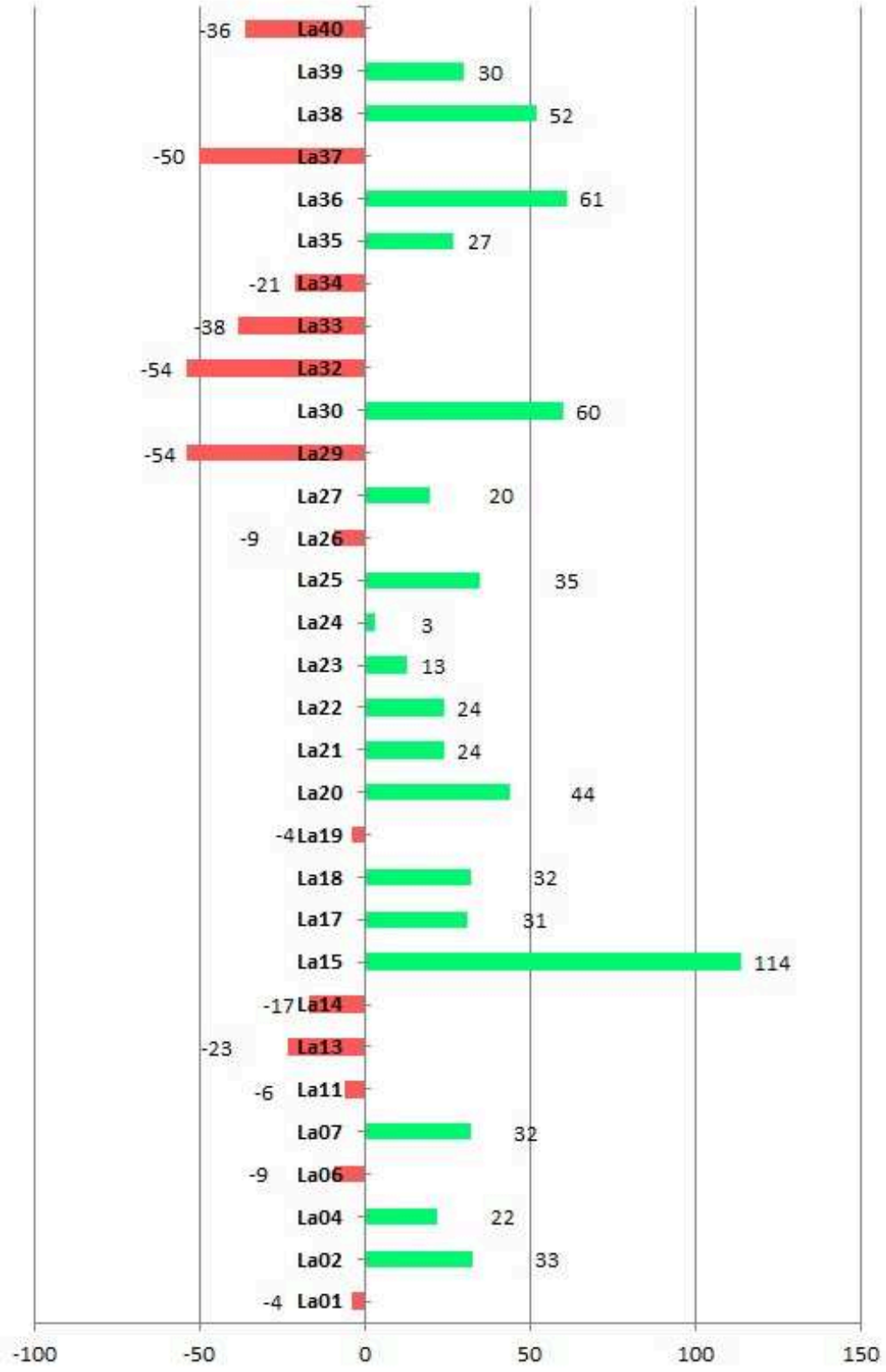



FIGURE 6.4 – $RA\_best - case_{FSM}$

concernant le critère du meilleur des cas, alors l'opération ayant la marge libre séquentielle la moins élevée est considérée la mauvaise décision.

Nous cherchons à déterminer le nombre de *mauvaises décisions* nécessaires pour détériorer le $T_{\max}$ de l'ordonnancement réalisé avec un facteur de $\Delta$ de retard maximum. Ce facteur est un pourcentage calculé par rapport au retard maximum de l'ordonnancement de référence initial. La dégradation de la performance du *meilleur des cas* pourra être calculée de manière similaire à l'équation 5.9. Cette performance est égale à $\underbrace{f}_{ordo.\,de\,ref.} X\,(1+\Delta)$.

Pour illustrer le rapport entre le nombre de mauvaises décisions et $\Delta$ sur un petit exemple, nous reprenons notre ordonnancement de groupes décrit dans la figure 4.1 et le tableau 2.1. Nous supposons que la mauvaise décision est d'exécuter $O_7$ avant $O_1$ pour le $C_{\max}$. Dans ce cas, le $C_{\max}$ de l'ordonnancement réalisé obtenu par l'algorithme $RA\_best-case$ est égale à 12 et $\Delta = 20\%$. Cela signifie qu'il est possible de prendre une seule mauvaise décision pour une performance dégradée de 20% par rapport au $C_{\max}$ optimal.

Pour déterminer le nombre de *mauvaises décisions* nécessaires pour dégrader la fonction objective dans l'expérimentation suivante, nous incorporons les mauvaises décisions aléatoirement dans l'algorithme $RA\_best-case$ au cours du processus d'aide à la décision. L'algorithme est exécuté une dizaine de fois sur chaque instance. Le nombre moyen de *mauvaises décisions* nommé *Nb MD* est mesuré en fonction de la moyenne de $\Delta$ pour chaque type d'instances comme indiqué sur le tableau 6.3 (*Nb D* représente le nombre moyen de décisions totales). L'objectif traité est le $T_{\max}$. Nous supposons que la date d'échéance d'un job est égale à deux fois la somme des durées opératoires de ses opérations.

TABLE 6.3 – *Relation entre $\Delta$ et le nombre de mauvaises décisions*

| | | | $\Delta \approx 10\%$ | | $\Delta \approx 20\%$ | | $\Delta \approx 40\%$ | |
|---|---|---|---|---|---|---|---|---|
| Instances | n X m | *Nb D* | *Nb MD* | $\Delta$ | *Nb MD* | $\Delta$ | *Nb MD* | $\Delta$ |
| La01..La05 | 10 X 5 | 34.6 | 3.4 | 8.25 | 6.2 | 18.68 | 9 | 37.19 |
| La06..La10 | 15 X 5 | 57.6 | 7.2 | 10.23 | 9.2 | 19.3 | 13.6 | 39.15 |
| La11..La15 | 20 X 5 | 80 | 10.6 | 9.54 | 15.6 | 19.53 | 24.8 | 38.52 |
| La16..La20 | 10 X 10 | 60.06 | 4.6 | 9.972 | 10 | 20.05 | 11.4 | 39.99 |
| La21..La25 | 15 X 10 | 98.6 | 5.6 | 10.25 | 9 | 19.95 | 15.2 | 39.92 |
| La26..La30 | 20 X 10 | 142 | 6 | 9.15 | 11.2 | 19.86 | 20.4 | 40.28 |
| La31..La35 | 30 X 10 | 231.6 | 17.8 | 10.62 | 22.2 | 20.46 | 37.4 | 40.24 |
| La36..La40 | 15 X 15 | 138.4 | 4.2 | 10.06 | 7.6 | 19.67 | 14.2 | 40.69 |

En comparant le nombre de mauvaises décisions indépendamment du nombre de décisions totales, nous remarquons que les instances 10 X 5 ont le plus faible nombre de mauvaises décisions et les instances 30 X 10 ont le plus grand nombre.

TABLE 6.4 – *Relation entre Δ et le nombre de mauvaises décisions*

| | Δ ≈ 60% | | Δ ≈ 80% | | Δ ≈ 100% | |
|---|---|---|---|---|---|---|
| Instances | *Nb MD* | Δ | *Nb MD* | Δ | *Nb MD* | Δ |
| La01..La05 | 9.8 | 57.78 | 11.8 | 76.37 | 13 | 91.05 |
| La06..La10 | 17.4 | 57.79 | 19 | 74.68 | 21 | 88.54 |
| La11..La15 | 27.6 | 57.27 | 32 | 73.93 | 33.2 | 80.94 |
| La16..La20 | 14.2 | 59.55 | 17.6 | 78.12 | 21 | 98.05 |
| La21..La25 | 22.4 | 59.95 | 29.6 | 79.59 | 34 | 98.23 |
| La26..La30 | 28.2 | 60.77 | 34.8 | 80.43 | 41.4 | 99.72 |
| La31..La35 | 55.2 | 60.20 | 65.6 | 80.12 | 80.4 | 99.41 |
| La36..La40 | 25.8 | 60.03 | 32.6 | 80.15 | 39.6 | 100.30 |

Cependant, en comparant le pourcentage de nombre moyen de mauvaises décisions par rapport au nombre total tel que le montre la figure 6.5, nous constatons que les instances 20 X 5 sont les plus robustes par rapport aux mauvaises décisions avec des pourcentages allant de 13% ($\Delta = 9.54$) à 42% ($\Delta = 80.94$). En outre, les instances de 15 jobs et 15 machines ont le plus faible pourcentage de mauvaises décisions (de 3% ($\Delta = 10.06$) à 29% ($\Delta = 100.3$)).

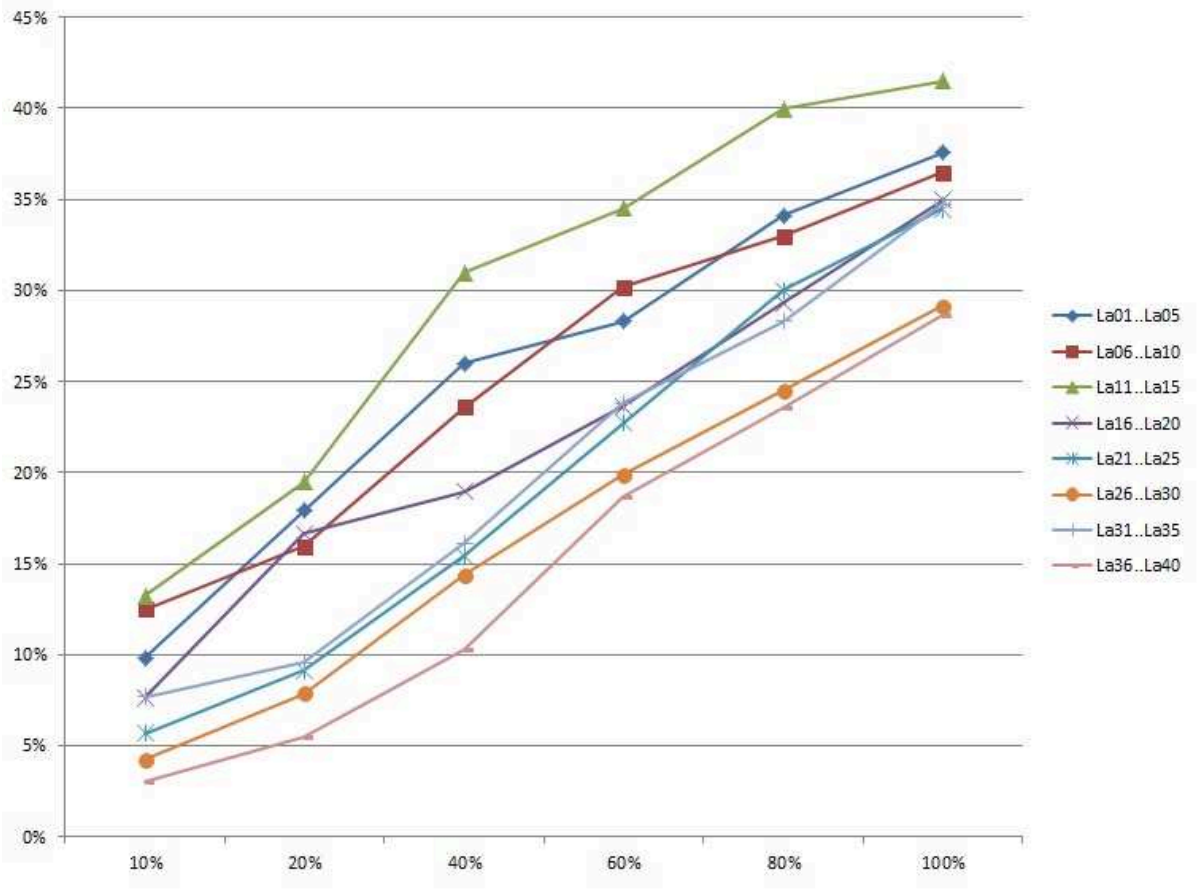



FIGURE 6.5 – *Le rapport entre le nombre moyen de mauvaises décisions et le nombre total de décisions*

Ces chiffres représentent des valeurs approximatives car les perturbations introduites sur toutes les instances sont aléatoires. Mais, nous pouvons définir des courbes de tendance linéaires pour chaque type d'instances de la figure 6.5. Dans ce cas, le pourcentage du nombre de mauvaises décisions est égal à : ($\alpha$ x $\Delta$) + $\beta$. Les valeurs minimales de $\alpha$ et $\beta$ sont respectivement 0.05 et 0.01.

