Ordonnancement de tâches et de périodes d'indisponibilité de durée variable

Ahmed Gara-Ali

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : **Génie Industriel**

Arrêté ministériel : 7 août 2006

Présentée par

Ahmed GARA-ALI

Thèse dirigée par **Marie-Laure ESPINOUSE**

préparée au sein du **G-SCOP**
dans **l'École Doctorale I-MEP2**

Ordonnancement de tâches et de périodes d'indisponibilité de durée variable

Thèse soutenue publiquement le **19 juillet 2016**
devant le jury composé de :

M. Bernard PENZ
Professeur, Grenoble INP, Président
M. Vincent T'KINDT
Professeur, Ecole Polytechnique de l'Université de Tours, Rapporteur
M. Farouk YALAOUI
Professeur, Université de Technologie de Troyes, Rapporteur
M. Julien MONCEL
Maître de conférences, Université Toulouse 1 - IUT Rodez, Examinateur
M. Éric SANLAVILLE
Professeur, Université du Havre, Examinateur
Mme. Marie-Laure ESPINOUSE
Professeur, Université Grenoble Alpes, IUT 1, Directrice de thèse

Ahmed Gara-Ali : *Ordonnancement de tâches et de périodes d'indisponibilité de durée variable.* Thèse de doctorat, 19 juillet 2016

Ce travail est dédié à la mémoire de

Gerd Finke

qui a largement contribué à l'aboutissement de ce projet de thèse.

Gerd Finke était Professeur à l'université Joseph Fourier à Grenoble. C'était un chercheur éminent en Recherche Opérationnelle, domaine qu'il a animé à Grenoble pendant plusieurs années. Il a dirigé le master Grenoblois de Recherche Opérationnelle et a il a su initier des vocations avec de nombreux élèves faisant rayonner la Recherche Opérationnelle en France et dans le monde. Il avait une renommée internationale et des amis à travers le monde entier.

Nous garderons l'image d'un Professeur créatif, généreux, avec une ouverture d'esprit remarquable.

À ma famille,
À mes amis,
À mes rencontres de la vie, d'aujourd'hui et d'hier,
À tous ceux qui m'aiment,
À tous ceux que j'aime,
À la mémoire de mes amis, Ahmed Toumi et Taher Ben Amor.

Remerciements

Cette thèse est l'aboutissement de plusieurs années de travail et n'aurait jamais pu voir le jour sans le soutien et l'aide de nombreuses personnes qui m'ont accompagnée tout au long de cette épopée ! Je tiens donc ici à les remercier et leur témoigner de ma reconnaissance.

Je voudrais tout d'abord exprimer mes plus profonds remerciements à ma directrice de thèse, Marie-Laure Espinouse, pour toute sa disponibilité, sa gentillesse et ses directives si précieuses. Merci pour sa patience, ses nombreux conseils, ses idées, ses nombreuses relectures, et surtout pour m'avoir fait confiance dans cette aventure. Un grand merci pour cet encadrement de qualité intellectuellement et humainement.

Je tiens également à remercier chaleureusement Gerd Finke, pour ses idées et sa collaboration dans la réalisation de la première partie de la thèse.

Mes remerciements les plus sincères aux membres du jury : Bernard Penz, qui m'a fait l'honneur de présider le jury ; Vincent T'kindt et Farouk Yalaoui, qui ont accepté d'être rapporteurs de ce mémoire, merci pour leurs conseils et améliorations qu'ils ont suggérées, ainsi que Julien Moncel et Éric Sanlaville, d'avoir examiné ma thèse.

Je remercie aussi tous les enseignants et les chercheurs du laboratoire G-SCOP. Ces années à G-SCOP m'ont donné l'occasion de passer de l'autre côté du miroir en travaillant avec ceux qui étaient auparavant mes enseignants. Je remercie particulièrement Fabien Mangione, Hadrien Cambazard, Khaled Hadj-Hamou, Pierre David, Pierre Lemaire, Yannick Frein.

Je souhaite aussi saluer l'ensemble des ingénieurs, stagiaires et doctorants du laboratoire G-SCOP, sans qui ces quelques années auraient paru bien plus mornes. Je pense spécialement à Amine, Anis, Emna, Karim, Khalil, Kléber, Rami, Saleh, Widad. Je remercie également mes co-bureaux, Idris Lalami, Matthieu Yager et Pierre Lavayssière.

Enfin, je remercie mes parents pour m'avoir soutenu durant toutes mes études, ma

soeur Safa, mon frére Basset et Amal, qui ont largement contribué à l'aboutissement de ce projet de thèse.

Une page se tourne, une autre s'ouvre, merci à tous ceux et celles qui y ont contribué et m'ont amené à faire les bons choix.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Les tendances actuelles de la recherche dans la théorie de l'ordonnancement visent, depuis plusieurs années, à mieux modéliser les problèmes rencontrés dans des cas pratiques et industriels. Ces recherches ont donné naissance à des modèles qui combinent l'aspect ordonnancement et la prise de décision logistique telles que la disponibilité des machines, le transport, etc.

Les problèmes d'ordonnancement ont été largement étudiés depuis soixante ans. Dans la plupart de ces problèmes, les machines sont considérées disponibles. Dans un cas réel, cette hypothèse n'est pas toujours vraie. En effet, les machines peuvent connaître des périodes d'indisponibilité correspondant à des périodes de maintenance préventive, pannes, ou un changement d'outils.

Depuis le début des années 1990, plusieurs travaux ont été menés sur l'ordonnancement en présence de périodes d'indisponibilité. La plupart de ces travaux considèrent que la durée de ces périodes est fixe, *i.e.* la date de début et de fin sont considérées comme connues et certaines (cf. [81, 102]).

Les périodes d'indisponibilité peuvent être considérées comme une variable de décision, on parle dans ce cas d'un ordonnancement simultané des tâches et des indisponibilités. En effet, la planification de périodes d'indisponibilité, souvent considérées comme des périodes de maintenance, est très importante et vise à améliorer l'efficacité de l'entreprise en termes de coûts de production et de délais de livraison. Par exemple, dans l'étude de Singh et *al.* [104], la planification de maintenance permet de réduire les temps d'arrêt de 13,64 % et d'augmenter les bénéfices de l'entreprise.

Outre cela, l'état de la machine peut affecter les performances de la production. Ces variations sont généralement modélisées, dans la littérature, par un effet de détérioration ou un effet d'apprentissage. Par exemple, plus la machine est utilisée, plus le temps opératoire de la tâche augmente, ce qui est dû au vieillissement de la machine. Ainsi, l'insertion

de ces périodes de maintenance est légitime afin de rétablir les conditions initiales de production.

Dans cette thèse, nous nous intéressons à l'ordonnancement simultané des tâches et des périodes d'indisponibilité, en particulier celles dont les durées sont variables, dans les différents ateliers de production. La résolution d'un problème d'ordonnancement simultané de tâches et de périodes d'indisponibilité consiste à trouver la séquence de tâches, le nombre de maintenances et leurs positions qui minimisent le critère d'optimisation. Notre travail vise à proposer des méthodes de résolution pour plusieurs critères d'optimisation.

Cette thèse est organisée en trois parties :

- **Partie I : Contexte et état de l'art**
  Cette première partie consiste à présenter d'une manière générale le contexte de notre étude et présenter les différents travaux menés sur les problèmes d'ordonnancement en présence de périodes d'indisponibilité.
  Nous présentons tout d'abord dans le premier chapitre (chapitre 1) les notions de base d'un problème d'ordonnancement, les notations ainsi que les différents ateliers de production. Pour faciliter la compréhension de la problématique, nous décrivons aussi les différents effets appliqués sur les machines et les différents modèles de maintenance. Dans le deuxième chapitre (chapitre 2), un état de l'art est réalisé, qui vise à présenter un état des lieux des connaissances de la prise en compte des indisponibilités dans les problèmes d'ordonnancement.
  L'objectif de cette partie est non seulement de retracer l'état de l'art des problèmes d'ordonnancement avec la prise en compte de périodes d'indisponibilité, mais aussi d'identifier les écarts entre les différents travaux et de bien positionner notre étude par rapport à la littérature existante.
- **Partie II : Approche globale**
  Dans cette partie, nous développons une approche globale pour résoudre les problèmes de la prise en compte des indisponibilités dans un atelier à une machine et à machines parallèles. Pour ceci, nous décrivons dans le chapitre 3 un modèle général qui englobe les modèles antérieurs existants dans la littérature, dans lequel un nouveau système de maintenance a été présenté. Nous présentons aussi, dans ce chapitre, le problème d'affectation linéaire qui sera le modèle sous-jacent principal pour développer notre approche de résolution.
  Les deux chapitres (4 et 5) sont consacrés à la résolution du modèle global dans un atelier à une machine et à machines parallèles, respectivement. Nous proposons finalement dans le chapitre 6 une étude des cas particuliers et des extensions issus du modèle global.
  Les travaux présentés, dans cette partie, permettent de résoudre des problèmes

d'ordonnancement non traités avant et aussi de généraliser et améliorer des résultats antérieurs.

– **Partie III : Problème flow shop**

Cette partie est consacrée à la résolution d'un problème d'ordonnancement dans un atelier flow shop à deux machines en présence d'une période d'indisponibilité sur la deuxième machine. Une étude de complexité est menée sur le problème. Ensuite, nous définissons des propriétés d'optimalité. En se basant sur ces propriétés, trois méthodes de résolution exacte sont proposées ; une méthode énumérative, un programme linéaire et une méthode basée sur l'approche de séparation et évaluation ($B\&B$). Finalement, nous étudions expérimentalement les différentes méthodes proposées sur plusieurs instances, afin de vérifier l'efficacité des méthodes.

À la fin de ce document, nous proposons une conclusion générale et donnons des perspectives de recherche concernant les problèmes d'ordonnancement de tâches et de périodes d'indisponibilité.

Première partie

Contexte et état de l'art

1

Ordonnancement de la production et périodes d'indisponibilité

Résumé : Ce chapitre présente le contexte global de la thèse. Nous proposons, en premier lieu, les notions de base des problèmes d'ordonnancement et les notations utilisées dans le reste de la thèse. En second lieu, nous présentons les différents effets appliqués sur la machine et aussi une analyse sur la prise en compte des périodes d'indisponibilité dans les problèmes d'ordonnancement.

Sommaire

1.1 Introduction

Une grande partie de la théorie d'ordonnancement suppose que les machines sont toujours disponibles. Cette hypothèse n'est pas toujours vraie. En effet, la réalité industrielle prévoit des périodes d'indisponibilité des machines pour la maintenance, le changement d'outil, l'indisponibilité des opérateurs, etc.

Dans ce chapitre, nous donnerons le contexte dans lequel se situe la thèse, en présentant un certain nombre de généralités et de définitions sur les problèmes d'ordonnancement, dans la première section.

La deuxième section sera consacrée à la présentation des différents effets appliqués sur les machines de production. Enfin, nous présentons une analyse sur les différents modèles d'indisponibilité utilisés dans la littérature.

1.2 Généralités & notations

1.2.1 Généralités

> « *Ordonnancer, c'est programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution.* [...]. *Enfin, il faut programmer les tâches de façon à optimiser un certain objectif...* »
>
> J. Carlier et P. Chrétienne, *Problèmes d'ordonnancement [16]*, 1988

Les problèmes d'ordonnancement consistent à organiser dans le temps la réalisation d'un ensemble de tâches en tenant compte de contraintes et en leur allouant les ressources requises, et dont le but est l'optimisation d'un objectif fixé.