Dans un système de coopération homme-machine, cette évaluation est intéressante pour estimer le nombre de perturbations tolérées par l'ordonnancement de groupes proposé. De plus, dans un système d'aide à la décision où l'humain est l'acteur principal des décisions, le nombre de perturbations tolérées permet de déclencher l'attention de l'opérateur afin qu'il soit plus attentif envers les décisions restantes pour que la performance attendue de l'ordonnancement ne soit pas trop dégradée.

# Étude de cas

L'objectif de ce chapitre est de valider les résultats théoriques obtenus dans les chapitres précédents. Pour cela, nous présentons une expérimentation d'un cas d'étude réalisé avec des étudiants jouant le rôle d'opérateurs de production.

## 7.1 Le meilleur des cas en pratique

Ce travail est motivé par les perspectives d'une expérimentation précédente réalisée par Mebarki et al. (2013) à laquelle j'ai participé lors du dépouillement des résultats. Cette précédente expérimentation a été réalisée avec l'équipe Psychologie Cognition et Technologie (PsyCoTec) de l'Institut de Recherche en Cybernétique de Nantes (IRCCyN) sur une chaîne de production flexible installée à l'Institut Universitaire de Technologie de Nantes (figure 7.1). Le protocole expérimental est détaillé dans la thèse de Guérin (2013). Dans cette section, nous présentons un résumé des résultats obtenus qui nous sert comme état des lieux pour notre nouvelle expérimentation présentée dans la section suivante.

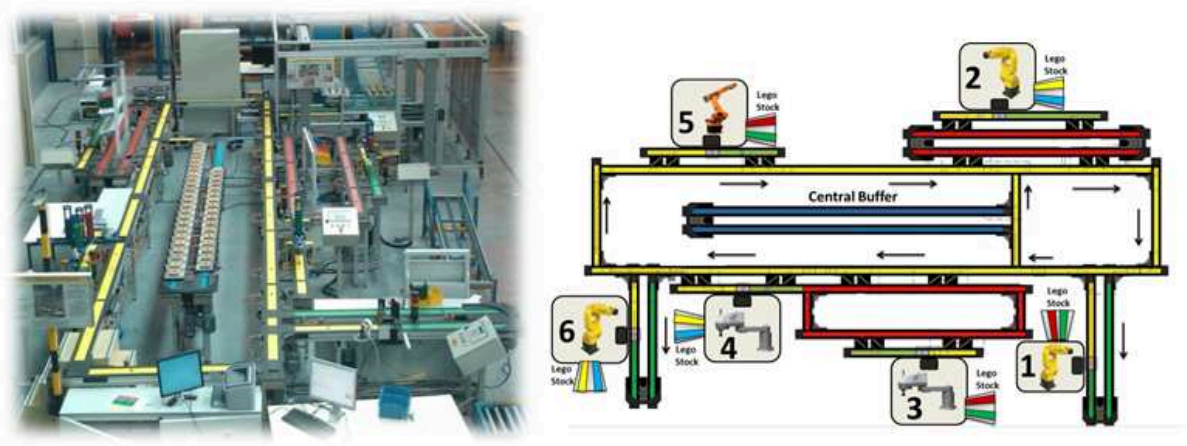



FIGURE 7.1 – *Chaîne de production inspirée de FILTRAUTO/SOGEFI*

Dans cette précédente expérimentation, un nouveau support d'aide à la décision multicritères a été proposé pour la phase en ligne de l'ordonnancement de groupes. Les critères de mesure proposés sont : la marge libre séquentielle, la séquence de l'opération dans la gamme, la durée de l'opération, le pire des cas et le meilleur des cas.

Cette précédente étude avait trois objectifs :

— Examiner l'intérêt d'un outil d'aide à la décision multicritères sur l'activité de l'opérateur dans un ordonnancement de groupes ; plus précisément évaluer l'utilisation de chacun des critères proposés sachant que les participants ont pour consigne de minimiser les retards.

— Étudier l'effet du contrôle mutuel sur les décisions d'ordonnancement ; plus précisément, le contrôle mutuel d'un agent "machine" sur l'activité d'ordonnancement d'un agent "humain". Pour cela, le mode de surveillance (Wezel et al. 2011) est utilisé. Chaque fois que l'opérateur humain prend une décision, la machine fait une contre-proposition, exerçant alors sur lui un contrôle mutuel.

— Examiner la gestion du risque de panne par les ordonnanceurs. Pour cela, les participants sont confrontés à deux scénarios : la présence d'un risque fort quant à la panne d'une machine contre l'absence d'un tel risque.

L'expérimentation a été réalisée sur un environnement de simulation avec neuf étudiants issus d'une formation de 3ème année génie industriel. Un psychologue a suivi les activités de chaque étudiant afin d'analyser les choix et les décisions de l'ordonnancement. Quatre résultats généraux ont été dérivés :

1. L'activité de l'opérateur humain a été amélioré avec le système d'aide à la décision multicritères comparé à un système mono-critère utilisant seulement la marge libre séquentielle
2. Dans le système multicritères, le meilleur des cas est sous-utilisé par rapport à la marge libre séquentielle. Cela est (probablement) dû aux conditions expérimentales.
3. Le contrôle mutuel n'a pas permis d'améliorer l'activité de l'humain.
4. Malgré l'amélioration de la qualité des décisions locales prises par les opérateurs, la performance de l'ordonnancement réalisé est dégradée.

Ce résultat prouve l'intérêt d'un système multicritères dans un IHM. La figure 7.2 illustre sous forme de boite à moustache, la proportion de bonnes solutions prises par l'opérateur ainsi que le temps moyen passé sur chaque décision. En comparant le système mono-critère de la marge libre séquentielle avec le système multicritère proposé, on constate que les opérateurs prennent plus de temps pour choisir une opération, ils sont donc plus actifs avec le système multicritère et cette amélioration de l'activité a permis d'améliorer la qualité des solutions obtenues par rapport au système mono-critère.

Toutefois, les conditions de l'expérimentation n'ont pas permis de répondre précisément à toutes les questions posées, en particulier concernant le deuxième et le quatrième point des résultats (i.e., sous-utilisation du meilleur des cas et dégradation de la performance globale d'ordonnancement).

Afin de remédier à ces lacunes, nous proposons un nouveau plan expérimental

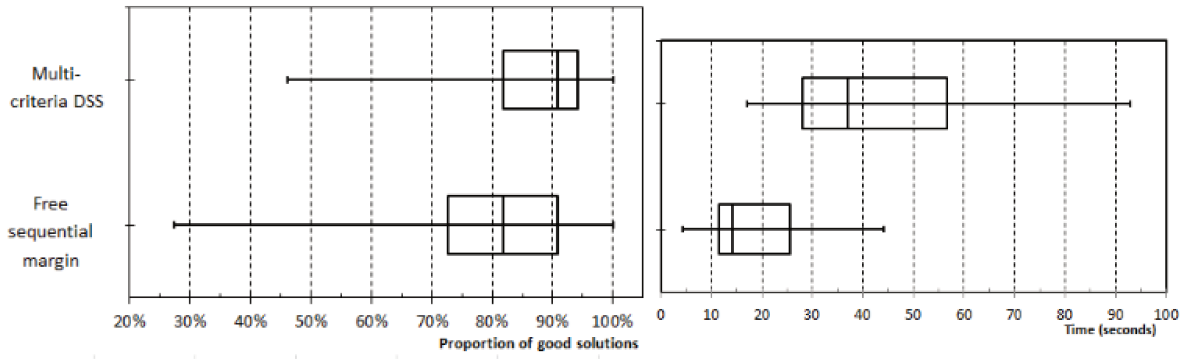



FIGURE 7.2 – *Système mono-critère VS multicritères*

basé sur le modèle proposé dans la figure 6.2. Cela permettra de reformuler certaines questions sur la modalité de la coopération homme-machine concernant :

1. Les critères de mesure adaptés à différents objectifs d'ordonnancement, en particulier le makespan et le retard maximum
2. L'intérêt du meilleur des cas dans un système d'aide à la décision multicritères
3. Le système d'aide à la décision multicritères et la qualité des décisions
4. La corrélation entre la qualité des décisions et la performance de l'ordonnancement réalisé

## 7.2 Nouvelle expérimentation

Afin d'implémenter le modèle IHM proposé dans cette section, une nouvelle expérimentation a été menée sur un échantillon de trente-cinq étudiants issus d'une formation en génie industriel. L'expérimentation a été réalisée sur une chaîne de production réelle appelée *MPS500* de l'entreprise allemande *FESTO*.

La chaîne *MPS500* représentée dans la figure 7.3 fabrique des vérins à faibles courses et est utilisée comme plate-forme de recherche et d'enseignement à l'université de Tlemcen.

*MPS500* est composée de six stations représentant un flow shop et contient huit palettes circulant sur un convoyeur de gauche vers la droite. Les palettes sont utilisées pour le transport des produits entre deux stations voisines ; chaque palette ne peut transporter qu'un seul produit à la fois. Une fois qu'une palette libre arrive à la première station (station de distribution), un produit est récupéré de l'entrepôt pour être transféré aux stations suivantes, jusqu'à ce qu'il arrive à la dernière station où il sera trié et prêt pour la livraison.