Une tâche est une entité caractérisée par un nombre d'opérations à réaliser. Chaque opération nécessite une durée de réalisation et des ressources. Selon les problèmes, les tâches peuvent être réalisées sans interruption (tâches non-préemptives) ou par morceaux (tâches morcelables ou encore préemptives).

Les ressources nécessaires à l'exécution de tâches peuvent être de différents types ; technique ou humain. Une ressource est renouvelable si après chaque utilisation elle reste disponible dans la même quantité et dans le même état de fonctionnement qu'avant l'utilisation. Lorsque la quantité d'une ressource diminue au fur et à mesure de son utilisation, on parle d'une ressource consommable.

1.2.1.1 Les ateliers

Dans les problèmes d'ordonnancement, plusieurs types d'ateliers de production sont considérés. Un atelier se définit par le nombre de machines qu'il contient et par son type. Les différents types possibles sont les suivants :

Une machine : les problèmes d'atelier à une machine sont les plus traités dans la littérature. Les tâches à réaliser sont composées d'une seule opération, et donc le problème consiste à déterminer le séquencement optimal des tâches sur la machine en respectant les contraintes.

Machines parallèles : les problèmes à machines parallèles sont considérés comme une généralisation des problèmes à une machine. Dans cet atelier, les tâches sont constituées d'une seule opération en présence de plusieurs machines ayant les mêmes fonctionnalités :

- Machines parallèles identiques (P) : les durées opératoires des tâches ne dépendent pas des machines, *i.e.* la vitesse d'exécution d'une tâche est la même sur toutes les machines.
- Machines parallèles uniformes (Q) : chaque machine a une vitesse d'exécution propre et constante.
- Machines parallèles non-uniformes (R) : la vitesse d'exécution est différente pour chaque machine et pour chaque tâche.

Machines dédiées : dans ce cas, chaque tâche est constituée de plusieurs opérations et les machines sont spécialisées à l'exécution de certaines opérations. Trois types d'ateliers sont considérés :

- Flow shop (F) : dans cet atelier le cheminement est unique, les tâches à réaliser passent par toutes les machines dans le même ordre. Dans la littérature, un cas particulier est souvent étudié, pour lequel la séquence de passage des tâches sur chaque machine est la même sur toutes les machines [1].
- Job shop (J) : cet atelier est considéré comme une généralisation du flow shop. En effet, la gamme opératoire est différente d'une tâche à l'autre, autrement dit, le passage des tâches sur les machines peut être différent d'une tâche à l'autre. Les premiers résultats théoriques concernant le job shop datent des années cinquante avec la résolution du problème à deux machines et d'un cas particulier à trois machines [44].
- Open shop (O) : cet atelier est le moins étudié dans la littérature et le moins utilisé dans les entreprises. L'ordre d'exécution des opérations n'est pas fixé.

1. Appelé ordonnancement de permutation.

1.2.1.2 Représentation des ordonnancements

Les ordonnancements sont représentés par un diagramme de Gantt, ce dernier a été développé par H.-L. Gantt en 1910. Le diagramme de Gantt est utilisé aussi dans les entreprises pour la planification des projets. Dans le cas de la gestion de la production, ce diagramme se compose de plusieurs lignes horizontales (ou des étages) qui représentent les machines, les tâches exécutées sur une machine donnée sont représentées par des rectangles de longueur proportionnelle à leur durée. Par conséquent, le diagramme de Gantt donne un aperçu des tâches exécutées sur l'ensemble des machines au cours du temps.

La figure 1.1 présente un diagramme de Gantt pour un atelier de trois machines parallèles et identiques. Les temps opératoires de chaque tâche sont présentés dans le tableau 1.1.

Tableau 1.1 – Durées des tâches

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
p_j	4	2,5	1,5	3	3	3	1	1	2

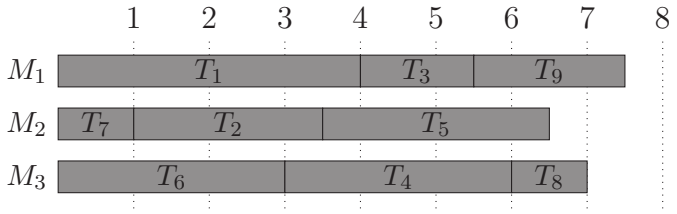

Figure 1.1 – Diagramme de Gantt

1.2.2 Notations

Avant d'entamer la présentation des différents travaux sur les problèmes d'ordonnancement en présence de périodes d'indisponibilité, il est primordial de définir les notations utilisées. Le tableau 1.2 introduit les notations que nous adoptons dans la suite de ce manuscrit, issues (pour la plupart) de la littérature. Soient $T = \{T_1, T_2, \ldots, T_n\}$ l'ensemble des tâches à ordonnancer et $M = \{M_1, M_2, \ldots, M_m\}$ l'ensemble des m machines.

Chaque tâche T_j sur une machine M_l est caractérisée par sa date de début s_j^l et sa date de fin C_j^l, notons que lorsque la préemption n'est pas autorisée, la différence entre la date de début et date de fin $(C_j^l - s_j^l)$ est égale au temps opératoire de la tâche T_j. D'autres données peuvent être associées aux tâches telles que la date de disponibilité de la tâche r_j et la date d'exécution au plus tard d_j.

Tableau 1.2 – Notations

n	: nombre de tâches
m	: nombre de machines
r	: position de la tâche dans la séquence de l'ordonnancement
$p(l, j, r)$	: temps opératoire de la tâche T_j à la position r sur M_l
C_j	: date de fin de la tâche T_j
m_{li}	: durée de la période d'indisponibilité i sur la machine M_l
t_{li}	: durée d'exécution séparant deux indisponibilités $i-1$ et i sur M_l
k_l	: nombre d'indisponibilités sur la machine M_l
$k = \sum_{l=1}^{m} k_l$	: nombre d'indisponibilités sur l'ensemble des machines
k_0	: borne supérieure du nombre d'indisponibilités
G_i^l	: groupe des tâches entre l'indisponibilité $i-1$ et i sur la machine M_l

Graham *et al.* [45] ont présenté une notation standard pour le problème d'ordonnancement. Cette notation se compose de trois champs $\alpha|\beta|\gamma$. Le champ α définit l'environnement d'exécution, et est composé de deux sous-champs α_1 et α_2 :

- α_1 : ce paramètre indique le type de l'environnement d'exécution ;
  $\alpha_1 = \emptyset$: une machine
  $\alpha_1 = P$: machines parallèles identiques
  $\alpha_1 = Q$: machines parallèles uniformes
  $\alpha_1 = R$: machines parallèles non-uniformes
  $\alpha_1 = F$: Flow shop
  $\alpha_1 = J$: Job shop
  $\alpha_1 = O$: Open shop
- α_2 : ce paramètre indique le nombre de machines utilisées. Si α_2 n'est pas précisé, le nombre de machines est quelconque.

Le champ β nous renseigne sur les caractéristiques des tâches et des machines (périodes d'indisponibilité, durée, date au plus tôt et au plus tard, contrainte de précédence, préemption ...).

Le dernier champ γ nous indique le critère d'optimisation à minimiser ou à maximiser. Le paramètre γ peut être une combinaison de plusieurs critères d'optimisation *(Problèmes d'ordonnancement multi-objectifs)*. Dans la sous-section suivante, nous présentons en détail les différents critères d'optimisation.

1.2.2.1 Critères d'optimisation

Généralement, l'objectif d'un problème d'ordonnancement consiste à optimiser un ou plusieurs critère(s) (minimisation de la durée totale, des pénalités...) ou à trouver une solution admissible qui respecte les contraintes du problème. Les variables intervenant le plus souvent dans l'expression de la fonction objectif sont : la date C_j de fin d'exécution

de la tâche T_j, les retards $L_j = C_j - d_j$, avec d_j la date d'exécution au plus tard de la tâche T_j, etc. Nous présentons, ci-dessous, une sélection des différents critères d'optimisation utilisés :

- **La durée totale (le makespan)** : notée aussi C_{max} est la durée totale de l'ordonnancement ou charge totale de la machine est égale à la date de fin de la tâche la plus tardive ; $C_{max} = max\{C_j\}$. Minimiser la durée totale de l'ordonnancement revient à minimiser C_{max}, *i.e.* $min\{\max_j\{C_j\}\}$.
- **La charge totale des machines (TML)** : la charge totale d'une machine correspond à sa durée d'exécution, ainsi pour les problèmes avec plusieurs machines (m machines) la charge totale des machines est définie par :

$$\text{Charge totale} = \sum_{l=1}^{m} C_{max}^{l}$$

  avec C_{max}^{l} la durée totale d'exécution sur la machine M_l.
- **La somme des dates de fin (ou minimiser les encours) TC** : ce critère consiste à minimiser la somme des dates de fin C_j de chaque tâche : $TC = \sum_{j=1}^{n} C_j$.
- **La somme des durées d'attente (TW ou $\sum W_j$)** : la durée d'attente W_j d'une tâche est égale à sa date de lancement $C_j - p_j$, avec C_j et p_j, respectivement, la date d'achèvement et le temps opératoire de la tâche j. Ainsi, minimiser la somme des durée d'attente est équivalent à minimiser cette expression :

$$TW = \sum_{j=1}^{n} W_j = \sum_{j=1}^{n} (C_j - p_j)$$

- **Les retards** : dans certains secteurs d'activité, la minimisation des retards est un critère important dans les problèmes d'ordonnancement. Deux types de retard sont considérés ;
  - retard algébrique de la tâche T_j (écart par rapport à la fin souhaitée) : $L_j = C_j - d_j$
  - retard vrai de la tâche j : $T_j = \max\{C_j - d_j, 0\}$

  Dans la littérature, plusieurs critères ont été étudiés tels que la minimisation de : $T_{max} = \max\{T_j\}$ (le retard maximum), $\bar{T} = \frac{1}{n}\sum_{j=1}^{n} T_j$ (le retard moyen) ou le retard algébrique maximum $L_{max} = \max\{L_j\}$. Parfois, on s'intéresse au nombre de retards et non aux durées des retards (exemple une commande de client est satisfaite ou non), on définit alors le nombre de tâches en retard $\sum_{j=1}^{n} U_j$, avec $U_j = 1$ si la tâche j est en retard, zéro sinon. Dans plusieurs travaux, des poids w_j (pénalité de retard) ont été ajoutés pour chaque tâche.
- **La variation des dates de fin (TADC)** : appelé aussi *Total absolute deviation of job completion times* ($TADC$). Ce critère d'optimisation a été introduit initialement

dans [65]. La variation des dates de fin est exprimée par l'expression suivante :

$$TADC = \sum_{i=1}^{n} \sum_{j=i}^{n} |C_j - C_i|$$

– **La variation des durées d'attente (TADW)** : appelé aussi *Total absolute deviation of job waiting times* ($TADW$) est exprimée par l'expression suivante (voir Bagchi [9]) :

$$TADW = \sum_{i=1}^{n} \sum_{j=i}^{n} |W_j - W_i|$$

– **Autres critères** : dans la littérature, plusieurs autres critères d'optimisation peuvent être étudiés. Citons par exemple des critères d'optimisation sous forme d'une fonction coût. Ces critères sont générés à partir d'une combinaison linéaire de critères présentés précédemment. Pour plus de détails sur ces critères, nous citons les travaux étudiés dans [42] et [10].

1.2.2.2 Notations liées à la maintenance et aux effets sur la machine

Dans cette sous-section, nous présentons les notations spécifiques de la notation à trois champs liées à la maintenance et aux effets appliqués sur la machine. Plusieurs notations ont été présentées dans la littérature. Dans les problèmes avec des périodes d'indisponibilité connues d'avance (intervalles d'indisponibilité), les indisponibilités sont considérées comme des contraintes liées à l'environnement d'exécution. Ainsi, un troisième sous-champ α_3 a été ajouté dans la notation à trois champs. Le paramètre $\alpha_3 \in \{\emptyset; h_{l,k}\}$ indique les périodes d'indisponibilité du problème ; si $\alpha_3 = \emptyset$, cela signifie que nous n'avons aucune période d'indisponibilité dans l'ordonnancement. $\alpha_3 = h_{l,k}$ indique que le problème possède un nombre quelconque d'indisponibilités sur chaque machine. Si l est remplacé par un entier positif, cela signifie que les indisponibilités sont seulement sur la machine M_l. Si k est remplacé par un entier positif, cela indique que le nombre de maintenances sur chaque machine est imposé, sinon le nombre d'indisponibilités est quelconque. Par exemple, lorsque α_3 égal à :

– $h_{1,k}$: le problème étudié considère un nombre quelconque de périodes d'indisponibilité sur la machine M_1 mais pas de maintenance sur les autres machines
– $h_{l,1}$: le problème étudié considère une seule période de maintenance sur toutes les machines
– $h_{2,1}$: le problème étudié considère une seule période de maintenance sur la machine M_2 seulement

Notons que pour les problèmes à une machine, nous pouvons supprimer l'indice l : $\alpha_3 = h_{1,k} = h_k$. Cette notation a été présentée en détail dans les surveys sur les problèmes

avec des indisponibilités connues d'avance de Ma *et al.* [81] et de Schmidt [102].

Un deuxième système de notation a été présenté dans d'autres articles (par exemple : [3], [134] et [126]), notamment dans les travaux avec des périodes d'indisponibilité considérées comme des variables de décision. Les informations des périodes d'indisponibilité ont été présentées dans le sous-champ β qui indique les caractéristiques du problème.

Dans cette thèse, nous considérons que les périodes d'indisponibilité sont liées aux caractéristiques du problème. Ainsi, nous ajoutons la notation liée à la maintenance et aux tâches dans le champ β de la notation à trois champs de Graham. Les sous-champs suivants sont considérés :

Notation liée aux durées variables des tâches

- $\beta_1 = \{\emptyset; p(l,j,r,t); p(l,j,r,t) = f(l,j,r,t)\}$ indique si le temps opératoire de la tâche est variable ou non. Si $\beta_1 = \emptyset$, cela signifie que les temps opératoires des tâches sont constants. Lorsque $\beta_1 = p(l,j,r,t)$, les temps opératoires varient selon (l,j,r,t) (avec l : la machine, j : la tâche, r : sa position dans la séquence de l'ordonnancement, t : la date de lancement de la tâche). Si $\beta_1 = \{p(l,j,r,t) = f(l,j,r,t)\}$, cela signifie que les temps opératoires suivent une fonction f bien définie et cette dernière varie en fonction de (l,j,r,t).

Notation liée aux maintenances

- $\beta_2 = \{\emptyset; \mathcal{M}_l; \mathcal{M}_l = k_0; \mathcal{M}_l \leq k_0\}$ le sous-champ β_2 indique le nombre de maintenances à insérer dans l'ordonnancement. Si $\beta_2 = \emptyset$, cela signifie qu'aucune période d'indisponibilité n'est autorisée dans le système. Si $\beta_2 = \mathcal{M}_l$, dans ce cas le nombre de maintenances à insérer sur chaque machine est une variable de décision. Si $\beta_2 = \{\mathcal{M}_l = k_0\}$, alors le nombre de maintenances sur chaque machine est imposé et est égal à cette valeur. Lorsque l est remplacé par un entier positif, cela signifie que les indisponibilités sont seulement sur la machine M_l. Lorsque $\beta_2 = \{\mathcal{M}_l \leq k_0\}$, on définit une borne supérieure de maintenances à insérer sur les machines. Notons que si on supprime l'indice l, *i.e.* $\mathcal{M} = k_0$ ou $\mathcal{M} \leq k_0$, on parle dans ce cas du nombre total de maintenances sur toutes les machines.
- $\beta_3 = \{\emptyset; m_{li} = g(l,j,i,r,t)\}$, le paramètre β_3 donne une idée sur la durée de maintenance, avec m_{li} la durée de la $i^{\text{ème}}$ maintenance sur la machine M_l. Si $\beta_3 = \{m_{li} = g(l,j,i,r,t)\}$, la maintenance suit une fonction g bien définie et varie selon les cinq paramètres (l,j,i,r,t) (avec l : la machine, j : la tâche, i : le groupe, r : la position dans le groupe, t : la date de lancement). Notons que pour les problèmes d'ordonnancement à une machine, nous pouvons supprimer l'indice l, *i.e.* m_i. Lorsque nous traitons un problème avec une seule période de maintenance ($\mathcal{M}_l = 1$) à ordonnancer sur chaque machine, nous pouvons supprimer l'indice i, *i.e.* m_l.

Afin de mieux comprendre cette notation, nous présentons ci-dessous des exemples de notation à trois champs :

Exemple 1. $1|p(j,r) = a_j + rb_j, \mathcal{M} \leq 2, m = cst|C_{max}$: représente le problème consistant à ordonnancer n tâches dans un atelier à une machine. Les temps opératoires suivent une fonction linéaire avec $p(j,r) = a_j + rb_j$. Dans ce problème, au plus deux maintenances doivent être insérées sur la machine avec des durées constantes égales à cst. L'objectif est la minimisation de la durée totale de l'ordonnancement.

Exemple 2. $1|p(j,r) = r^{a_j}, \mathcal{M}, m_i = cst_i|C_{max}$: représente le problème consistant à ordonnancer n tâches dans un atelier à une machine. Les temps opératoires suivent la fonction $p(j,r) = r^{a_j}$. La durée de la maintenance varie selon son rang i dans l'ordonnancement (cst_i). L'objectif du problème est de trouver la séquence de tâches, le nombre et les positions des maintenances qui minimisent le makespan.

Exemple 3. $Rm|p(l,j,r), \mathcal{M}_l, m_{li} = cst_l|TADC$: nous étudions dans cet exemple un problème avec m machines non liées. L'objectif est la minimisation de la variation des dates de fin. Les temps opératoires des tâches suivent une fonction quelconque. Le nombre de maintenances est une variable de décision à déterminer afin de minimiser le critère d'optimisation. Notons que les durées des périodes de maintenance sont constantes par machine (cst_l est la durée de chaque période de maintenance sur la machine $\mathcal{M}_l$).

Exemple 4. $Rm|p(l,j,r) = a_{lj} + rb_{lj}, \mathcal{M} \leq k_0, m_{li} = cst_{li}|\sum C^l_{max}$: dans ce problème, nous considérons un problème avec m machines non liées. Les tâches suivent une variation linéaire. Le nombre total de maintenances à ordonnancer sur l'ensemble des machines a une borne supérieure k_0. La durée de la maintenance varie selon la machine l et le rang de la maintenance i. L'objectif est la minimisation de la charge totale des machines.

Exemple 5. $F2|\mathcal{M}_2 = 1, m = f(t)|C_{max}$: cette notation représente le problème consistant à ordonnancer n tâches dans un atelier flow shop à deux machines et une maintenance sur la deuxième machine. La durée de la maintenance suit une fonction $f(t)$ qui varie selon sa date de début. L'objectif est de trouver la séquence de tâches et la position de la maintenance qui minimisent la durée totale de l'ordonnancement.

Notons que d'autres sous-champs peuvent être ajoutés au champ β tels que ; la préemption est autorisée ou non, les dates de début au plus tôt r_j et les dates de fin au plus tard d_j, etc.

1.3 Temps opératoires variables

Dans les problèmes classiques d'ordonnancement, les temps opératoires des tâches sont généralement constants et connus d'avance. Comme mentionné dans Rinnooy Kan [64], les hypothèses restrictives traditionnelles peuvent correspondre à une image un peu simpliste de la réalité, mais elles peuvent améliorer davantage la complexité du calcul. La disponibilité continue de la machine n'est pas toujours vraie, en effet, les durées de traitement des tâches peuvent être variables suite à la variation des conditions de la production. Dans plusieurs secteurs d'activités, l'état de la machine ou l'expérience des opérateurs peut affecter les performances de la production. Ces variations sont généralement modélisées dans la littérature par un effet de détérioration ou un effet d'apprentissage.

Quel que soit l'effet appliqué sur la machine (détérioration ou apprentissage), deux types de classes de variation des tâches sont étudiées dans la littérature :

- *Position-dependent effects :* la durée de traitement d'une tâche T_j varie en fonction de sa position dans l'ordonnancement r, *i.e.* $p(j, r)$. Citons notamment les travaux de Bachman et Janiak [7] et Rudek et Rudek [97].
- *Time-dependent effects :* la durée de traitement d'une tâche T_j varie en fonction de sa date de début t, *i.e.* $p(j, t)$. Cheng *et al.* présentent dans [20] une synthèse des articles qui considèrent des problèmes d'ordonnancement avec des tâches variables en fonction de leur date de début. Citons également le livre de Gawiejnowicz [39] et l'article de Alidaee et Womer [2].

1.3.1 Effets de détérioration

Les problèmes d'ordonnancement avec un effet de détérioration [2] ont été étudiés, au cours des dernières décennies, avec plusieurs configurations de problèmes et de mesures de performance. Gupta et Gupta [48] et Browne et Yechiali [14] ont probablement été les premiers chercheurs à introduire l'effet de la détérioration dans les problèmes d'ordonnancement.

Dans les problèmes avec un effet de détérioration, plus la machine est utilisée, plus le temps opératoire de la tâche augmente, cette variation est due au vieillissement de la machine.

L'effet de détérioration peut être décrit par l'exemple suivant ; supposons que certains produits nécessitent un traitement par un outil spécifique, par exemple une opération de perçage. Le temps requis pour le traitement d'un seul produit dépend de la qualité de l'outil. À cause de l'usure de l'outil, le temps opératoire nécessaire pour traiter certains

2. Appelé aussi effet de vieillissement : Aging effects.

produits augmente en fonction du nombre de produits déjà traités. Ainsi, le temps opératoire d'une tâche peut être décrit par une fonction croissante selon le nombre de tâches précédemment exécutées.

Plusieurs fonctions analytiques ont été utilisées, dans la littérature, pour illustrer l'effet de la détérioration. On distingue, comme cela a été précisé précédemment :

– Détérioration en fonction de la position de la tâche : plus la position de la tâche est grande, plus sa durée de traitement augmente

$$p(j,1) \leq p(j,2) \leq \ldots \leq p(j,n), \quad j = 1, \ldots, n. \tag{1.1}$$

avec $p(j,r)$ le temps opératoire de la tâche j dans la position r.

– Détérioration en fonction de la date de début de la tâche : plus la date de début d'exécution de la tâche est tardive, plus sa durée de traitement augmente

$$p(j,t_1) \leq p(j,t_2), \quad j = 1, \ldots, n, \quad \forall\, t_1 \leq t_2. \tag{1.2}$$

avec t_r la date de début de la tâche ordonnancée dans la position r et $p(j,t_r)$ son temps opératoire.