### 7.2.1 Protocole de l'expérimentation

Afin de faciliter les conditions de l'expérimentation et accélérer le processus de fabrication, nous avons construit un logiciel d'émulation de la chaîne *MPS500* en uti-

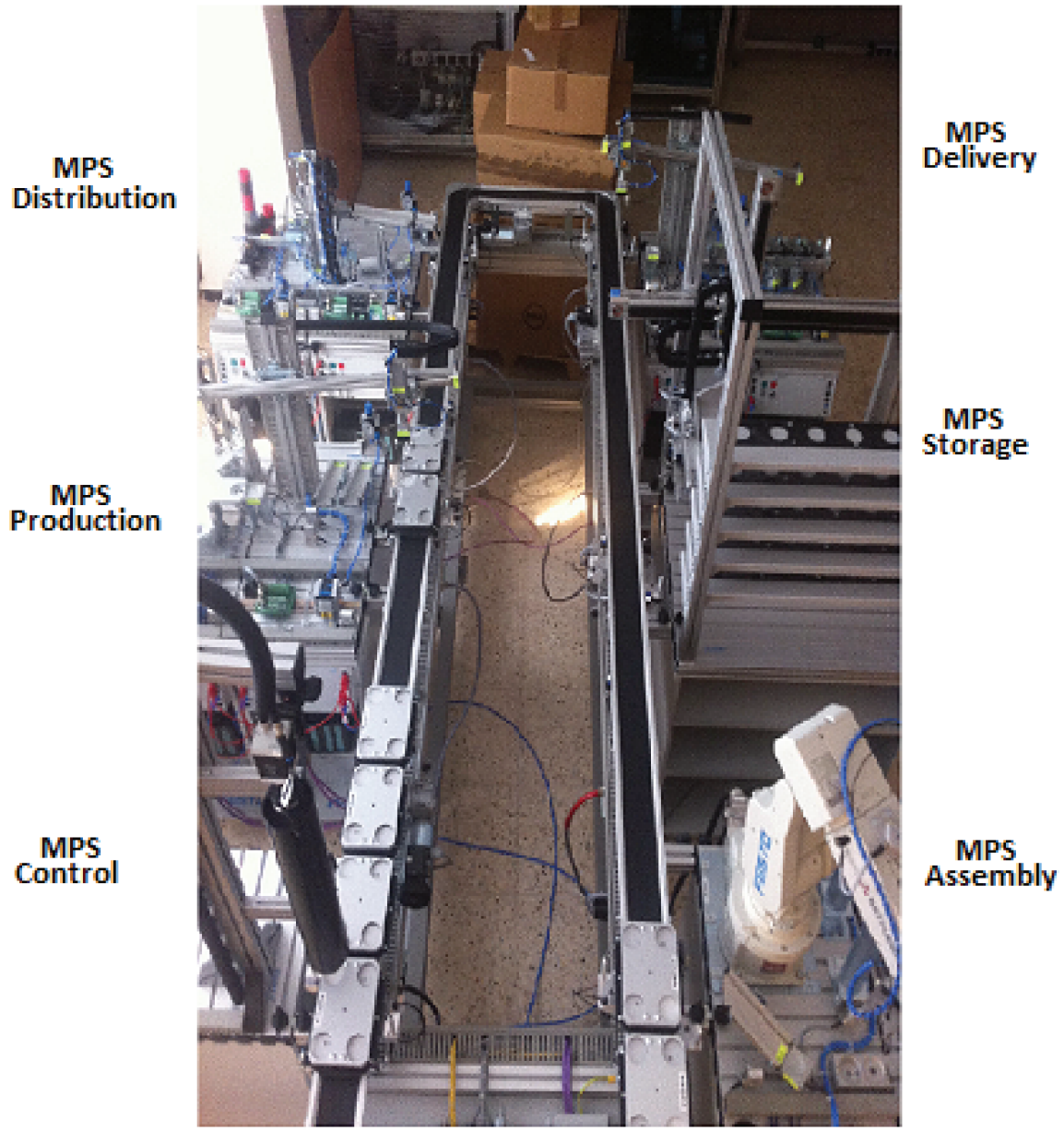



FIGURE 7.3 – *MPS500*

lisant la programmation concurrente sous JAVA. Ce logiciel simule le comportement dynamique des composants de *MPS500*.

Les données incorporées dans le module *Data storage* de l'IHM représentent un flow shop de de 20 jobs et 5 machines extrait de la référence Taillard (1993) (problème 20 X 5). Cette instance a été ensuite adaptée à la chaîne *MPS500* où une sixième machine à été ajoutée.

À partir de cette instance, une solution de référence contenant 8 groupes d'opérations permutables et caractérisant 12 décisions est construite en utilisant l'algorithme *EBJG*. Cependant, le regroupement a été réalisé uniquement sur la cinquième station de travail sous l'hypothèse que l'ordonnancement est possible dans cette station. L'ordonnancement de groupes est représenté dans la figure 7.4.

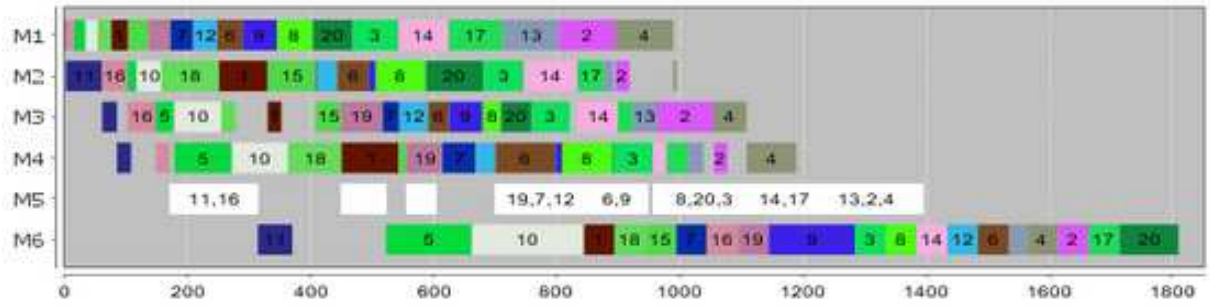



FIGURE 7.4 – *Simulateur de la chaîne MPS500*

La consigne donnée aux opérateurs (étudiants) est de prendre en temps réel les 12 décisions sur la cinquième station tout en minimisant les retards $T_{\max}$ et en optimisant le $C_{\max}$ à la fois.

L'interface du logiciel de simulation contient également un diagramme de Gantt de l'ordonnancement de référence tel qu'illustré dans la figure 7.5. Ce diagramme est mis à jour en temps réel chaque fois qu'une décision est prise. De plus, le temps de transport dans l'ordonnancement est considéré nul.

**"Decision-aid Interface" : interface d'aide à la décision**

Une fois que les opérations permutables arrivent à la file d'attente de la cinquième station, la simulation est interrompue par l'interface d'aide à la décision. Dans cette interface l'opérateur doit sélectionner la prochaine opération à exécuter dans le groupe. Cependant, pour aider l'opérateur dans ces décisions, cinq critères sont présentés dans cette interface :

1. Le meilleur des cas
2. Le pire des cas
3. La marge libre séquentielle de l'opération
4. La date de début au plus tôt de l'opération
5. La date d'échéance du job

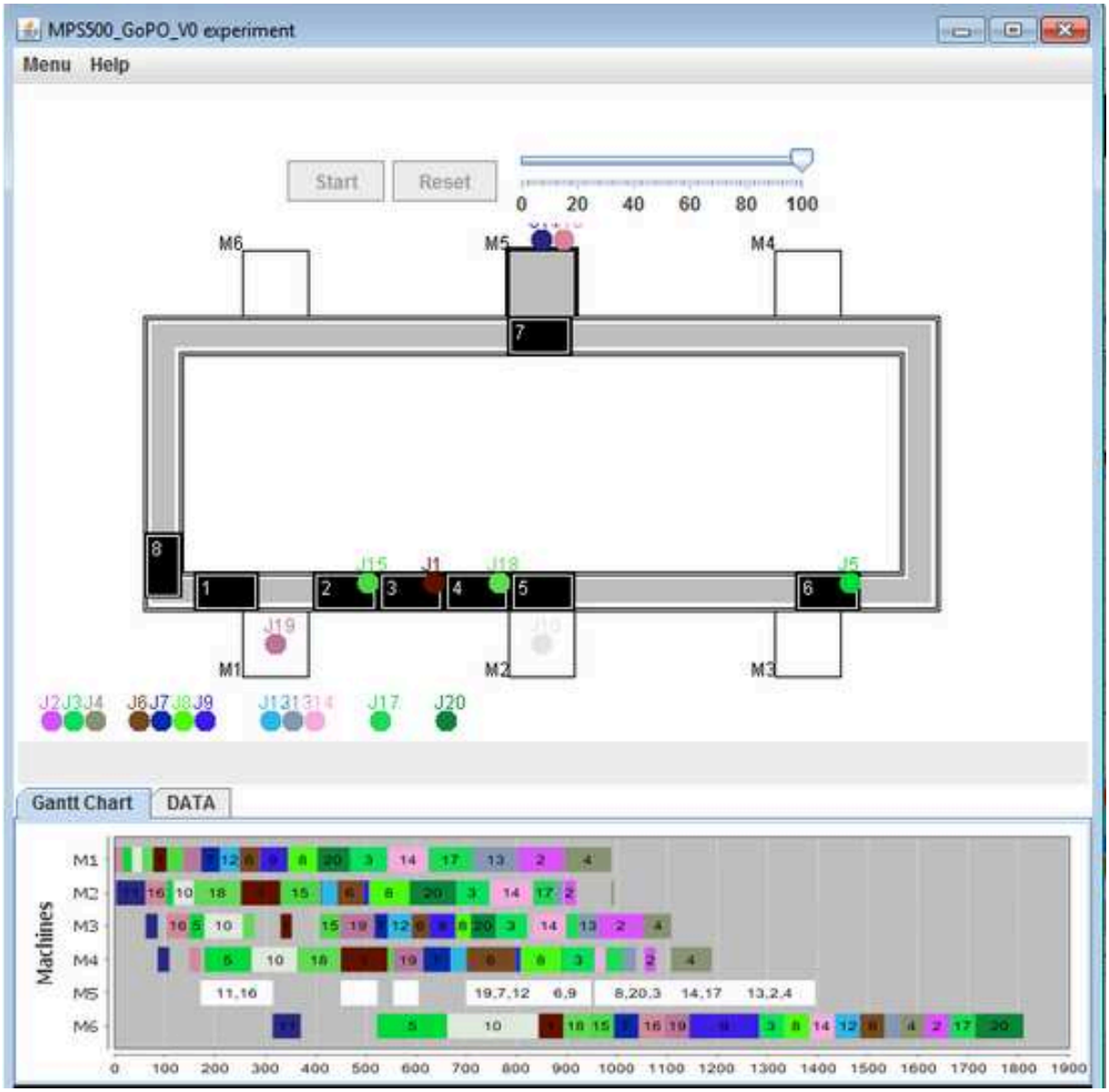



FIGURE 7.5 – *Simulateur de la chaîne MPS500*

Les trois premiers critères sont des critères d'évaluation de l'ordonnancement de groupes. Ils permettent d'anticiper la performance de l'ordonnancement réalisé si une opération dans un groupe d'opérations permutables est exécutée en premier. Pour le paramètre du meilleur des cas, nous utilisons la borne inférieure (équation 5.7) développée dans la section 5.3.

Les deux derniers critères ($r'_{k,j}$ et $d_{k,j}$ respectivement) permettent de donner des informations directes sur l'opération/job du groupe de la décision.

Cependant, avant de choisir un critère d'aide à la décision, l'opérateur doit d'abord sélectionner un critère de performance (i.e., $T_{\max}$ ou $C_{\max}$). De plus, la marge libre séquentielle et la date d'échéance contrairement aux autres critères ne sont affichées que pour le $T_{\max}$. La figure 7.6 illustre cette étape.

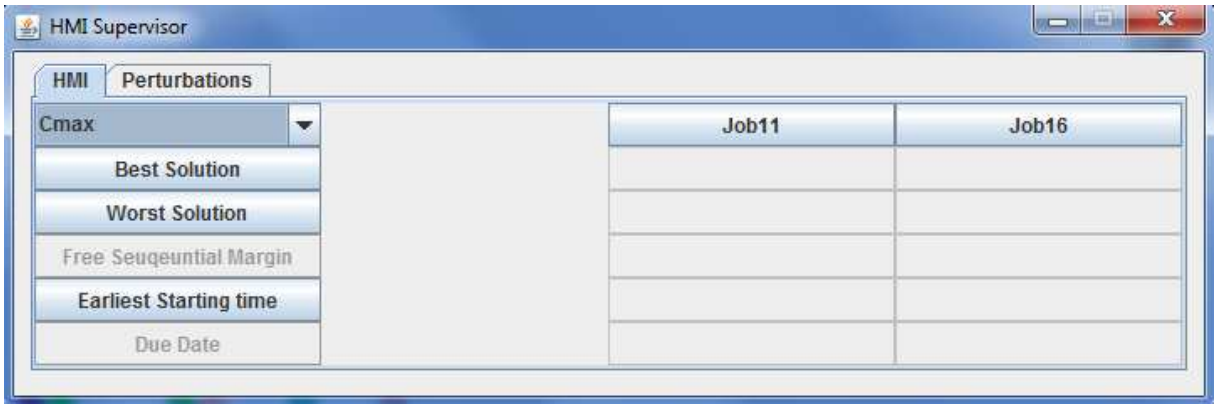



FIGURE 7.6 – *Interface de décision*

**Scénarios de l'expérimentation**

Dans une situation réaliste, différents types de perturbations peuvent se produire pendant l'étape 2 de l'ordonnancement de groupes (Billaut et al. 2008). Dans cette expérimentation, nous nous limitons au traitement des perturbations liées aux temps opératoires où certaines opérations prennent plus de temps à s'exécuter que prévu. Nous introduisons une première perturbation sur la cinquième décision et une autre perturbation sur la onzième.

Le module *Machine* calcule les critères d'aide à la décision demandés en fonction de l'objectif sélectionné sans prendre en compte les changements des temps opératoires causés par les deux perturbations. Par conséquent, les résultats affichés pour chaque critère peuvent être erronés. Cependant, l'opérateur doit détecter la présence de perturbations en vérifiant le temps de début/fin des jobs/opérations sur chaque machine. Sur l'exemple représenté dans la figure 7.7, il est montré que le job 6 a commencé (resp. terminé) son exécution à l'instant 626 (resp. 680) alors qu'il était censé commencer (resp. finir) à l'instant 614 (resp. 668).

Dans cette situation, l'opérateur doit vérifier si les résultats des critères demandés sont affectés ou non par la perturbation. Nous proposons deux scénarios :

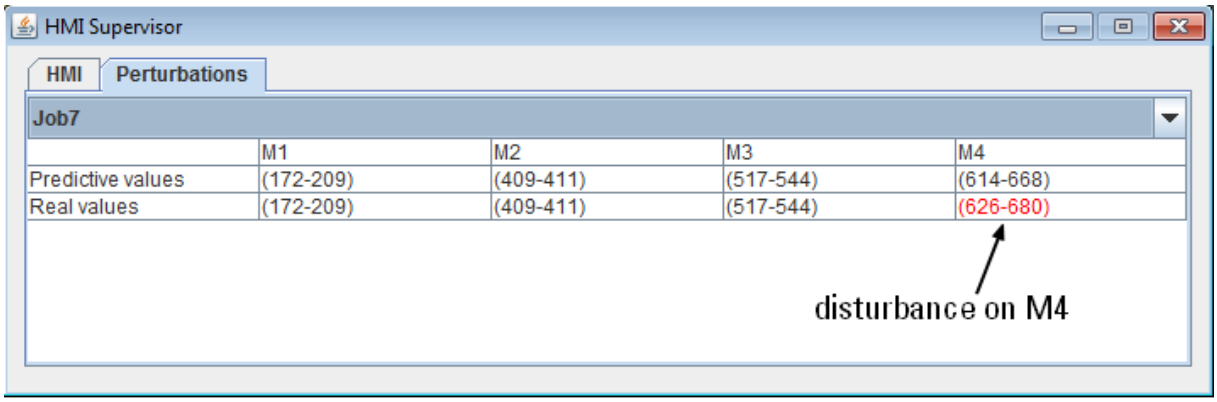



FIGURE 7.7 – *Détection des perturbations*

— Dans le premier scénario, l'opérateur doit simplement lui-même calculer par n'importe quel outil, les conséquences d'une perturbation détectée sur l'ordonnancement et les résultats des critères sollicités.
— Dans le deuxième scénario, contrairement au premier, les perturbations et leurs conséquences sont affichées directement sur le diagramme de Gantt (figure 7.8). Ce scénario permet de réduire la charge de travail mental de l'opérateur afin de renforcer sa concentration sur l'adaptation des résultats obtenus par les critères de décision.

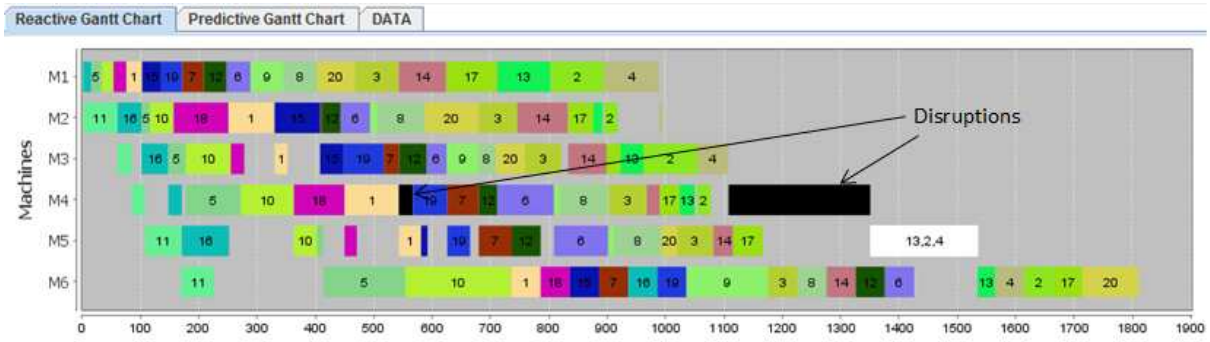



FIGURE 7.8 – *Représentation des perturbations en temps réel*

Afin d'étudier la performance relative à chaque scénario, les étudiants ont été décomposés en deux groupes. *GS_G1* représente l'ordonnancement réalisé par un étudiant du groupe 1 avec le scénario 1 et *GS_G2* représente l'ordonnancement réalisé obtenu par un étudiant du groupe 2 dans le deuxième scénario.

### Mesures

Afin de comprendre comment l'opérateur utilise les différents critères d'aide à la décision proposés pour chaque objectif, l'interface d'aide à la décision n'affiche qu'un seul résultat à la fois ; chaque fois qu'un critère est demandé, les résultats précédents sont effacés de l'interface. Toutefois, l'opérateur peut solliciter un même critère plusieurs fois.

Une fois que l'opérateur a arrêté son choix, il peut finaliser sa décision en sélection-

nant le prochain job à exécuter en première position. Ce processus est répété jusqu'à la fin de l'ordonnancement où tous les produits sont prêts à être livrés.

La machine enregistre précisément toutes les décisions et actions de l'opérateur tel que, le nombre de requêtes, le temps consacré à chaque objectif d'ordonnancement et chaque critère concernant chaque décision.

### 7.2.2 Résultats

#### Performance

La performance du modèle IHM a été mesurée par le biais de deux variables. La première variable représente la *performance globale*, qui désigne la performance de l'ordonnancement réalisé en ce qui concerne les deux objectifs d'ordonnancement $C_{\max}$ et $T_{\max}$. La deuxième variable est la *performance locale* et désigne la qualité d'une décision. Cette variable est représentée par une valeur binaire indiquant si la décision prise par l'opérateur est optimale ou non concernant l'un des deux objectifs.

Par exemple, pour calculer la performance locale de la décision de l'opérateur (pour le $C_{\max}$) sur la décision représentée dan la figure 7.6, nous devons d'abord calculer l'ordonnancement optimal pouvant être obtenu si l'opération du job $J_{11}$ est exécutée en premier (dans ce cas $C_{\max} = 1571$), ainsi que l'ordonnancement optimal lorsque l'opération du job $J_{16}$ est exécutée en premier (1658) (ces deux valeurs sont calculées à l'aide de la procédure de séparation et d'évaluation développée dans la section 5.4). Dans ce cas, si l'opérateur choisit $J_{11}$, alors la performance locale concernant le $C_{\max}$ est égale à un, sinon cette performance locale est égale à zéro. Dans le reste de cette section, lorsque le terme *performance locale* n'est pas suivi par un des deux objectifs d'ordonnancement choisis (i.e, $C_{\max}$ ou $T_{\max}$), il se réfère à la *performance moyenne locale* calculée sur les deux objectifs (en terme de pourcentage).

La figure 7.9 présente le résultat de la performance locale obtenue par *GS_G1* et *GS_G2* concernant chaque décision. Ce résultat montre que le deuxième groupe était meilleur sur 66% des décisions [1]. Toutefois, lorsque les perturbations ont lieu, durant les décisions 5 et 11, *GS_G1* a légèrement dépassé *GS_G2*. Mais, cette amélioration est immédiatement détériorée dans les décisions suivantes.

L'amélioration de la performance locale est principalement due à la façon dont les perturbations sont représentées pour le deuxième groupe ; *GS_G2* a été en mesure d'avoir une vue plus claire sur l'impact des perturbations. Cela peut également être expliqué par le fait que ce groupe était plus actif que le premier groupe sur 75% des décisions, comme le montre le tableau 7.1. Dans ce tableau, est indiqué le nombre moyen de critères utilisés par groupe. Rappelons que le nombre total de critères est huit (3 pour le $C_{\max}$ et 5 pour le $T_{\max}$).

L'amélioration de l'activité de l'opérateur a permis d'améliorer la performance locale et cela devrait améliorer la performance globale de l'ordonnancement. Dans

1. Ce résultat est significatif en utilisant le test des variances avec 10% d'erreur.

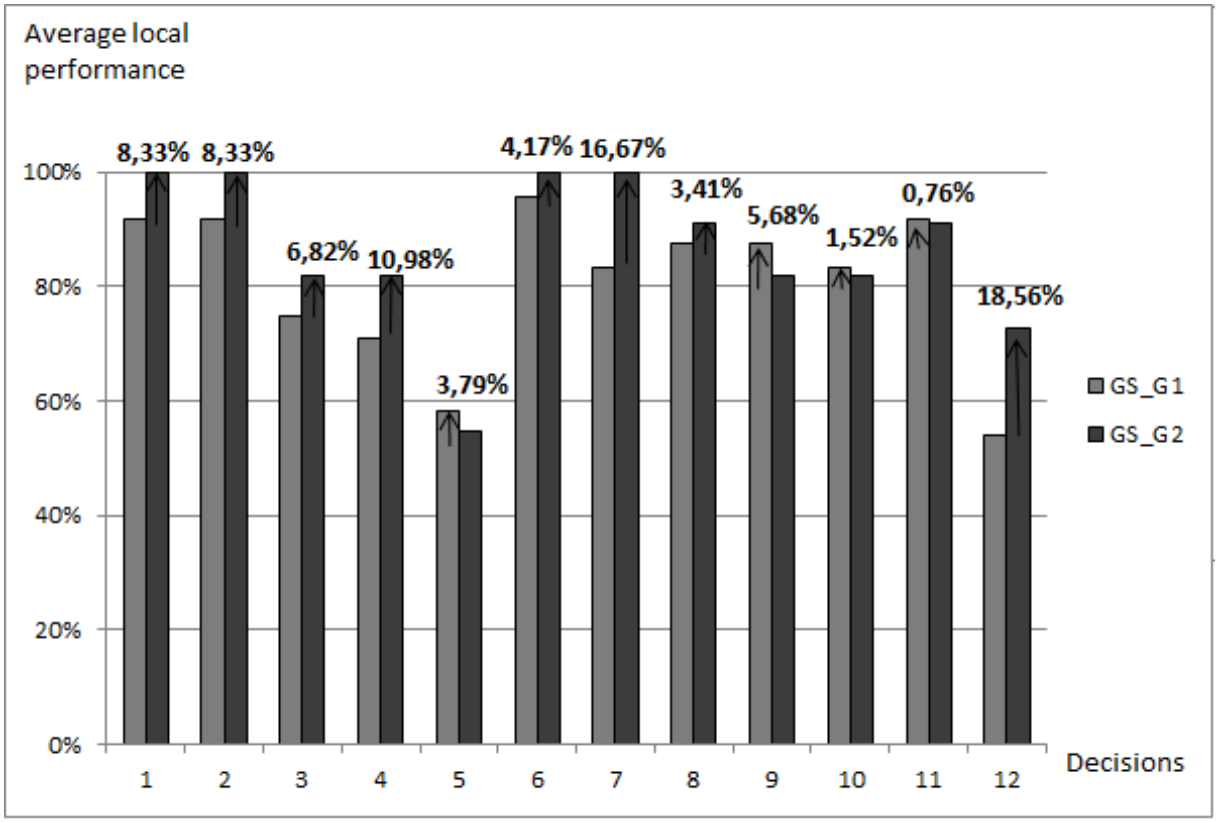



FIGURE 7.9 – *Comparaison entre la performance locale de GS_G1 et GS_G2*

TABLE 7.1 – *Le nombre moyen de critères utilisés pour chaque décision*

| Décision | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *GS_G1* | 5.50 | 5.13 | 5.29 | 5.96 | 6.25 | 5.21 | 5.38 | 5.63 | 4.96 | 5.50 | 4.96 | 4.46 |
| *GS_G2* | 6.18 | 6.73 | 5.82 | 6.00 | 6.45 | 5.18 | 5.18 | 5.45 | 5.45 | 5.63 | 5.27 | 4.72 |

la figure 7.10, nous illustrons l'impact de la performance locale sur la performance globale de l'ordonnancement réalisé.