La motivation de mener des études sur ces problèmes découle de l'existence de nombreuses applications de la vie réelle qui peuvent être formulées sous la forme de problèmes d'ordonnancement de tâches avec un effet de détérioration. Par exemple, la détérioration a été utilisée dans plusieurs travaux pour modéliser les systèmes de production avec ou sans périodes de maintenance. Yang et *et al.* [123] ont étudié un système de production à machines parallèles avec des outils de coupe. Les machines suivent un effet de détérioration, à cause de l'usure de l'outil de coupe. Des périodes de maintenance peuvent être insérées pour changer les outils, ainsi la machine revient à son état initial avant la détérioration. Gupta *et al.* [49] ont utilisé un problème à une machine avec un effet de détérioration pour étudier les politiques de paiement optimales pour rembourser plusieurs prêts. Rachaniotis et Pappis, dans [96], montrent, quant à eux, que l'allocation d'une seule ressource lorsque plusieurs incendies doivent être maîtrisés dans une forêt, peut être modélisée par un problème d'ordonnancement à une machine avec un effet de détérioration.

De nombreuses recherches ont été menées sur l'ordonnancement avec un effet de détérioration, citons notamment Bachman et Janiak [5, 6], Bachman *et al.* [8], Browne et Yechiali [14] et Ng *et al.* [92]. Les tableaux 1.3 et 1.4 présentent un aperçu des différentes fonctions utilisées (positionnelles et temporelles) dans la littérature.

1.3.2 Effets d'apprentissage

L'apprentissage est un facteur important dans les systèmes de production industriels. L'effet d'apprentissage a été initialement introduit dans les problèmes d'ordonnancement par Biskup [11]. Depuis lors, les problèmes d'ordonnancement avec effet d'apprentissage ont reçu une attention croissante. Dans l'ordonnancement avec un effet d'apprentissage, la durée de traitement d'une tâche devient plus courte, à condition que la tâche soit planifiée tard. Nous pouvons illustrer cet effet par l'exemple suivant ; supposons qu'un opérateur doit assembler un certain nombre de produits similaires. Le temps nécessaire par l'opérateur pour assembler un seul produit dépend de sa connaissance, ses compétences, l'organisation de son lieu de travail et d'autres facteurs. Pendant le processus d'assemblage, l'opérateur apprend à produire. Ainsi, chaque produit assemblé par l'opérateur a une influence sur ses compétences, une meilleure organisation de son lieu de travail et sa connaissance. Grâce à son apprentissage, le temps nécessaire pour assembler les prochains produits diminue. Cette diminution du temps opératoire de la tâche peut être présentée par une fonction décroissante selon le nombre de produits déjà assemblés.

Nous citerons les travaux de Biskup [11], Kuo et Yang. [69] et Ji *et al.* [62]. Pour plus de précision, nous renvoyons le lecteur vers la revue de Biskup [12] sur les problèmes qui considèrent des machines avec un effet d'apprentissage.

On distingue, comme cela a été précisé précédemment, des fonctions qui varient :

– en fonction de la position de la tâche :

$$p(j,1) \geq p(j,2) \geq \ldots \geq p(j,n), \quad j = 1, \ldots, n. \tag{1.3}$$

avec $p(j,r)$ le temps opératoire de la tâche j dans la position r.

– en fonction de la date de début de la tâche :

$$p(j,t_1) \geq p(j,t_2), \quad j = 1, \ldots, n, \quad \forall\, t_1 \leq t_2. \tag{1.4}$$

avec t_r la date de début de la tâche ordonnancée dans la position r et $p(j,t_r)$ son temps opératoire.

Plusieurs applications réelles de l'effet d'apprentissage ont été étudiées. Souvent, l'effet d'apprentissage est utilisé dans les problèmes qui modélisent les systèmes de production manuels et semi-automatiques. Ho *et al.* [54] ont utilisé un problème d'ordonnancement avec un effet d'apprentissage pour modéliser un problème de reconnaissance des menaces aériennes, en effet le temps nécessaire de reconnaissance diminue lorsque la menace s'approche de la station radar.

Partant du modèle d'apprentissage de Biskup [11], une grande variété de modèles basés sur la position de la tâche r ont été étudiés dans la littérature. Les tableaux 1.3 et 1.4

donnent un aperçu des différents modèles proposés (positionnels et temporels).

Plusieurs autres extensions existent dans la littérature, citons par exemple Cheng *et al.* [23], Okolowski et Gawiejnowicz [93], Yin *et al.* [137], Low et Lin [79], Lu *et al.* [80], Cheng *et al.* [22], Wang et Wang. [113] et Ji *et al.* [61].

Tableau 1.3 – Les différents modèles de temps opératoire présentés dans la littérature (variation selon la position r)

Fonction $p(j,r)$	Effets de détérioration où a, a_j, b_j, σ et σ_j sont des facteurs de détérioration	Effet d'apprentissage où a, a_j, b_j, σ et σ_j sont des facteurs d'apprentissage	Références
$p_j r^a$	$a > 0$	$a < 0$	Biskup [11] ; Mosheiov [87, 88]
$p_j r^{a_j}$	$a_j > 0$	$a_j < 0$	Mosheiov et Sidney [89] ; Yang et Yang [134, 135] ; Zhao et Tang [138]
$p_j + b_j r$	$b_j > 0$	$b_j < 0$	Bachman et Janiak[7] ; Yang et Yang [134] ; Yang *et al.* [123] ; Hsu *et al.* [56]
$p_j \sigma^{r-1}$	$\sigma > 1$	$0 < \sigma < 1$	Gordon *et al.* [41]
$p_j \sigma_j^{r-1}$	$\sigma_j > 1$	$0 < \sigma_j < 1$	Yang et Yang [136]
$p_j g(r)$	$g(r)$ fonction croissante	$g(r)$ fonction décroissante	Rustogi et Strusevich [98, 99]
$p_j g_j(r)$	$g_j(r)$ fonction croissante	$g_j(r)$ fonction décroissante	Rustogi et Strusevich [98, 99]

Tableau 1.4 – Les différents modèles de temps opératoire présentés dans la littérature (variation selon le temps t)

Fonction $p(j,t)$	Effets de détérioration où b, b_j et $f(t)$ sont des facteurs de détérioration	Effet d'apprentissage où b, b_j et $f(t)$ sont des facteurs d'apprentissage	Références
$p_j t$	✓	-	Mosheiov [85] ; Gawiejnowicz [38] ; Cheng et Sun [18] ; Ji *et al.* [60] ; Wu *et al.* [120]
$p_j(a + bt)$	$b > 0$	$b < 0$	Kononov [66] ; Guo et Wang [47] ; Wang *et al.* [112] ; Wang et Xia [115]
$p_j + b_j t$	$b_j > 0$	$b_j < 0$	Gawiejnowicz et Pankowska [40] ; Gupta et Gupta [48] ; Tanaev *et al.* [105] ; Cheng et Ding [19] ; Bosio et Righini [13] ; Wang *et al.* [110] ; Hsu *et al.* [56]
$p_j + f(t)$	$f(t)$ fonction croissante	$f(t)$ fonction décroissante	Kuo et Yang [70]

1.3.3 Autres effets

D'autres modèles ont été utilisés dans la littérature qui combinent les deux effets d'apprentissage et de détérioration. Par exemple, Yang [126] a étudié un problème d'ordonnancement avec un effet combiné ; l'effet d'apprentissage varie en fonction la date de début de la tâche et un effet de détérioration varie en fonction de la position de la tâche.

$$p_{jr} = (p_j - at)r^b \tag{1.5}$$

avec $a > 0$ le facteur d'apprentissage, $b \geq 0$ le facteur de détérioration et t la date de début de la tâche T_j.

Les problèmes d'ordonnancement qui considèrent les deux effets en même temps ont été largement étudiés dans la littérature, voir Ji *et al.* [61], Huang *et al.* [57] ; Kuo et Yang [107] ; Toksari et Güner [106] ; Wang [108, 109] ; Wang *et al.* [111] ; Wang et Wang [118] ; Wu *et al.* [121] ; Yang et Kuo [124].

1.4 Périodes d'indisponibilité

Dans la littérature, plusieurs modèles de périodes d'indisponibilité ont été étudiés. À cet égard, il devrait être noté, avant d'entamer la présentation de ces modèles, que deux points importants caractérisent une période d'indisponibilité :

- l'impact de la période d'indisponibilité sur la machine.
- la nature de la période d'indisponibilité, *i.e.* connue d'avance ou variable de décision.

Nous examinons en détail ces points dans les deux sections suivantes.

1.4.1 Impact de l'indisponibilité sur la machine

Initialement, lors de l'introduction de l'indisponibilité dans les problèmes d'ordonnancement en 1996 par Lee [72, 73], les périodes d'indisponibilité ont été considérées comme des périodes d'indisponibilité sans impact sur les performances de la machine.

Peu de temps après, Lee et Leon [75] ont été les premiers à étudier un problème d'ordonnancement avec une période d'indisponibilité qui permet de changer les performances de la machine (RMA en anglais : *'The rate-modifying activity'*). Dans ce modèle, si la tâche T_j est ordonnancée avant la maintenance, la durée d'exécution de cette tâche est égale à p_j, la durée d'exécution est égale à $\delta_j p_j$ si la tâche T_j est ordonnancée après la maintenance, avec $0 < \delta_j$ le taux de modification des performances par la tâche T_j, voir figure 1.2. D'autres modèles ont aussi été étudié dans [90] avec b_j et a_j, respectivement, la durée de la tâche T_j avant et après la maintenance (figure 1.3). Cette variation peut

être représentée dans les notations à trois-champs par : (b_j, a_j).

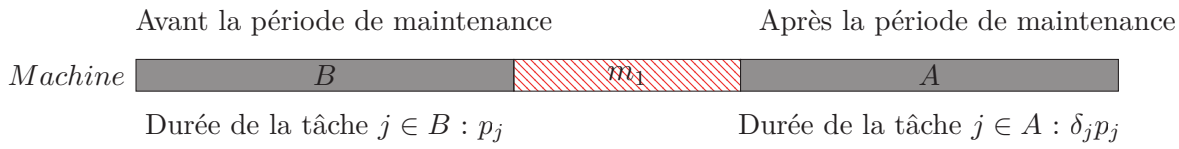

Figure 1.2 – Une période d'indisponibilité avec changement des performances $(p_j, \delta_j p_j)$

Figure 1.3 – Une période d'indisponibilité avec changement des performances (b_j, a_j)

Notons que pour les RMA, deux cas sont possibles :

- l'insertion de la période d'indisponibilité peut améliorer les performances de la machine lorsque $0 < \delta_j \leq 1$ (ou plus généralement $b_j \geq a_j$). C'est le cas le plus étudié dans les travaux antérieurs et le plus proche de la réalité industrielle. Dans ces problèmes, la période d'indisponibilité est souvent considérée comme une période de maintenance.
- l'insertion de la période d'indisponibilité peut augmenter le temps opératoire de la tâche lorsque $1 < \delta_j$ (ou plus généralement $b_j < a_j$), avec $j = 1, \ldots, n$.

Plus récemment, les périodes d'indisponibilité ont été insérées pour contrer les effets de détérioration appliqués sur les machines et ainsi rétablir les conditions initiales de production. Plusieurs travaux ont été menés dans cet axe de recherche. Pour bien expliquer ce type de problème, nous présentons dans l'exemple ci-dessous (figure 1.4), un problème d'ordonnancement à une machine avec un effet de détérioration exponentielle $p(j, r) = p_j r^a$ et une période de maintenance MP de durée t_0 qui permet de rétablir les conditions initiales de la machine. Soit la séquence suivante $S = \{T_3; T_1; T_2; MP; T_4; T_5\}$, nous insérons après la période de maintenance les tâches T_4 et T_5.