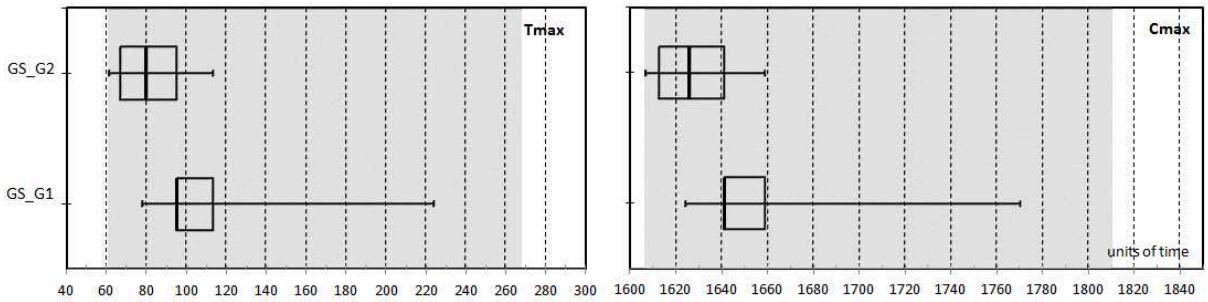



FIGURE 7.10 – *Comparaison entre la performance globale de GS_G1 et GS_G2*

Les zones en surbrillance représentent les performances tolérables qui peuvent être obtenues à partir de l'ordonnancement de référence présenté dans la figure 7.4 (sans perturbations). Cet ordonnancement caractérise plus de dix milles solutions avec une performance dans le meilleur des cas (resp. pire des cas) égale à 1607 (resp. 1810) pour le $C_{\max}$ et 61 (resp. 264) pour le $T_{\max}$.

Cependant, en comparant les performances obtenues par les deux groupes, il est très clair que le groupe *GS_G2* a obtenu les meilleurs résultats. On peut remarquer que la médiane de *GS_G2* est presque égale à la meilleure valeur obtenue par *GS_G1*. De plus, le premier quartile de *GS_G1* est le même que le troisième quartile de *GS_G2*.

Il est difficile de comparer ces résultats avec ceux de l'expérimentation précédente (Guérin 2013) à cause des différences dans les conditions des deux expérimentations. Cependant, il est à noter que tous les étudiants ont pu obtenir un ordonnancement réalisé acceptable (appartenant à la zone en surbrillance). L'étudiant qui a pris les pires décisions en utilisant l'IHM proposée a eu un ordonnancement réalisé avec un $C_{\max}$ de 1770 et un $T_{\max}$ de 224.

**Critères d'aide à la décision**

À la fin de l'expérimentation, nous avons demandé aux étudiants de répondre à un questionnaire dont certaines questions concernaient leur avis sur le(s) critère(s) qu'ils trouvaient le(s) plus pertinent(s). Pour 68% des étudiants ayant répondu au questionnaire, le critère d'aide à la décision le plus intéressant est le meilleur des cas. Pour le reste, 18%, 9% et 5% ont préféré respectivement : la marge libre séquentielle, la date d'échéance et la date de début au plus tôt.

De plus, le nombre de requêtes (nombre de clics) pour les critères d'aide à la décision a été analysé. Les résultats du nombre total de requêtes pour chaque objectif sont présentés dans le tableau 7.2.

Il ressort de ce tableau que les étudiants étaient plus actifs sur le $T_{\max}$ parce qu'ils avaient plus de critères à utiliser. Cependant, certains des critères proposés étaient moins utilisés comparés les uns aux autres.

| | $C_{max}$ | $T_{max}$ | $C_{max}(\%)$ | $T_{max}(\%)$ |
|---|---|---|---|---|
| Meilleur des cas | 607 | 428 | 44.27% | 22.7% |
| Pire des cas | 398 | 306 | 29.03% | 16.18% |
| Marge libre séquentielle | - | 421 | - | 22.33% |
| Date de début au plus tôt | 366 | 316 | 26.7% | 16.77% |
| Date d'échéance | - | 415 | - | 22.02 |
| Somme | 1371 | 1885 | 100% | 100% |

TABLE 7.2 – *Le nombre total de requêtes pour chaque critère et pour chaque objectif.*

La figue. 7.11 représente en boite à moustache le nombre de requêtes pour les trois critères concernant le $C_{max}$. La proportion moyenne du nombre de requêtes/par-chaque-décision pour le critère du meilleur des cas a significativement surpassé les deux autres critères avec un niveau d'erreur de 1% (test de variances).

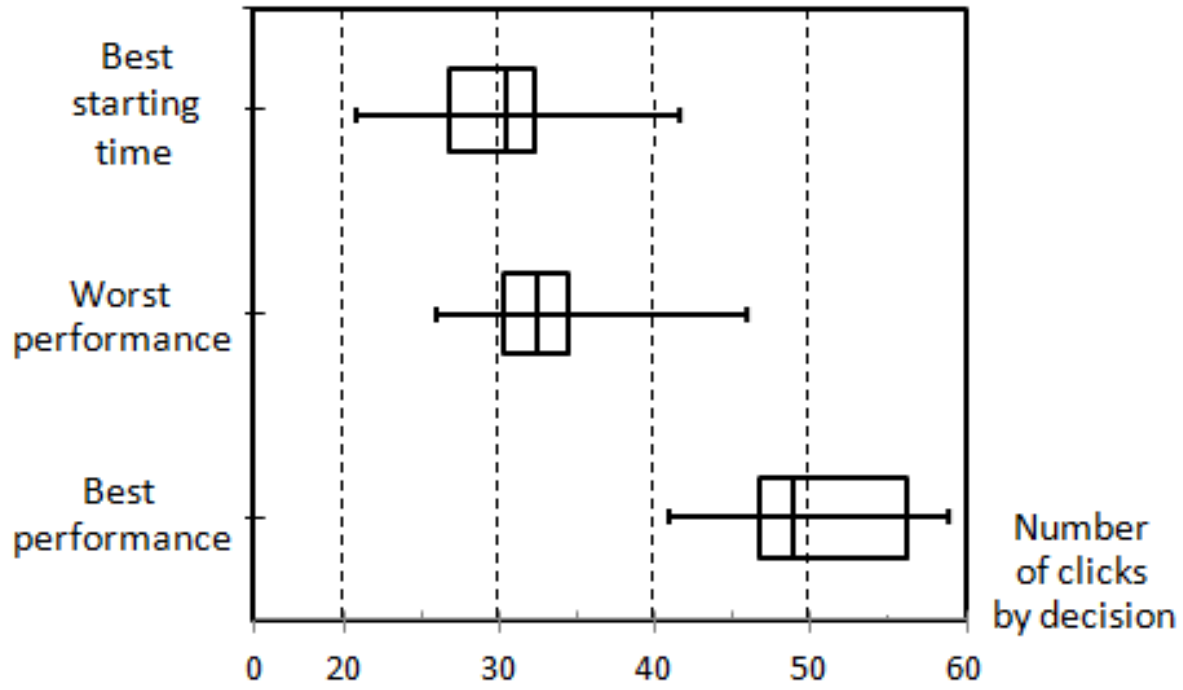



FIGURE 7.11 – *Nombre de clics sur chaque critère pour le* $C_{max}$

Ceci peut être expliqué par le fait que le meilleur des cas a une meilleure anticipation sur le $C_{max}$, alors que les autres critères ne peuvent être utilisés qu'en cas d'égalité sur les valeurs obtenues par le meilleur des cas.

Le nombre de requêtes pour les cinq critères concernant le $T_{max}$ est représenté également sous la forme de boîtes à moustache dans la figure 7.12.

De cette figure, on peut conclure que les deux critères : pire des cas et date de début au plus tôt sont moins utilisés. Cependant, le critère du pire des cas est significativement moins utilisé que le meilleur des cas, la marge libre séquentielle et la date d'échéance. Ceci n'est pas totalement surprenant, car ce critère ne peut pas garan-

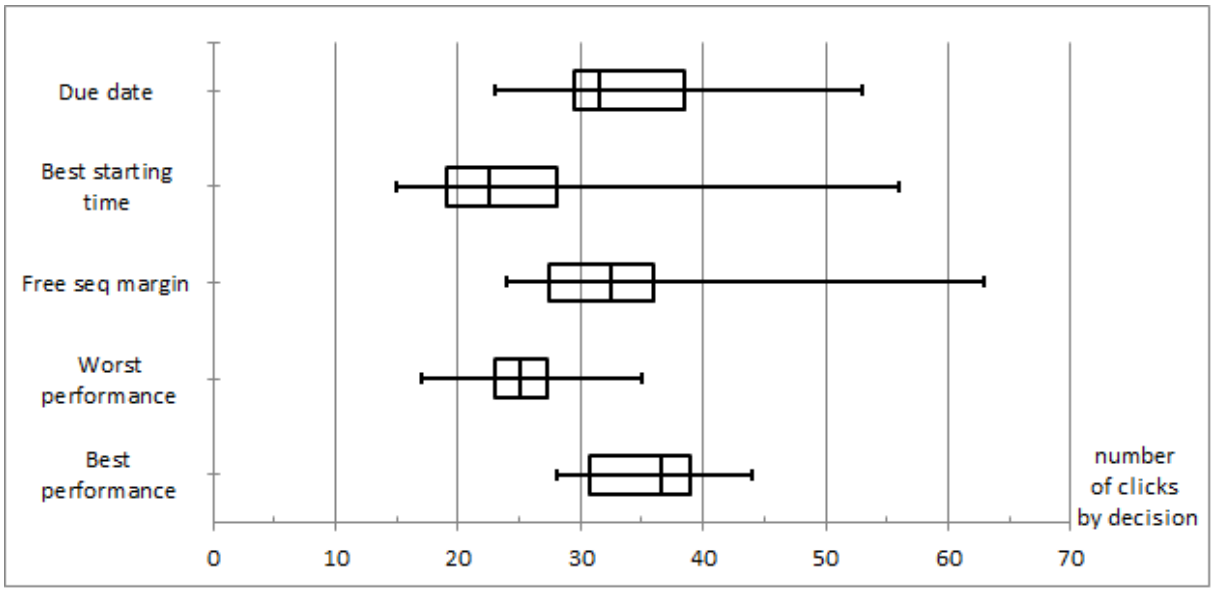



FIGURE 7.12 – *Nombre de clics sur chaque critère pour le* $T_{\max}$

tir une meilleure anticipation sur la performance de l'ordonnancement réalisé, mais permet d'empêcher l'opérateur de prendre les pires décisions.

Étonnamment, le critère de la date d'échéance a montré son utilité pour le $T_{\max}$. Ce critère avait à peu près les mêmes résultats que ceux du meilleur des cas et la marge libre séquentielle. Les étudiants ont passé près de 40% du temps total de leurs décisions sur ce critère. Cela peut être expliqué par le fait que ce critère ne donne pas d'information directe sur la performance attendue de l'ordonnancement réalisé ce qui oblige l'opérateur à calculer explicitement les retards éventuels. Cela a été confirmé par le feedback sur le questionnaire proposé.

### 7.2.3 Discussion

Le résultat présenté ci-dessus illustre les critères les plus utiles qui devraient être utilisés dans un système d'aide à la décision pour chacun des deux objectifs choisis. Les opérateurs ont pu atteindre au moins 75% de la meilleure performance atteignable sur presque toutes les décisions et cela a permis d'améliorer la performance globale de l'ordonnancement. Cependant, la représentation des perturbations sur le système HMI a permis d'améliorer les décisions de l'opérateur et donc la performance d'ordonnancement. Dans l'ensemble, nous pouvons caractériser le processus décisionnel en deux phases :

— La phase stable, qui concerne les quatre premières décisions avant que la première perturbation a eu lieu.
— La phase de perturbation (instable) : cette phase comprend toutes les décisions de la décision 5 à la décision 12 où les perturbations se sont produites.

Nous avons mesuré la performance globale dans chacune de ces deux phases. Nous avons remarqué que l'intervalle moyen entre l'ordonnancement réalisé qui

pourra être atteint et celui obtenu par les étudiants est compris entre zéro et 26 unités pour le $C_{\max}$ et 31 pour le $T_{\max}$.

Il a été remarqué qu'à la fin de la phase de décision stable, plus de 50% des étudiants n'étaient pas capables de prendre des décisions optimales, même si ces décisions étaient assez simples parce qu'aucune perturbation ne s'est produite. Ceci peut être expliqué par le fait que les opérateurs ont mal utilisé certains critères proposés ou ont utilisé plus de critères que nécessaires. Cependant, les opérateurs qui ont pris des décisions optimales au cours de cette phase ont donné la priorité principalement au critère du meilleur des cas.

Dans ce contexte, deux suggestions peuvent être proposées. La première est de limiter le nombre de critères de décision en fonction de chaque phase de décision, de telle sorte que l'opérateur sera en mesure de capturer la décision optimale, en particulier pendant la phase stable de l'atelier. Cette suggestion peut ouvrir la question sur les critères qui doivent être utilisés dans chaque phase de décision.

La deuxième suggestion est d'alouer le contrôle de décision à la machine durant la phase stable. Ainsi, l'opérateur pourra laisser la main à la machine s'il juge que l'atelier est stable. Dans ce cas, nous recommandons que la machine prenne les décisions selon les deux algorithmes $RA\_best - case$ ou $RA\_best - case_{FSM}$ proposés dans la section 6.3.

# Conclusion générale et perspectives

Dans cette thèse, nous nous sommes intéressés au problème d'ordonnancement d'atelier de production en présence d'incertitudes. L'objectif principal est de favoriser la coopération homme-machine durant l'exécution d'un ordonnancement à l'aide d'un système d'aide à la décision.

Plusieurs méthodes existent dans la littérature. Les plus performantes consistent à introduire de la flexibilité dans l'ordonnancement durant la phase de planification hors-ligne, pour ensuite adapter l'ordonnancement à l'état réel de l'atelier. Nous avons choisi la méthode d'ordonnancement de groupes comme modèle théorique et pratique de base pour répondre à cette problématique. Cette méthode (en plus de favoriser la coopération homme-machine) est une des approches les plus étudiées dans la littérature consacrée à l'ordonnancement sous incertitudes. Elle consiste à proposer une solution d'ordonnancement flexible caractérisant un ensemble fini non énuméré d'ordonnancements admissibles. Un opérateur doit sélectionner l'ordonnancement qui répond le mieux aux perturbations survenues dans l'atelier.

L'idée de la méthode de résolution proposée est basée sur le critère du meilleur des cas représentant l'ordonnancement semi-actif optimal de l'ordonnancement de groupes. Ce critère pourra être utilisé dans un système d'aide à la décision pour aider l'opérateur à choisir le meilleur ordonnancement en fonction de l'état réel de l'atelier. Dans ce cadre, deux thématiques de recherche ont été abordées et quatre contributions majeures ont été proposées.

**La première thématique** est centrée sur la résolution exacte et approchée du calcul du meilleur des cas :

— Dans un premier lieu, nous avons proposé trois bornes inférieures pour calculer le meilleur des cas dans un atelier à cheminement multiple. Ces bornes sont inspirées de la borne classique du job shop qui a été adaptée à l'ordonnancement de groupes ; puisque le meilleur des cas est un problème partiel du job shop. Cependant, le calcul d'une borne inférieure pour le meilleur des cas consiste à calculer les dates de début/fin des jobs/groupes dans l'ordonnancement de groupes ; un job/groupe ne pourra commencer son exécution qu'après la date de fin de ses groupes/jobs prédécesseurs. Une fois que ces dates sont estimées, la relaxation du problème en $m$ machines est adaptée à l'ordonnancement de groupes. Pour l'objectif du $C_{max}$ par exemple, une simple borne inférieure est représentée par le maximum des $C_{max}$ pour les $m$ sous

problèmes. L'expérimentation réalisée sur des instances de benchmark a montré que le calcul de cette borne est très rapide et a permis de résoudre d'une manière optimale plus de 50% des instances en moins d'une seconde.
Ensuite, deux améliorations ont été proposées à cette borne. La première amélioration (deuxième borne proposée) consiste à ajuster les dates de début/fin des jobs/groupes. Une règle de précédence entre les groupes voisins a été définie et testée sur les instances de Lawrence. Les résultats ont montré que cette amélioration est significative mais plus coûteuse en temps sur certaines instances. Cependant, la plupart des instances ont été résolues en moins d'une seconde. La troisième borne proposée, inspirée des travaux de Pinot (2008), consiste à relaxer le problème en différents sous problèmes où chaque groupe représente un problème à une machine. Cependant, les résultats de comparaisons ont montré que cette borne n'a pas pu améliorer les résultats de la deuxième borne. De plus, elle nécessite deux fois le temps de calcul de la deuxième borne. Au final, la deuxième borne semble la plus efficace en termes de compromis performance/temps de calcul. Pour cette raison, nous avons pris la décision de l'utiliser pour la suite.

— Malgré la précision des bornes proposées, plus de 40% des instances n'ont pas pu être résolu d'une manière optimale. La raison est principalement liée à la complexité du problème pour les instances de grandes tailles. Pour résoudre ce problème, nous avons implémenté une méthode exacte de séparation et d'évaluation. Dans ce contexte, la deuxième contribution de la thèse consiste à proposer trois méthodes de séparation. Les deux premières méthodes génèrent un ordre de traitement des groupes. L'objectif est de commencer le parcours de l'arbre de recherche par les nœuds avec les moins de relations possibles. Ceci permettra de réduire l'espace de recherche et par conséquent de confirmer rapidement la solution optimale une fois qu'elle est trouvée. Les expérimentations réalisées ont montré que la troisième méthode de séparation proposée est la plus performante. Cette méthode permet d'éviter la séparation sur les groupes n'appartenant pas au chemin critique. Les expérimentations ont confirmé l'utilité de la méthode exacte en temps réel dans un système d'aide à la décision, particulièrement pour les instances de petites et moyennes tailles.

**La deuxième thématique** de la thèse liée à la coopération homme-machine, traite l'utilité du meilleur des cas dans un système d'aide à la décision. Dans ce contexte, deux contributions sont proposées :

— La première s'est intéressée à l'implémentation du critère du meilleur des cas dans un système de coopération homme-machine. L'état de l'art sur les applications de la coopération homme-machine en ordonnancement montre qu'il existe un manque d'interface homme-machine ergonomique pour l'ordonnancement d'ateliers de productions. Notre proposition rentre dans ce cadre et consiste en une plate-forme d'IHM adaptée à l'ordonnancement de groupes. Cependant, avant de mettre en place cette plate-forme, nous avons d'abord étu-

dié l'intérêt du meilleur des cas dans un système d'aide à la décision à l'aide d'un algorithme de simulation. Le principe de l'algorithme proposé est d'exécuter dans un groupe d'opérations permutables l'opération avec la meilleure borne inférieure. Deux autres algorithmes de simulation de prise de décision (pour le pire des cas et la marge libre séquentielle) ont également été comparés. Contrairement au résultat présenté dans la thèse de Guérin (2013), le résultat du meilleur des cas était nettement meilleur que celui des algorithmes de décision utilisant la marge libre séquentielle. Les résultats de simulation ont également montré qu'un système d'aide à la décision multicritères est avantageux dans le contexte de l'ordonnancement de groupes, surtout dans le cas où les bornes inférieures ne sont pas très précises.

— Ces résultats encourageants nous ont permis de franchir le pas vers une expérimentation d'un cas d'étude avec l'intervention de l'humain dans le processus de prise de décision dans le modèle IHM proposé. Ceci représente notre dernière contribution. En se basant sur les conditions d'expérimentation de Guérin (2013), nous avons mis en place un nouveau plan expérimental qui nous a permis de réaliser une expérimentation avec des opérateurs humains. Cependant, ce nouveau plan expérimental soulève également de nouvelles questions telles que l'impact de la représentation des perturbations sur les décisions et la relation entre l'objectif de l'ordonnancement et les critères d'aide à la décision.
  La nouvelle expérimentation menée avec trente-cinq étudiants a permis de répondre à ces questions et d'analyser les points forts et les points faibles de l'IHM proposée :
  — Le résultat majeur de cette expérimentation a permis de valider l'intérêt du système multicritères dans un système de coopération homme-machine, en augmentant la part d'activité de l'humain et par conséquent la qualité de ses décisions.
  — Le meilleur des cas est le critère le plus utilisé pour le $C_{max}$ .
  — Pour les objectifs liés au retard, le meilleur des cas est plus performant combiné avec la marge libre séquentielle et la date d'échéance.
  — La représentation de la perturbation et la façon dont cette dernière est traitée sont des facteurs très importants dans une interface homme-machine.
  — Malgré, tous les moyens mis en place pour aider l'ordonnanceur à prendre des bonnes décisions notamment avec l'indicateur du meilleur des cas, l'opérateur n'a pas pu prendre des décisions optimales mêmes en absence de perturbations. Ceci est peut être du au choix des critères d'aide à la décision proposés pour chacun des deux objectifs d'ordonnancement.

Par ailleurs, ces derniers résultats ont permis d'ouvrir de nouvelles perspectives pour les études futures sur le plan théorique et pratique des deux thématiques traitées dans cette thèse.

Une première perspective consiste à calculer le meilleur des cas pour d'autres objectifs d'ordonnancement. Les bornes inférieures développées dans cette thèse per-

mettent de contribuer à la réponse à ce point. En pratique, différents objectifs sont à optimiser dans un problème d'ordonnancement d'atelier. Dans ce cas, une des pistes possibles serait de calculer le compromis entre le meilleur des cas de chaque objectif.

Une deuxième perspective intéressante serait de développer des heuristiques/méta-heuristiques calculant le meilleur des cas, surtout pour les objectifs non-réguliers où lorsque les bornes inférieures ne peuvent pas être calculées en temps polynomial.

La troisième et dernière perspective concerne les limites du modèle IHM proposé. Dans ce contexte, il faut développer en premier lieu des interfaces ergonomiques permettant de mieux représenter les perturbations et leurs conséquences.

De plus, le système de contrôle des décisions utilisé dans l'expérimentation exposée dans la dernière partie de cette thèse pourrait être amélioré. Il serait intéressant de proposer un système de contrôle permettant de déléguer les décisions entre l'humain et la machine en prenant en compte l'état réel de l'atelier.