Après la période d'indisponibilité MP, la machine revient à son état initial. Ainsi, les rangs r sont réinitialisés après la maintenance. Le temps opératoire de la tâche T_4 est $p(4, r = 1) = p_4 1^a$ et $p(5, r = 2) = p_5 2^a$ pour la tâche T_5, voir figure 1.4.

1.4.2 Nature de la période d'indisponibilité

Dans la littérature plusieurs modèles ont été proposés. Deux types de périodes d'indisponibilité ont été étudiés, le premier type concerne les périodes connues d'avance, *i.e.*

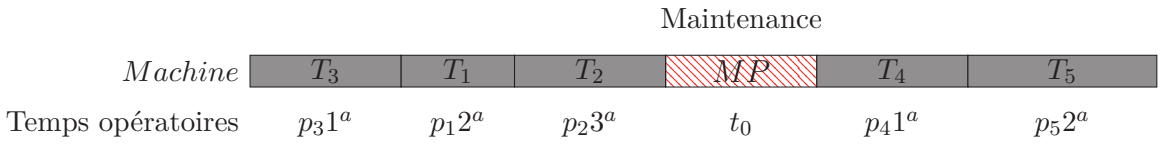

Figure 1.4 – Problème à une machine avec une indisponibilité en présence d'un effet de détérioration

les dates de début et de fin sont fixées. Dans le second type, la période d'indisponibilité est une variable de décision.

Lorsque la période d'indisponibilité est connue d'avance, elle peut être présentée sous forme d'un intervalle d'indisponibilité ; autrement dit, la machine est indisponible entre $[s, t]$, voir figure 1.5. Dans ce cas, la date de début et de fin de l'indisponibilité est connue d'avance.

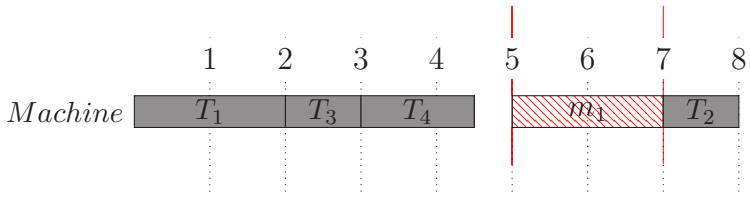

Figure 1.5 – Problème avec une seule fenêtre d'indisponibilité $[5, 7]$

Outre cette vision statique, les périodes d'indisponibilité peuvent être considérées comme des variables de décision, dans ce cas le problème à résoudre est un problème d'ordonnancement simultané des tâches et des périodes d'indisponibilité. Notons que ces dernières sont souvent considérées comme des périodes de maintenance. Dans la suite, nous allons focaliser notre propos sur l'ordonnancement conjoint de la production et des périodes d'indisponibilité.

Dans la littérature, plusieurs travaux ont été menés sur les ordonnancements simultanés des tâches et des périodes d'indisponibilités. Dans ces travaux, une ou k périodes d'indisponibilité peuvent être placées sur un système de production. Différentes durées de maintenance ont été proposées.

La durée de toutes les périodes de maintenance peut être constante : $m_i = \beta$, $\forall i = 1, \ldots, k$, avec β la durée élémentaire de la maintenance. Ce modèle reflète le fait que très souvent la maintenance est constituée d'un nombre fixe d'actions de maintenance de routine à effectuer.

Les durées de la maintenance peuvent être constantes, mais différentes d'une période à autre, *i.e.* $m_i = \beta_i$ pour la période i, voir figure 1.6. En effet, plusieurs équipes de maintenance avec différentes compétences et expériences peuvent être amenées à intervenir.

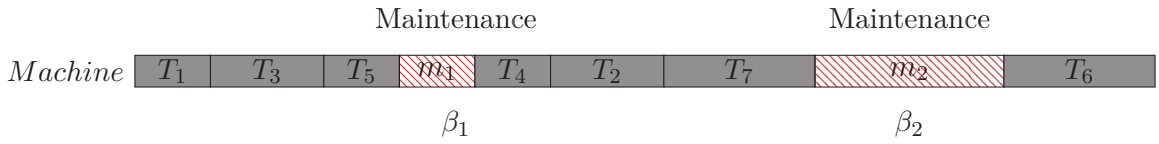

Figure 1.6 – Deux périodes de maintenance de durée constante sur une seule machine

Dans [134], Yang et Yang ont utilisé ce modèle de maintenance pour illustrer un effet d'apprentissage. La durée de la $i^{\text{ème}}$ maintenance est $m_i = t_0 i^b$, avec $b < 0$ (figure 1.7). En effet, l'équipe de la maintenance peut apprendre en répétant les mêmes activités de maintenance, et donc la durée de l'intervention de l'équipe de la maintenance est moins longue que la durée de la période de maintenance précédente.

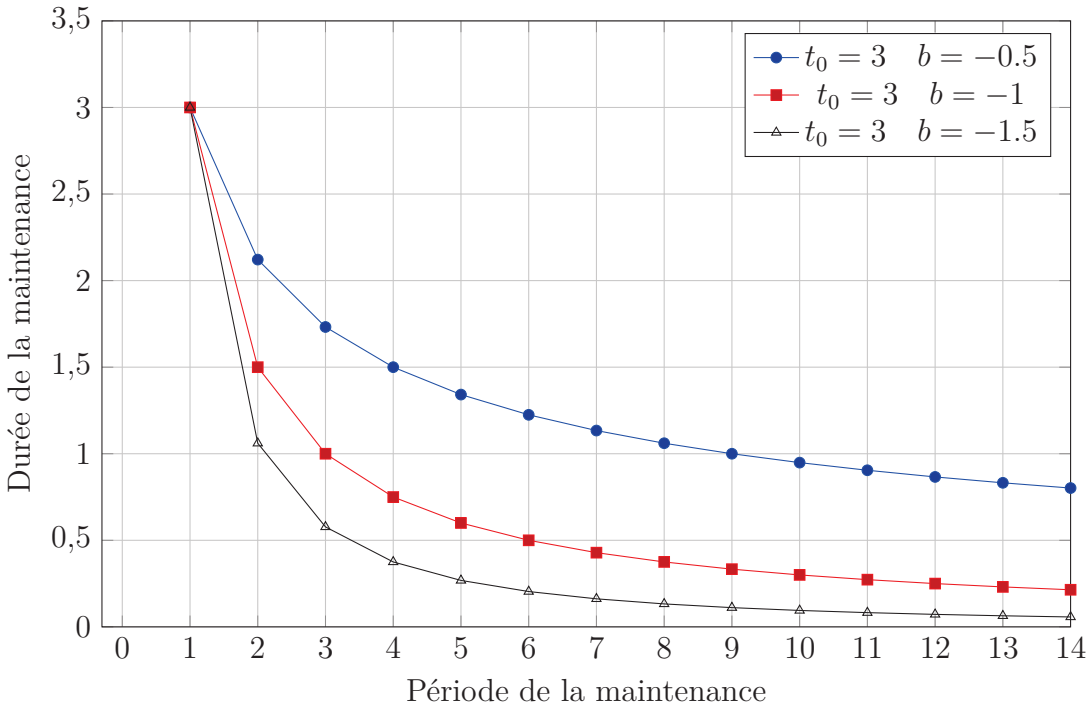

Figure 1.7 – Maintenance avec un effet d'apprentissage, $t_0 = 3$ et $b = \{-0.5; -1; -1.5\}$, voir [134]

En 2005, Kubzin et Strusevich ont introduit un nouveau modèle de maintenance de durée variable. La durée de la maintenance varie linéairement en fonction de sa date de début. En effet, plus la période séparant deux maintenances est importante, plus la maintenance à réaliser est lourde et donc plus sa durée est importante.

> *The goal of this paper is to initiate research on scheduling models with maintenance periods of controllable length. In many manufacturing environments the length of machine maintenance depends on its start time. This happens when maintenance includes cleaning, recharging, refilling, or partial replacement of tools or parts that have been subject to essential wear.*
>
> M. A. Kubzin et V. A. Strusevich, *Planning machine maintenance in two-machine shop scheduling [68]* , 2006

Dans [67] et [68], Kubzin et Strusevich considèrent la durée de la maintenance sous la forme d'une fonction linéaire (problème avec une seule période de maintenance) :

$$m = \alpha \times t + \beta \tag{1.6}$$

avec t la date début de la maintenance, α le facteur de la détérioration et β sa durée élémentaire.

Pour un système de production multi-maintenance, la durée de la $i^{\text{ème}}$ période de maintenance est :

$$m_i = \alpha_i \times t_i + \beta_i, \quad \forall i = 1, \ldots, k \tag{1.7}$$

avec t_i la durée d'exécution entre la maintenance $(i-1)$ et i, α_i et β_i le facteur de la détérioration et la durée élémentaire de la $i^{ème}$ maintenance, respectivement. Voir exemple sur la figure 1.8.

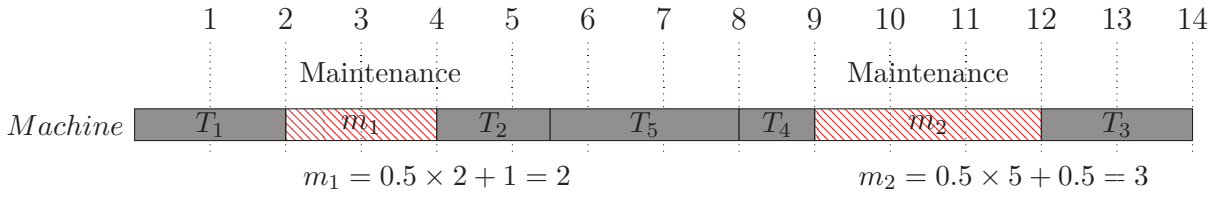

Figure 1.8 – Modèle de maintenance linéaire avec $\alpha_1 = \alpha_2 = 0,5$, $\beta_1 = 1$ et $\beta_2 = 0.5$

En se basant sur les sous-sections précédentes, la figure 1.9 illustre un récapitulatif sur la prise en compte des périodes d'indisponibilité dans les problèmes d'ordonnancement.