# Bibliographie


Aloulou, M., Kovalyov, M., et Portmann, M.C. Maximization in single machine scheduling. *Annals of Operations Research*, 129(1-4), p. 21–32, 2004. 44

Aloulou, M. A. et Artigues, C. Flexible solutions in disjunctive scheduling : general formulation and study of the flow-shop case. *Computers and Operations Research*, 37 (5), p. 890–898, 2010. 38

Ammons, J.C., Govindaraj, T., et Mitchell, C.M. Human aided scheduling for fms : a supervisory control paradigm for real-time control of flexible manufacturing systems. *Annals of Operations Research*, 15(1-4), p. 313–335, 1986. 80

Artigues, C. *Ordonnancement en temps réel d'ateliers avec temps de préparation des ressources*. PhD thesis, LAAS - Laboratoire d'analyse et d'architecture des systèmes (Toulouse), 1997. 38, 46

Artigues, C. Optimisation et robustesse en ordonnancement sous contraintes de ressources, 2004. URL `http://www.laas.fr/~artigues/hdrChristianArtigues.pdf`. Habilitation à diriger des recherches, Université d'Avignon. 7, 22, 35

Artigues, C., Billaut, J.C., et Esswein, C. Maximization of solution flexibility for robust shop scheduling. *European Journal of Operational Research*, 165(2), p. 314–328, 2005. 38, 44, 45, 54

Artigues, Christian, Billaut, Jean-Charles, Cherif, Bachir, Mebarki, Nasser, et Yahouni, Zakaria. Robust machine scheduling based on group of permutable jobs. Dans *Robustness Analysis in Decision Aiding, Optimization, and Analytics, International Series in Operations Research and Management Science*, Doumpos, M., Zopounidis, C., et Grigoroudis, E. (éditeurs). Springer, 2016. 22, 35, 39, 54

Baptiste, P., Flamini, M., et F., Sourd. Lagrangian bound for just-in-time job-shop scheduling. *Computers & Operations Research*, 35(3), p. 906–915, 2008. 53

Beasley, J. Job shop benchmark instances, 2004. URL `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/jobshop1.txt`. 87

Beck, J. C. et Wilson, N. Proactive algorithms for job shop scheduling with probabilistic durations. *journal of artificial intelligence research*, 28, p. 183–232, 2007. 28

Berglunda, M. et Karltuna, J. Human, technological and organizational aspects influencing the production scheduling process. *International Journal of Production Economics*, 110(1-2), p. 160–174, 2007. v, 80, 81

Bidot, Julien, Vidal, Thierry, Laborie, Philippe, et Beck, J. Christopher. A theoretic and practical framework for scheduling in a stochastic environment. *Journal of Scheduling*, 12(3), p. 315–344, 2008. 25, 32

Billaut, J-C. *Prise en compte des ressources multiples et des temps de préparation dans les problèmes d'ordonnancement en temps réel. Thèse de doctorat*. PhD thesis, Université Paul Sabatier, 1993. 38, 46

Billaut, J.C., Moukrim, A., et Sanlaville, E. *Flexibility and Robustness in Scheduling*. Wiley-ISTE, December 2008. 22, 23, 25, 35, 86, 101

Blazewicz, J. *Scheduling in Computer and Manufacturing Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996. 11

Brucker, P., Burke, E. K., et Groenemeyer, S. A branch and bound algorithm for the cyclic job-shop problem with transportation. *Computers & Operations Research*, 39 (12), p. 3200–3214, 2012. 54

Brucker, P., Drexl, A., Möhring, R., Neumann, K., et Pesch, E. Resource-constrained project scheduling : Notation, classification, models, and methods. *European journal of operational research*, 112(1), p. 3–41, 1999a. 28

Brucker, P., Jurisch, B., et Sievers, B. A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics*, 49, p. 107–127, 1994a. 16

Brucker, P. et Knust, S. Complexity results for scheduling problems, 2007. URL `http://www.mathematik.uni-osnabrueck.de/research/OR/class/`. 55

Brucker, P., Kravchenko, S.A., et Sotskov, Y.N. Preemptive job-shop scheduling problems with a fixed number of jobs. *Mathematical Methods of Operations Research*, 49 (1), p. 41–76, 1999b. 17

Brucker, Peter et Jurisch, Bernd. A new lower bound for the job-shop scheduling problem. *European Journal of Operational Research*, 64(2), p. 156–167, 1993. 53, 69

Brucker, Peter, Jurisch, Bernd, et Kramer, Andreas. The job-shop problem and immediate selection. *Annals of Operations Research*, 50, p. 73–114, 1994b. 53

Brusco, M.J. et Stahl, S. *Branch-and-Bound Applications in Combinatorial Data Analysis*, volume 221 de *Statistics and Computing*. Springer New York, xii édition, 2005. 18

Bui, Tung et Lee, Jintae. An agent-based framework for building decision support systems. *Decision Support Systems*, 25(3), p. 225–237, 1999. 80

Cardin, Olivier, Mebarki, Nasser, et Pinot, Guillaume. A study of the robustness of the group scheduling method using an emulation of a complex fms. *International Journal of Production Economics*, 146(1), p. 199–207, 2013. 38

Carlier, J. The one machine problem. *European Journal of Operational Research*, 11(1), p. 42–47, 1982. 52, 59, 60

Carlier, J. et Pinson, E. An algorithm for solving the job-shop problem. *Management Science*, 35(2), p. 164–176, 1989. 16, 52, 53

Carlier, J. et Pinson, E. A practical use of jackson's preemptive schedule for solving the job shop problem. *Annal of Operation Research*, 26(1-4), p. 269–287, 1991. 52

Carlier, J. et Pinson, E. Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research*, 78(2), p. 146–161, 1994. 53

Chaari, Tarek. *Un algorithme génétique pour l'ordonnancement robuste : application au problème du flow shop hybride*. PhD thesis, Université de valencienne, 2010. 22

Chan, F.T.S, Chan, H.K., et Lau, H.C.W. The state of the art in simulation study on fms scheduling : a comprehensive survey. *International Journal of Advanced Manufacturing Technology*, 19(11), p. 830–849, 2002. 29

Crawford, S., MacCarthy, B.L., Wilson, J.R., et Vernon, C. Investigating the work of industrial schedulers through field study. *Cognition, Technology and Work*, 1(2), p. 63–77, 1999. 80

Crawford, S. et Wiers, V.C.S. From anecdotes to theory : a review of existing knowledge on human factors of planning and scheduling. Dans *Human performance in planning and scheduling*, MacCarthy, B.L. et J.R., Wilson. (éditeurs), pages 15–43. Taylor and Francis, London, 2001. 80

Daniels, R. L. et Carrillo, J. E. Beta-robust scheduling for single-machine systems with uncertain processing times. *IIE Transactions*, 29(11), p. 977–985, 1997. 28, 34

Davenport, A.J. et Beck, J.C. A survey of techniques for scheduling with uncertainty, 2000. URL `http://tidel.mie.utoronto.ca/publications.php`. 23, 26

Debels, Dieter et Vanhoucke, Mario. Future research avenues for resource-constrained project scheduling : search space restriction or neighbourhood search extension ? Dans *Proceedings of the 10th International Workshop on Project Management and Scheduling*, pages 110–113, 2006. 31

Dewess, G. An existence theorem for packing problems with implications for the computation of optimal machine schedules. *Optimization*, 25(2-3), p. 261–269, 1992. 53

Dimopoulos, Christos, Cegarra, Julien, Papageorgiou, George, Gavriel, George, et Chouchourelou, A. Interdisciplinary design of scheduling decision support systems in small-sized sme environments : The i-desme framework. *Journal of Scheduling*, 24 (3), p. 227–254, 2015. 82

Drummond, Mark, Bresina, John, et Swanson, Keith. Just-in-case scheduling. Dans *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 2)*, AAAI'94, pages 1098–1104, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence. 29

Erschler, Roubellat. An approach for real time scheduling for activities with time and resource constraints, 1989. In Slowinski, R. and Weglarz, J., editors, Advances in project scheduling. Elsevier. 38

Esswein, Carl. *Un apport de flexibilité séquentielle pour l'ordonnancement robuste*. PhD thesis, Tours University, Tours, 2003. v, 8, 10, 22, 23, 24, 38, 44, 45, 46, 47, 49, 69

Esswein, Carl, Billaut, Jean Charles, et Strusevich, V. A. some heuristics for incorporating flexibility into job shop solutions, 2003. HEURO/INFORMS Joint International Meeting, Istanbul (Turkey). 46

Fang, C., Kolisch, R., Wang, L., et Mu, C. An estimation of distribution algorithm and new computational results for the stochastic resource-constrained project scheduling problem. *Flexible Services and Manufacturing Journal*, 27(4), p. 585–605, 2015. ISSN 1936-6582. 27

Fisher, H. et Thomson, G.L. Probabilistic learning combinations of local job-shop scheduling rules. *Industrial scheduling*, pages 225–251, 1963. 16

Fox, M.S. Constraint-guided scheduling - a short history of research at cmu. *Computers in Industry*, 14, p. 79–88, 1990. 80

Fu, Na, Varakantham, Pradeep, et Lau, Hoong Chuin. Towards finding robust execution strategies for rcpsp/max with durational uncertainty. Dans *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010*, pages 73–80, 2010. 34

Gacias, Bernat, Cegarra, Julien, et Lopez, Pierre. Scheduler-oriented algorithms to improve human-machine cooperation in transportation scheduling support systems. *Engineering Applications of Artificial Intelligence*, 25(4), p. 801–813, 2012. 82

Gantt, H. L. *Work, wages, and profits*. New York : The Engineering magazine co., 1919. 14

Garey, Michael R. et Johnson, David S. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. 17

Garey, M.R., Johnson, D.S., et Sethi, R. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2), p. 117–129, 1976. 17

Gasser, Roland, Fischer, Katrin, et Wäfler, Toni. Decision making in planning and scheduling : A field study of planning behaviour in manufacturing. Dans *Behavioral Operations in Planning and Scheduling*, Fransoo, C. Jan, Waefler, Toni, et Wilson, R. John (éditeurs), pages 11–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. 80

Gigerenzer, Gerd, Todd, P.M., et Group, ABC. *Simple Heuristics that Make us Smart*. New York : Oxford University Press, 1999. 79

Gilovich, T., Griffin, D., et Kahneman, D. *Heuristics and Biases : The Psychology of Intuitive Judgment*. Cambridge University Press, Cambrighe, 2002. 81

Goldratt, E. Critical chain. north river : Great barringto, 1997. 27

Gotha. Les problèmes d'ordonnancement. *RAIRO - Operations Research*, 27(1), p. 77–150, 1993. 7, 8

Gotha. Flexibilité et robustesse en ordonnancement, 2002. bulletin de la ROADEF. 24, 25

Graham, R.L. et Lawler, E.L. Optimization and approximation in deterministic sequencing and scheduling : a survey. *Annals of Discrete Math.*, 5, p. 287–326, 1979. 11, 12

Guérin, Clément. *Gestion de contraintes et expertise dans les stratégies d'ordonnancement*. PhD thesis, Université de Nantes, 2013. 95, 105, 111

Hardman, D. et Macchi, L. *Thinking : Psychological Perspectives on Reasoning, Judgment and Decision Making*. Chichester : Wiley, 2003. 81

Herroelen, W. et Leus, R. Project scheduling under uncertainty : Survey and research potentials. *European Journal of Operational Research*, 165(2), p. 289–306, 2005. 26, 28, 29

Hoc, J.M., Guerin, C., et Mebarki, N. The nature of expertise in scheduling : The case of timetabling. *Human Factors and Ergonomics in Manufacturing & Service Industries*, 24(2), p. 192–206, 2012. 82

Hoc, J.M., Mebarki, N., et Cegarra, J. L'assistance à l'opérateur humain pour l'ordonnancement dans les ateliers manufacturiers. *Le travail humain*, 64(2), p. 181–208, 2004. 84

Jackson, J.R. Scheduling a production line to minimize maximum tardiness, 1955. Research Report 43, Management Science Research Project, University of California, Los Angeles, CA. 51

Jackson, S., Wilson, J.R., et MacCarthy, B.L. A new model of scheduling in manufacturing : tasks, roles, and monitoring. *Human Factors*, 46(3), p. 533–550, 2004. 80

Jain, Anant Singh et Meeran, Sheik. A state of the art review of job-shop scheduling techniques, 1998. v, 18

Kouvelis, P. et Yu, G. *Robust Discrete Optimization and Its Applications*. Springer, US, 1997. 23

Kravchenko, S.A. Minimizing the number of late jobs for the two-machine unit-time job-shop scheduling problem. *Discrete Applied Mathematics*, 98(3), p. 209–217, 1999. 17

Kubiak, W. et Timkovsky, V.G. A polynomial-time algorithm for total completion time minimization in two-machine job-shop with unit-time operations. *European Journal of Operation Research*, 94, p. 310–320, 1996. 17

La, Hoang Trung. *Utilisation d'ordres partiels pour la caractérisation de solutions robustes en ordonnancement*. PhD thesis, Institut Polytechnique de Hanoi, 2005. 22, 25, 35

Lamas, Patricio et Demeulemeester, Erik. A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. *Journal of Scheduling*, 19(4), p. 409–428, 2015. 28

Lambrechts, Olivier, Demeulemeester, Erik, et Herroelen, Willy. Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. *Journal of Scheduling*, 11(2), p. 121–136, 2008. 32

Lancia, G., Rinaldi, F., et P., Serafini. A time indexed lp-based approach for min-sum job-shop problems. *Annals of Operations Research*, 186(1), p. 175–198, 2011. 53