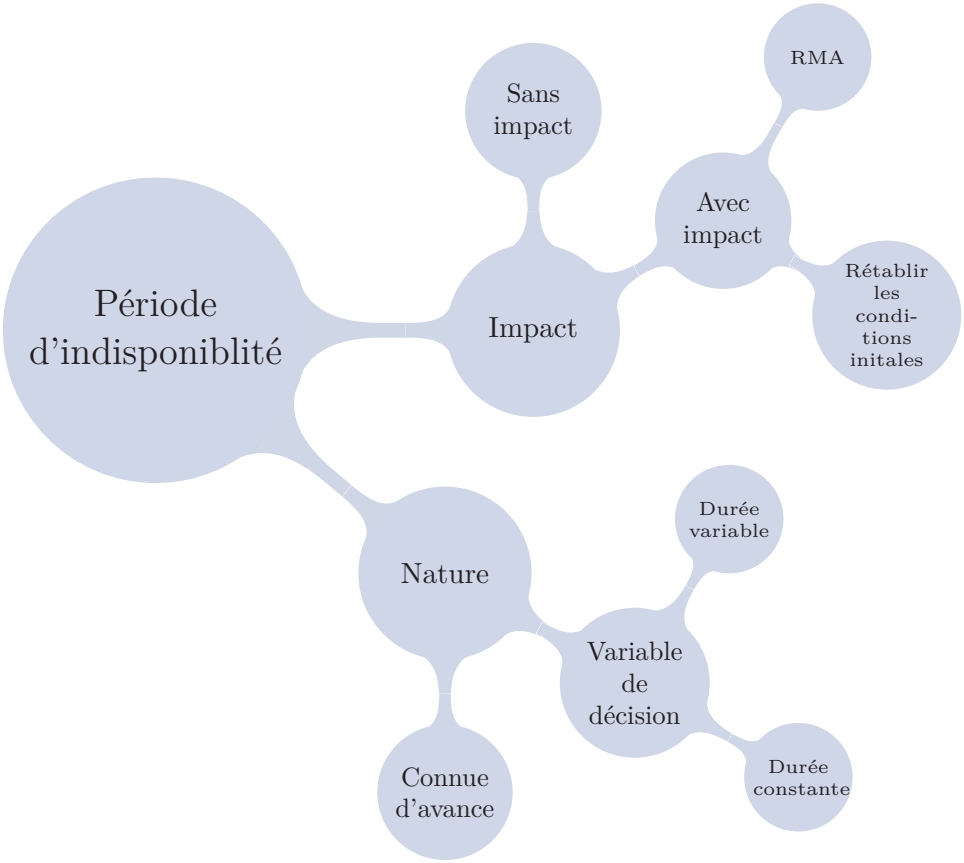

Figure 1.9 – Prise en compte des indisponibilités dans les problèmes d'ordonnancement : schéma récapitulatif

1.5 Conclusion

Dans ce chapitre, nous avons présenté les notions de base d'un problème d'ordonnancement et les notations liées aux périodes d'indisponibilité. D'autre part, nous avons exposé les effets appliqués sur les ressources telles que l'effet d'apprentissage et l'effet de détérioration. Ensuite, nous avons présenté une analyse sur l'insertion des périodes d'indisponibilité dans les problèmes d'ordonnancement ainsi que les différents modèles proposés.

Afin de mieux positionner notre travail, le chapitre suivant consiste en un état de l'art sur les problèmes en présence de périodes d'indisponibilité, et en particulier, celles qui sont considérées comme des variables de décision.

2

État de l'art

Résumé : Ce chapitre est dévolu à retracer l'état de l'art des problèmes d'ordonnancement en présence de périodes d'indisponibilité. En effet, deux classes ont été étudiées dans la littérature. La première classe considère des périodes d'indisponibilité fixes (les dates de début et fin sont connues d'avance), tandis que la deuxième classe considère les périodes d'indisponibilité comme des variables de décision. Nous nous focalisons en particulier dans cette thèse sur la deuxième classe de problèmes.

Nous présentons également les résultats des travaux antérieurs avec les différents ateliers tels que les problèmes à une machine, machines parallèles et les problèmes flow shop et open shop. Enfin, nous positionnons nos travaux par rapport à la littérature existante.

Sommaire

2.1 Introduction

L'objectif de ce chapitre est non seulement de retracer l'état de l'art des problèmes d'ordonnancement avec la prise en compte de périodes d'indisponibilité, mais aussi d'identifier les écarts entre les différents travaux et de bien positionner notre étude par rapport à la littérature existante.

Afin de comparer les problèmes étudiés dans la littérature, nous définissons dans la section suivante 2.2 la méthodologie de recherche et les critères utilisés pour la classification des travaux antérieurs. Puis, nous présentons dans les sections 2.3 et 2.4 l'état de l'art. Enfin, nous positionnons nos travaux par rapport à la littérature existante.

2.2 Méthodologie et classification des problèmes

Dans notre étude bibliographique, nous avons fait des recherches selon les mots-clés suivants [1] : *scheduling, maintenance, rate-modifying activities, unavailability constraint, deteriorating, learning, time-dependent maintenance*. Notons que dans cette analyse de l'état d'art, nous nous focalisons en particulier sur les problèmes d'ordonnancement dont la période d'indisponibilité est une variable de décision et nous excluons les problèmes avec une fenêtre d'indisponibilité (*i.e.* cas déterministe : la période d'indisponibilité est connue d'avance).

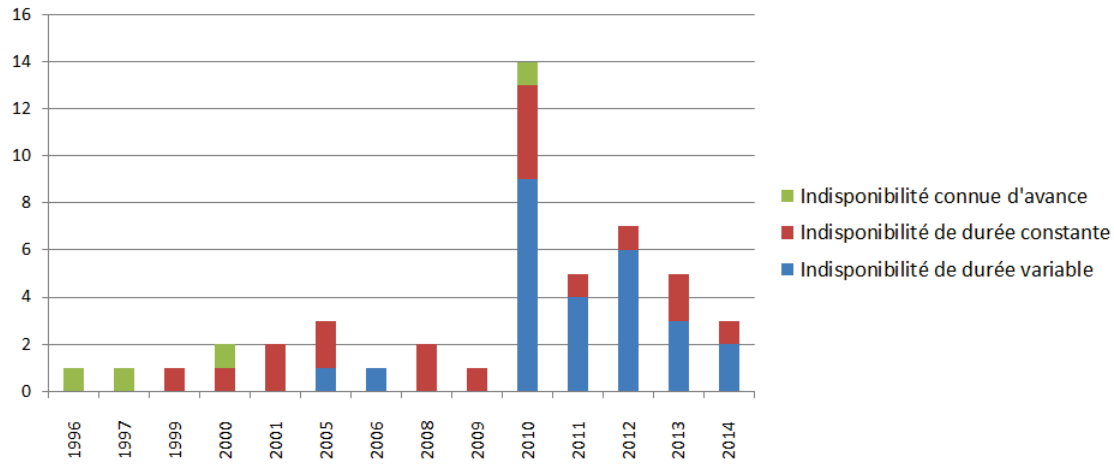

Figure 2.1 – Nombre d'articles par an

Historiquement, comme le montre la figure 2.1, trois dates-clés caractérisent les études sur les problèmes d'ordonnancement en présence de périodes d'indisponibilité. En effet, jusqu'au milieu des années 80, les machines étaient considérées comme des ressources disponibles. En 1985, Dolev et Warmuth [27, 28] ont été les premiers à introduire une

1. Les recherches ont été réalisées via des moteurs de recherche tels que : ScienceDirect, Google Scholar.

fenêtre d'indisponibilité. Nous citons aussi les travaux de Liu et Sanlaville [77]. En 1996, Lee [72, 73] a étudié plusieurs configurations de problèmes en présence d'une fenêtre d'indisponibilité. Cet axe de recherche a attiré de plus en plus l'attention des chercheurs, voir les survey de Schmidt [102] en 2000 et de Ma *et al.* [81] en 2010. Notons que dans la figure 2.1, nous avons mentionné seulement les références clés (les publications de Lee [72, 73] et les deux survey [81, 102]) pour les problèmes avec des indisponibilités connues d'avance.

Les problèmes d'ordonnancement simultané de tâches et d'indisponibilités, *i.e.* la période d'indisponibilité est une variable de décision, ont été introduits par Qi *et al.* [95] en 1999 et par Lee et Leon [75] en 2001, en étudiant une ligne d'assemblage électronique. Les durées de ces périodes sont constantes. La troisième date importante marque les premières études menées par Kubzin et Strusevich [67] sur un modèle avec une période de maintenance considérée comme une variable de décision et de durée variable en 2005. Les auteurs ont été les premiers à étudier des indisponibilités avec des durées variables. Ainsi, nous pouvons résumer l'ordre chronologique d'études menées sur les problèmes d'ordonnancement en présence de périodes d'indisponibilité par la figure 2.2.

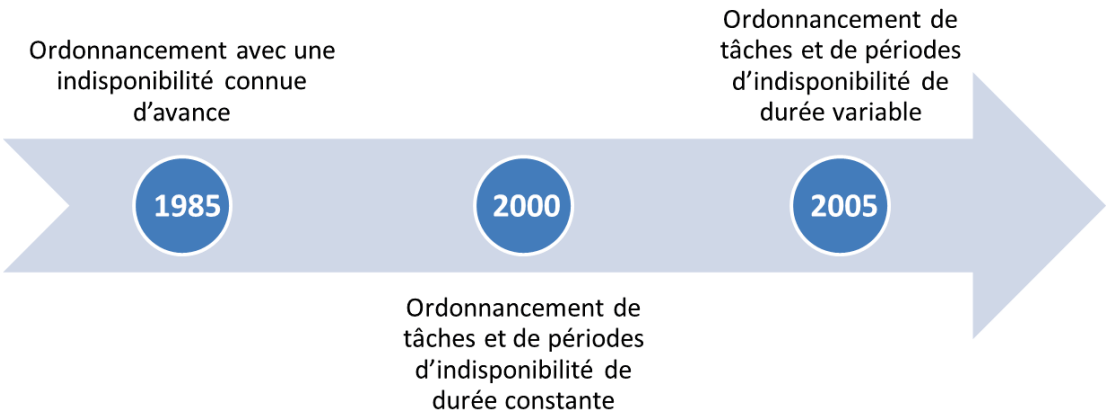

Figure 2.2 – Dates clés des études sur les problèmes d'ordonnancement en présence de périodes d'indisponibilité

Dans les problèmes d'ordonnancement en présence de périodes d'indisponibilité, plusieurs critères de classification peuvent être définis, nous présentons dans ce qui suit une liste des critères classés dans l'ordre d'importance décroissant :

– **Connue d'avance / Variable de décision** : Comme nous l'avons dit plus tôt, les problèmes peuvent être classés selon le type de l'indisponibilité. Si la période d'indisponibilité est fixée en avance (fenêtre d'indisponibilité sur la machine), l'objectif est d'ordonnancer les tâches sur les machines en tenant compte de la fenêtre d'indisponibilité. Lorsque la période d'indisponibilité est une variable de décision, le problème est donc un problème d'ordonnancement simultané des tâches et des

périodes d'indisponibilité. Notons que dans cette thèse nous nous focalisons sur les problèmes d'ordonnancement dont la période d'indisponibilité est une variable de décision.

- **Durée d'indisponibilité variable / constante** : Nous pouvons aussi classer les problèmes selon la durée de la période d'indisponibilité. Le modèle de période d'indisponibilité de durée variable a été introduit en 2005 par Kubzin et Strusevich [67, 68].
- **Impact de la période d'indisponibilité** : Les problèmes peuvent être classés selon l'impact de la maintenance sur la machine. Nous pouvons avoir plusieurs types d'indisponibilité :
  - des indisponibilités sans impact sur les machines. Nous citons, par exemple, les travaux de Lee [72, 73], Grigoriev *et al.* [46], Allaoui *et al.* [3].
  - des indisponibilités qui modifient les performances de la machine avec un taux δ ou plus généralement b_j et a_j, respectivement, la durée de la tâche T_j avant et après l'indisponibilité. Nous citons, par exemple, les travaux de Lee et Leon [75], He *et al.* [51], Hsu *et al.* [55], Zhao *et al.* [139], Yang et Yang [125]
  - des indisponibilités qui rétablissent les conditions initiales de la machine, lorsque la machine subit un effet de variation (détérioration, apprentissage...). Nous citons, par exemple, les travaux de Kuo et Yang [71] Zaho et Tang [138], Yang et Yang [134], Yang [126].

En nous basant sur ces critères de classification, nous présentons dans les deux sections suivantes les différents travaux menés sur les problèmes d'ordonnancement en présence de périodes d'indisponibilité.