Lawler, E. L. Optimal sequencing of a single machine subject to precedence constraints. *Management Science*, 19(5), p. 544–546, 1973. 17, 55

Lawrence, S. Resource constrained project scheduling : an experimental investigation of heuristic scheduling techniques (supplement), 1984. 59

Le Gall, A. *Un système intéractif d'aide à la décision pour l'ordonnancement et le pilotage en temps réel d'atelier*. PhD thesis, Université Paul Sabatier (France), 1989. 38

Le Pape, C. *Constraint propagation in Planning and Scheduling*. PhD thesis, Robotics Laboratory, Department of Computer, stanford, 1991. 24

Lenstra, J.K. et Rinnooy Kan, A.H.G. Computational complexity of discrete optimization problems. *Annals of Discrete Mathematics*, 4, p. 121–140, 1979. 17

Lenstra, J.K., Rinnooy Kan, A.H.G., et Brucker, P. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1, p. 343–362, 1977. 17

Leon, V. Jorge, Wub, S. David, et Storer, Robert H. Robustness measures and robust scheduling for job shops. *IIE Transactions*, 26(5), p. 32–43, 1994. 24, 27

Lobo, Benjamin J., Hodgson, Thom J., King, Russell E., Thoney, Kristin A., et Wilson, James R. An effective lower bound on l max in a worker-constrained job shop. *Comput. Oper. Res.*, 40(1), p. 328–343, 2013. 53, 54

Lopez, P. et Roubellat, F. *Production scheduling*. John Wiley & Sons, London ; Hoboken, NJ, December 2008. 44, 86

Mebarki, N. L'ordonnancement d'atelier sous incertitudes : l'intégration du facteur humain., 2012. Habilitation à diriger des recherches, Université de Nantes. 8, 22, 23, 82

Mebarki, N., Cardin, O., et Guérin, C. Evaluation of a new human-machine decision support system for group scheduling. Dans *12th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems, IFAC HMS 2013, Las Vegas, Nevada, USA, August 11-15, 2013.*, Narayanan, Sundaram (éditeur), pages 211–217. International Federation of Automatic Control, 2013. 95

Monfared, M.A.S. et Yang, J.B. Design of integrated manufacturing planning, scheduling and control systems : a new framework for automation. *The International Journal of Advanced Manufacturing Technology*, 33(5), p. 545–559, 2007. 39, 80

Morikawa, Katsumi et Takahashi, Katsuhiko. A flexible branch and bound method for the job shop scheduling problem. *Industrial Engineering & management systems*, 8 (4), p. 239–246, 2009. 52, 69

Muise, Christian, Beck, J. Christopher, et McIlraith, Sheila A. Flexible execution of partial order plans with temporal constraints. Dans *International Joint Conference On Artificial Intelligence*, pages 2328–2335, 2013. 34

Muth, J.F. et Thompson., G.L. *Industrial Scheduling*. Prentice-Hall, Englewood Cliffs, 1963. 52, 53

Ourari, Samia. *De l'ordonnancement déterministe à l'ordonnancement distribué sous incertitudes*. PhD thesis, Université Paul Sabatier, 2011. 22, 23

Pinedo, Michael L. *Scheduling : Theory, Algorithms, and Systems*. Springer-Verlag New York, 4rd édition, 2012. 8

Pinot, G. *Coopération homme-machine pour l'ordonnancement sous incertitudes*. PhD thesis, Université de Nantes (France), November 2008. 2, 22, 23, 38, 43, 44, 46, 49, 54, 55, 58, 60, 64, 69, 110

Plessner, H., Betsch, C., et Betsch, T. *Intuition in Judgement and Decision Making*. Taylor and Francis, 2008. 81

Policella, Nicola, Oddi, Angelo, Smith, Stephen F., et Cesta, Amedeo. Generating robust partial order schedules. Dans *Principles and Practice of Constraint Programming – CP 2004 : 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004. Proceedings*, Wallace, Mark (éditeur), pages 496–511. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. 33, 34

Roubellat, F., Billaut, J.C., et Villaumié, M. Ordonnancement d'atelier en temps réel : d'orabaid à ordo. revue d'automatique et de productique appliquées. *Revue d'automatique et de productique appliquées*, 8(5), p. 683–713, 1995. 86

Roy, B. et Sussmann, B. Les problèmes d'ordonnancement avec contraintes disjonctives, 1964. SEMA, Research report. 13

Sabuncuoglu, I. et Goren, S. Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research. *International Journal of Computer Integrated Manufacturing*, 22(2), p. 138–157, 2009. 23, 25, 26, 30

Sadeh, E., Otsuka, S., et Schelback, R. Predictive and reactive scheduling with the microboss production scheduling and control system. *Workshop on knowledge-Based Production Planning, Scheduling and Control*, pages 293–306, 1993. 27

Sanderson, Penelope M. The human planning and scheduling role in advanced manufacturing systems : An emerging human factors domain. *Human Factors*, 31(6), p. 635–666, 1989. 79

Sanlaville, Eric. *Ordonnancement sous conditions changeantes : Comment prendre en compte les variations, aléas, incertitudes sur les données ?* PhD thesis, HDR, Université Blaise Pascal, 2005. 23

Schrage, L. Solving resource-constrained network problems by implicit enumeration : Non preemptive case. *Operations Research*, 18(2), p. 263–278, 1970. 52

Sevaux, M. et Sörensen, K. A genetic algorithm for robust schedules in a just-in-time environment. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2(2), p. 129–147, 2004. 25

Slowinski, R. et Weglarz, J. *Advances in project scheduling*. Elsevier, 1989. 86

Smith, S. Reactive scheduling systems. Dans *Intelligent Scheduling Systems*, volume 3, page 155–192. Brown, D. E. and W. T., Springer US, 1994. 27

Sotskov, Y.N. On the complexity of shop scheduling problems with two or three jobs. *European Journal of Operational Research*, 53(3), p. 323–336, 1991. 17

Sotskov, Y.N. et Shakhlevich, N.V. NP-hardness of shop-scheduling problems with three jobs. *Discrete Applied Mathematics*, 59(3), p. 237–266, 1995. 17

Taillard, E. Benchmarks for basic scheduling problems. *European journal of operational research*, 64(2), p. 278–285, 1993. 99

Tanaka, Shunji, Sadykov, Ruslan, et Detienne, Boris. A new lagrangian bound for the min-sum job-shop scheduling. Dans *Proceedings of the International Symposium on Scheduling ISS'2015*, Kobe, Japan, Jul 2015. 53

Thomas, V. *Aide à la décision pour l'ordonnancement d'atelier en temps réel*. PhD thesis, Université Paul Sabatier, 1980. 38, 43

Timkovsky, V.G. Is a unit-time job shop not easier than identical parallel machines ? *Discrete Applied Mathematics. Combinatorial Algorithms, Optimization and Computer Science*, 85(2), p. 149–162, 1998. 17

T'Kindt, Vincent et Billaut, Jean-Charles. *Multicriteria Scheduling : Theory, Models and Algorithms*. Springer-Verlag Berlin Heidelberg, 2006. 30

Trentesaux, D., Pesin, P., et Tahon, C. Distributed artificial intelligence for fms scheduling, control and design support. *Journal of Intelligent Manufacturing*, 11(6), p. 573–589, 2000. 80

Tversky, A. et Kahneman, D. Judgment under uncertainty : Heuristics and biases. *Science*, 185(4157), p. 1124–1131, 1974. 79

Van de Vonder, S. *Proactive/reactive procedures for robust project schedulin*. PhD thesis, Research Center for Operations Management. Katholieke Universiteit Leuven, Belgium, 2006. 27

Van de Vonder, Stijn, Demeulemeester, Erik, et Herroelen, Willy. A classification of predictive-reactive project scheduling procedures. *Journal of Scheduling*, 10(3), p. 195–207, 2007. 32

van de Vonder, Stijn Van, Ballestínb, Francisco, Demeulemeestera, Erik, et Herroelena, Willy. Heuristic procedures for reactive project scheduling. *Computers and Industrial Engineering*, 52(1), p. 11–28, 2007. 31

Wezel, Wout, Cegarra, Julien, et Hoc, Jean-Michel. Allocating functions to human and algorithm in scheduling. Dans *Behavioral Operations in Planning and Scheduling*, Fransoo, C. Jan, Waefler, Toni, et Wilson, R. John (éditeurs), pages 339–370. Springer Berlin Heidelberg, 2011. 81, 96

Wua, Christine Wei, Browna, Kenneth N., et Beckb, J. Christopher. Scheduling with uncertain durations : Modeling b-robust scheduling with constraints. *Computers and Operations Research*, 36(8), p. 2348–2356, 2009. 34

Yang, Jian et Yu, Gang. On the robust single machine scheduling problem. *Journal of Combinatorial Optimization*, 6(1), p. 17–33, 2002. 28

UNIVERSITE BRETAGNE LOIRE

CENTRALE NANTES

# Thèse de Doctorat

Zakaria YAHOUNI

## Titre de la thèse : Le meilleur des cas pour l'ordonnancement de groupes : un nouvel indicateur proactif-réactif pour l'ordonnancement sous incertitudes

## Title of thesis: The best-case for groups of permutable operations: a new proactive-reactive parameter for scheduling under uncertainties

**Résumé**

Cette thèse représente une étude d'un nouvel indicateur d'aide à la décision pour le problème d'ordonnancement d'ateliers de production sous présence d'incertitudes. Les contributions apportées dans ce travail se situent dans le contexte des groupes d'opérations permutables. Cette approche consiste à proposer une solution d'ordonnancement flexible caractérisant un ensemble fini non-énuméré d'ordonnancements. Un opérateur est ensuite censé sélectionner l'ordonnancement qui répond le mieux aux perturbations survenues dans l'atelier. Nous nous intéressons plus particulièrement à cette phase de sélection et nous mettons l'accent sur l'intérêt de l'humain pour la prise de décision. Dans un premier temps, nous présentons le meilleur des cas; indicateur d'aide à la décision pour le calcul du meilleur ordonnancement caractérisé par l'ordonnancement de groupes. Nous proposons des bornes inférieures pour le calcul des dates de début/fin des opérations. Ces bornes sont ensuite implémentées dans une méthode de séparation et d'évaluation permettant le calculer du meilleur des cas. Grâce à des simulations effectuées sur des instances de job shop de la littérature, nous mettons l'accent sur l'utilité et la performance d'un tel indicateur dans un système d'aide à la décision. Enfin, nous proposons une Interface Homme-Machine (IHM) adaptée à l'ordonnancement de groupes et pilotée par un système d'aide à la décision multicritères. L'implémentation de cette IHM sur un cas d'étude réel a permis de soulever certaines pratiques efficaces pour l'aide à la décision dans le contexte de l'ordonnancement sous incertitudes.

**Mots clés**
**Ordonnancement, ordonnancement de groupes, borne inférieure, séparation et évaluation, aide à la décision, coopération homme-machine.**

**Abstract**

This thesis represents a study of a new decision-aid criterion for manufacturing scheduling under uncertainties. The contributions made in this work relate to the groups of permutable operations context. This approach consists of proposing a flexible scheduling solution characterizing a non-enumerated and finite set of schedules. An operator is then supposed to select the appropriate schedule that best copes with the disturbances occurred on the shop floor.
We focus particularly on this selection phase and we emphasize the important of the human for decision-making. First, we present the best-case; a decision-aid criterion for computing the best schedule characterized by the groups of permutable operations method. We propose lower bounds for computing the best starting/completion time of operations. These lower bounds are then implemented in a branch and bound procedure in order to compute the best-case. Through to several simulations carried out on literature benchmark instances, we stress the usefulness of such criterion in a decision-aid system. Finally, we propose a Human-Machine-Interface (HMI) adapted to the groups of permutable operations and driven by a multi-criteria decision-aid system. The implementation results of this HMI on a real case study provided some insight about the practice of decision-making and scheduling under uncertainties.

**Key Words**
**Scheduling, Groups of permutable operations, Lower bound, Branch and Bound, Decision-aid System, Human-Machine Cooperation.**