2.3 Périodes d'indisponibilité connues d'avance

Depuis le début des années 1990, plusieurs travaux ont été menés sur les problèmes d'ordonnancement en présence de périodes d'indisponibilité connues d'avance, *i.e.* la durée et la position (date de début et date de fin) de la période d'indisponibilité sont fixes. Dans ce cas, la période d'indisponibilité est considérée comme un gap (fenêtre d'indisponibilité) sur la machine.

Lee a étudié dans [72] et [73] des problèmes d'ordonnancement avec une période d'indisponibilité connue d'avance sur plusieurs configurations (une machine, machines parallèles et flow shop). Nous renvoyons le lecteur intéressé aux surveys de Sanlaville et Schmidt [100], Ma *et al.* [81] et Schmidt [102] qui présentent une synthèse des articles qui considèrent des périodes d'indisponibilité connues d'avance.

Des problèmes avec des périodes de maintenance qui doivent être planifiées périodi-

quement (des périodes d'indisponibilité fixes chaque T unités de temps) ont été étudiés dans [46], [4], [17] et [101].

2.4 Périodes d'indisponibilité : variables de décision

Dans cette section, nous passons en revue, de manière synthétique, les différents problèmes d'ordonnancement qui intègrent des périodes d'indisponibilité considérées comme des variables de décision.

Les premiers travaux dans cet axe de recherche ont été menés à la fin des années 90, et en particulier avec les publications de Qi *et al.* [95] et Lee et Leon [75], respectivement, en 1999 et 2001. Généralement, l'insertion de ces périodes est accompagnée par un changement de la performance de la machine ou par une contrainte temporelle (par exemple une fenêtre de réalisation pour la maintenance). Ces périodes sont souvent considérées comme des périodes de maintenance. En effet, si la période de maintenance est insérée sans aucune contrainte ou impact sur la machine, le problème peut être facilement résolu en considérant la maintenance comme une tâche.

Dans ces problèmes d'ordonnancement simultané de tâches et de périodes d'indisponibilité le décideur a besoin de connaître la séquence de tâches, le nombre et les positions optimales de maintenances permettant la minimisation du critère d'optimisation.

Deux classes de problèmes ont été étudiées dans la littérature ; des périodes d'indisponibilité de durées fixes et des périodes d'indisponibilité de durées variables.

Nous nous intéressons dans la première partie aux problèmes d'ordonnancement dont la période d'indisponibilité est une variable de décision avec une durée constante. Ensuite, nous présentons les travaux sur les périodes d'indisponibilité avec durées variables.

2.4.1 Périodes d'indisponibilité de durées constantes

Les problèmes d'ordonnancement de tâches et de périodes d'indisponibilité de durées constantes ont été largement étudiés dans la littérature. Dans ces problèmes, les périodes d'indisponibilité peuvent être identiques ; l'équipe de maintenance réalise les mêmes activités de maintenance à chaque intervention. Lorsque les périodes de maintenance ne sont pas identiques, les durées des maintenances sont constantes, mais elles peuvent varier selon :

- le rang de la maintenance : par exemple, la durée de la $i^{\text{ème}}$ maintenance est égale β_i.
- la machine : par exemple, la durée de maintenance sur une machine M_l est égale à β_l. Dans ce cas, la durée de la période de maintenance est constante, mais chaque machine possède une durée d'intervention spécifique.

Nous pouvons aussi avoir des périodes de maintenance de durées constantes, mais qui varient selon la machine et le rang de la maintenance en même temps, par exemple la durée de la $i^{\text{ème}}$ période de maintenance sur la machine M_l est :

$$m_{li} = \beta_{li}$$

Dans ce qui suit, nous présentons les différents problèmes d'ordonnancement étudiés en présence d'une période d'indisponibilité constante dans les différents ateliers de production.

2.4.1.1 Problèmes à une machine

Dans cette partie, nous nous intéressons aux ateliers à une machine. Vu le caractère fondamental de ce type d'atelier, les problèmes à une machine sont largement étudiés dans la littérature.

a. Minimisation du C_{max}

Les premiers problèmes d'ordonnancement avec des périodes d'indisponibilité variables de décision ont été présentés dans les travaux de Lee et Leon [75] en 2001 avec l'apparition de la notion de RMA (Rate-Modifying-Activity, voir section 1.4.1). Les auteurs considèrent le problème $1|(p_j; \delta_j p_j), \mathcal{M} \leq 1, m = \beta|C_{max}$, où au plus une seule période de maintenance de durée constante β est à ordonnancer sur la machine. Comme cela a été précisé précédemment dans la section 1.4, $(p_j; \delta_j p_j)$ signifie qu'avant la période de maintenance le temps opératoire de la tâche T_j est égale à p_j, et $\delta_j p_j$ si la tâche est insérée après la maintenance, avec $\delta_j > 0$. Lee et Leon [75] ont montré que ce problème est polynomial et que sa résolution consiste à placer arbitrairement, avant la maintenance, les tâches dont les temps opératoires augmentent après la maintenance, *i.e.* $\delta_j \geq 1$. Le reste des tâches (l'ensemble $V = \{T_j : \delta_j < 1\}$) sera placé après la maintenance si le gain en durée obtenu par le déplacement de ces tâches $s = \sum_{j \in V} p_j(1 - \delta_j)$ est supérieur à la durée de la maintenance β, sinon la séquence optimale est obtenue sans période de maintenance.

Un modèle plus restreint a été étudié par He *et al.* [51]. Les auteurs ont étudié le même problème, mais avec un ensemble de dates de lancement prédéfinies s_i pour les maintenances. En effet, la période de maintenance ne peut être lancée que dans des instants s_i prédéfinis. Les auteurs considèrent deux types de maintenance : maintenance obligatoire et maintenance optionnelle (au plus, une période de maintenance est sur la machine). Ils ont montré que les deux problèmes sont NP-difficiles.

Depuis lors, plusieurs travaux ont été réalisés sur cet axe de recherche.

Plus récemment, les problèmes d'ordonnancement en présence de maintenances ont

été étudiés avec des tâches de durées variables selon la position $p(j,r)$ ou la date de début $p(j,t)$. Dans ces problèmes, la variation des durées des tâches modélise un effet de détérioration. Ainsi, l'insertion de la maintenance permet de contrer cette détérioration et rétablir les conditions initiales de la machine.

Kuo et Yang [71], ont étudié le problème $1|p(j,r) = p_j r^a, \mathcal{M}, m = \beta_0|C_{max}$, avec $a > 0$ et avec des maintenances identiques de durées β_0. Le nombre de maintenances est une variable de décision à déterminer. Pour résoudre le problème, les auteurs considèrent initialement un sous-problème $1|p(j,r) = p_j r^a, \mathcal{M} = k, m = \beta_0|C_{max}$, avec un nombre de maintenances fixe égal à k.

Lorsque le nombre de maintenances est fixe (égal à k), nous obtenons $k+1$ groupes de tâches. Kuo et Yang [71] ont montré que les groupes sont équilibrés (Propriété d'équilibrage – Group balance principle). Cette propriété indique que dans l'ordonnancement optimal, la différence entre le nombre de tâches dans deux groupes quelconques est au plus égale à une tâche. Ils ont montré également que les tâches sont ordonnancées dans les groupes par ordre décroissant de temps opératoires (LPT : longest processing time). Par conséquent, un algorithme de résolution du problème $1|p(j,r) = p_j r^a, \mathcal{M} = k, m = \beta_0|C_{max}$ de complexité $\mathcal{O}(n)$ a été développé qui consiste à trouver les tailles des groupes et placer les tâches dans l'ordre LPT une par une dans la plus petite position disponible. En essayant toutes les valeurs de k possibles, $1 \leq k \leq n$, les auteurs obtiennent une solution du problème $1|p(j,r) = p_j r^a, \mathcal{M}, m = \beta_0|C_{max}$ dans un temps $\mathcal{O}(n^2)$. Les auteurs ont étendu aussi ces résultats au modèle linéaire $p(j,r) = p_j + b_j r$ avec $b_j > 0$.

Zaho et Tang [138] ont étendu le modèle de Kuo et Yang [71], en utilisant un facteur de détérioration qui dépend de la tâche $a_j \geq 0$. Ainsi, le problème d'ordonnancement étudié est $1|p(j,r) = p_j r^{a_j}, \mathcal{M}, m = \beta_0|C_{max}$. Pour un nombre de maintenances fixe, les auteurs ont montré que la propriété d'équilibrage est toujours vérifiée, ce qui permet de déterminer la taille exacte de chaque groupe (k+1 groupes) dans l'ordonnancement optimal. Ensuite, pour déterminer la séquence optimale qui minimise le C_{max}, Zhao et Tang ont modélisé le problème $1|p(j,r) = p_j r^{a_j}, \mathcal{M} = k, m = \beta_0|C_{max}$ sous forme d'un problème d'affectation linéaire avec $p_j r^{a_j}$ le coût d'affectation d'une tâche T_j dans la position r. Alors le problème $1|p(j,r) = p_j r^{a_j}, \mathcal{M} = k, m = \beta_0|C_{max}$ est résolu avec une complexité $\mathcal{O}(n^3)$. En énumérant toutes les valeurs de k possibles, la solution optimale de $1|p(j,r) = p_j r^{a_j}, \mathcal{M}, m = \beta_0|C_{max}$ est obtenue en $\mathcal{O}(n^4)$.

Un modèle avec un effet de détérioration général et plusieurs périodes de maintenance a été présenté par Rustogi et Strusevich [99]. Dans ce modèle, les auteurs considèrent que la détérioration varie selon le rang et le groupe de tâches. $p^{(i)}(j,r)$ est le temps opératoire de la tâche T_j ordonnancée dans le rang r du groupe i (ensemble de tâches entre la maintenance $i-1$ et i). La durée de la $i^{\text{ème}}$ période de maintenance est $m_i = \beta_i$.

L'enjeu du problème $1|p^{(i)}(j,r),\mathcal{M},m_i=\beta_i|C_{max}$ est de trouver le nombre optimal de maintenances et la séquence optimale des tâches qui minimisent C_{max}. Le tableau 2.1 résume les différentes formes de détérioration considérées par les auteurs, on note par la fonction g le facteur de détérioration décroissant selon r.

Tableau 2.1 – Les différents modèles étudiés par Rustogi et Strusevich [99]

Temps opératoire	Durée de la maintenance	Complexité
$p(j,r)=p_jg(r)$	$m_i=\beta_i$	$\mathcal{O}(n^2)$
$p^{(i)}(j,r)=p_jg(i,r)$	$m_i=\beta_i$	$\mathcal{O}(n^2)$
$p^{(i)}(j,r)=p_jg(j,i,r)$	$m_i=\beta_i$	$\mathcal{O}(n^4)$

Motivés par les problèmes de la fatigue des opérateurs, Lodree *et al.* [78] ont étudié le problème suivant $1|p(j,t)=\alpha_jt_j,\mathcal{M}\leq 1,m=\beta_0|C_{max}$. Les tâches suivent un effet de détérioration linéaire selon la date de début de la tâche $p(j,t)=\alpha_jt_j$, avec t_j la date de début de la tâche et $\alpha_j\geq 1$ le facteur de détérioration. Dans ce problème, la période d'indisponibilité de durée β_0 représente une pause de repos, ainsi le système revient à son état initial après la période d'indisponibilité. Les auteurs ont montré que la solution est obtenue avec l'insertion de "la pause" au milieu de l'ordonnancement.

b. Minimisation de la somme des dates de fin

Les problèmes de minimisation de la somme des dates de fin avec la prise en compte de périodes d'indisponibilité ont été également largement étudiés à partir des années 2000. Nous nous intéressons dans cette sous-section à la présentation des problèmes de minimisation de $\sum C_j$ avec des périodes d'indisponibilité de durée constante.

Qi *et al.* [95] ont étudié un problème avec plusieurs périodes de maintenance identiques à ordonnancer sur une machine afin de minimiser la somme des dates de fin. Les auteurs imposent un temps d'exécution maximum entre deux maintenances successives égal à T. Notons que l'insertion de la maintenance n'a aucun impact sur la machine, le temps de traitement des tâches, avant et après les maintenances, est invariant. Le problème peut être défini par $1|\mathcal{M},m=\beta_0|\sum C_j$. Ils ont montré que dans l'ordonnancement optimal les tâches dans chaque groupe (entre deux maintenances successives) sont insérées dans l'ordre décroissant de p_j (SPT). Ensuite, les auteurs ont montré que le problème est NP difficile au sens fort. Trois heuristiques ont été proposées pour résoudre le problème.

Contrairement au problème précédent, Lee et Leon [75] ont étudié la minimisation de la somme des dates de fin avec au plus une période de maintenance qui modifie les performances de la machine. Nous présentons le problème d'ordonnancement sous la notation de Graham par $1|(p_j;\alpha_jp_j),\mathcal{M}\leq 1,m=\beta_0|\sum C_j$, avec $\alpha_j>0$. L'objectif est donc de trouver la séquence de tâches qui minimise la $\sum C_j$ et de trouver la position optimale de

la maintenance (si elle existe). Pour une position de maintenance fixe, les auteurs modélisent le problème sous forme d'un problème d'affectation linéaire résolue en $\mathcal{O}(n^3)$. Ainsi la solution optimale est obtenue, en énumérant les n positions possibles de la maintenance. D'où la complexité totale de l'algorithme proposé par Lee et Leon [75] : $\mathcal{O}(n^4)$.

He *et al.* [51] ont étudié un problème similaire. Deux modèles de maintenance ont été étudiés :

- une maintenance optionnelle, *i.e.* au plus une période de maintenance à insérer,
- une maintenance obligatoire à insérer dans l'ordonnancement.

L'objectif est de trouver la séquence de tâches et la position de la maintenance qui minimisent la somme des dates de fin. Notons que les dates d'insertion possibles de la maintenance sont connues d'avance. Les auteurs ont montré que le problème est NP-difficile.

Nous résumons l'ensemble des résultats sur les problèmes d'ordonnancement à une machine avec une maintenance constante dans le tableau 2.2. Nous indiquons pour chaque référence :

- l'impact de la maintenance sur la durée des tâches : durée invariante, effet appliqué sur la machine (plusieurs paramètres de variations : position, temps et groupe) ou RMA,
- le nombre de maintenances à insérer dans l'ordonnancement : un nombre fixe, une borne supérieure ou un nombre à déterminer avec l'ordonnancement,
- la complexité de résolution.

2.4.1.2 Problèmes à machines parallèles

Dans la littérature, les problèmes d'ordonnancement dans un atelier à une seule machine ont été largement étudiés. Néanmoins, l'atelier à machines parallèles est plus proche des problèmes réels rencontrés dans la pratique.

Nous nous intéressons dans cette sous-section aux problèmes d'ordonnancement simultané de tâches et de périodes de maintenance de durée constante dans le cadre des machines parallèles, et en particulier aux problèmes de la minimisation de la somme des dates de fin ($\sum C_j$) et de la charge totale des machines ($\sum_{l=1}^{m} C_{max}^l$).

a. Minimisation de la somme des dates de fin sur chaque machine

Les problèmes d'ordonnancement simultanés de tâches et de maintenances sur des machines parallèles ont été introduits, en 2000, par Lee et Chen [74]. Les auteurs considèrent m machines identiques. Chaque machine doit être maintenue une seule fois entre $[0, T]$.

Tableau 2.2 – Problèmes d'ordonnancement simultané des tâches et des périodes d'indisponibilité de durées constantes sur une machine

Référence		Durée des tâches					Nombre de maintenances			Complexité
		Invariante	Paramètres de variations			RMA	Fixe	Borne sup.	A déterminer	
			position	temps	groupe					
Minimisation du makespan C_{max}										
Lee et Leon	[75]					✓		≤ 1		$\mathcal{O}(n)$
He *et al.*	[51]					✓		≤ 1		*NP-difficile*
He *et al.*	[51]					✓	1			*NP-difficile*
Kuo et Yang	[71]		✓						✓	$\mathcal{O}(n^2)$
Zaho et Tang	[138]		✓						✓	$\mathcal{O}(n^4)$
Jr *et al.*	[78]			✓				≤ 1		$\mathcal{O}(n)$
Rustogi et Strusevich	[99]		✓						✓	$\mathcal{O}(n^2)$
Rustogi et Strusevich	[99]		✓		✓				✓	$\mathcal{O}(n^2)$
Rustogi et Strusevich	[99]		✓		✓				✓	$\mathcal{O}(n^4)$
Minimisation de $\sum C_j$										
Qi *et al.*	[95]	✓							✓	*NP-difficile*
Lee et Leon	[75]					✓		≤ 1		$\mathcal{O}(n^4)$
He *et al.*	[51]					✓		≤ 1		*NP-difficile*
He *et al.*	[51]					✓	1			*NP-difficile*

Deux cas de maintenance ont été étudiés :

Cas 1 : Les ressources de la maintenance sont limitées. Ainsi, au plus une seule machine peut être maintenue à un instant t donné, *i.e.* il n'y a pas de chevauchement de maintenance.

Cas 2 : Les ressources de la maintenance sont suffisantes. Ainsi si nécessaire, plusieurs machines peuvent être maintenues simultanément.

L'objectif du problème, étudié par Lee et Chen [74], est la minimisation de la somme pondérée des dates de fin, noté $Pm|\mathcal{M}_l = 1,\ m = \beta_0|\sum w_j C_j$. Les auteurs ont montré que le problème est NP-difficile dans les deux cas présentés précédemment. Une méthode de résolution basée sur une approche de séparation et évaluation (Branch & Bound) a été développée, permettant la résolution des instances de petite taille.

Zhao *et al.* [139] ont étendu le problème de Lee et Leon [75] à un atelier à deux machines identiques. Au plus, une période d'indisponibilité de durée constante β_l doit être insérée sur la machine M_l, avec $l = \{1, 2\}$. Le temps opératoire de la tâche T_j ordonnancée avant la période d'indisponibilité est p_j et $\delta_j p_j$ si la tâche T_j est ordonnancée après la période d'indisponibilité, avec $\delta_j > 0$ le taux de modification provoqué par la période d'indisponibilité. Les auteurs ont décomposé le problème en sous-problèmes. Ils ont montré que lorsque le nombre de tâches sur la première et sur la deuxième machine, ainsi que la position de la période d'indisponibilité sur les deux machines sont fixés, le problème peut être modélisé sous forme d'un problème d'affectation linéaire de complexité $\mathcal{O}(n^3)$. Alors, pour résoudre le problème initial $P2|(p_j, \delta_j p_j), \mathcal{M}_l \leq 1, m_l = \beta_l|\sum C_j$, il suffit de vérifier tous les sous-problèmes possibles, *i.e.* énumérer toutes les combinaisons possibles : nombres de tâches sur les machines, positions des indisponibilités. Notons que le nombre de combinaisons possibles est de $\mathcal{O}(n^3)$, le problème peut donc être résolu avec une complexité totale de $\mathcal{O}(n^6)$.

Une extension a été étudiée dans le même article, avec m machines identiques, le problème est toujours polynomial si $m > 2$, avec une complexité $\mathcal{O}(n^{2m+3})$. Les auteurs ont aussi proposé un algorithme pseudo-polynomial pour la minimisation de $\sum w_j C_j$.

Notons que, pour le problème étudié par Zhao *et al.* [139], il n'y a pas de restriction sur la valeur de δ_j. Autrement dit, on peut avoir un taux qui diminue la durée de la tâche ($\delta_j < 1$) ou augmente la durée ($\delta_j \geq 1$).

Hsu *et al.* [55] ont étendu le problème de Zhao *et al.* [139] aux machines parallèles non-liées, avec $0 < \delta_{lj} \leq 1$ le taux de modification de la durée de la tâche T_j sur la machine M_l, *i.e.* l'insertion de la période d'indisponibilité améliore les performances de la machine. L'objectif est la minimisation de la somme des dates de fin. Les auteurs ont montré que le problème est polynomial et ont proposé un algorithme de résolution avec une complexité $\mathcal{O}(n^{m+3})$.

En 2014, Yang *et al.* [132] ont amélioré les résultats obtenus par Hsu *et al.* [55]. Les auteurs ont montré que le problème $Rm|(p_{lj}, \delta_{lj} p_{lj}), \mathcal{M}_l \leq 1, m_l = \beta_l|\sum C_j$ peut être résolu avec une complexité $\mathcal{O}(n^{m+3})$ quel que soit le taux $\delta_{lj} > 0$.

Dans les travaux cités précédemment, les auteurs supposent qu'au plus une seule période de maintenance est autorisée sur une machine. Néanmoins, dans les cas réels de production, une machine peut avoir besoin d'être maintenue plusieurs fois afin d'améliorer son efficacité de production au cours de l'ordonnancement.

Yang et Yang [125] ont étudié le même problème que [132], mais avec plusieurs périodes d'indisponibilité de durée constante possibles sur chaque machine $Rm|(p_{lj}, \delta_{lji} p_{lj}), \mathcal{M}_l \leq k_l, m_l = \beta_l|\sum C_j$, avec $0 < \delta_{lji}$ le taux de modification des performances de la machine lié à la période d'indisponibilité numéro i sur la machine M_l. Au plus, k_l périodes de mainte-

nance sont autorisées sur la machine M_l. Si le nombre de machines m est fixe, les auteurs ont proposé un algorithme de résolution polynomial avec une complexité $\mathcal{O}(n^{3+\sum_{l=1}^{m} k_l})$.

Ji et Cheng [59] ont étudié la minimisation de la somme des dates de fin sur un problème d'ordonnancement à machines parallèles identiques avec un effet d'apprentissage exponentiel ($p(j,r) = p_j r^{a_j}$: le temps opératoire de la tâche T_j ordonnancée à la position r sur une machine et $a_j \leq 0$ le facteur d'apprentissage de la tâche). Les auteurs supposent que plusieurs périodes de maintenance de durée constante sont autorisées sur les machines, avec k_l le nombre de maintenances maximal autorisé sur la machine M_l et β_{li} la durée de la $i^{\text{ème}}$ période de maintenance sur la machine M_l. En se basant sur un problème d'affectation, Ji et Cheng [59] ont montré que le problème est polynomial avec une complexité $\mathcal{O}(n^{m+2+\sum_{l=1}^{m} k_l})$ avec un nombre de machines m fixe.

b. Minimisation de la charge totale des machines

Comme cela a été précisé précédemment, la minimisation de la charge totale des machines (TML) correspond à la minimisation de la somme des durées d'exécution des machines $\sum_{l=1}^{m} C_{max}^{l}$. Ce critère d'optimisation est un indicateur de performance pertinent pour les problèmes d'ordonnancement avec un effet d'apprentissage ou de détérioration. C'est pour cette raison que la minimisation de la charge totale des machines est souvent étudiée en présence d'un effet de détérioration ou d'apprentissage.

Nous nous intéressons dans ce paragraphe à la minimisation de la charge totale des machines en présence de périodes d'indisponibilité de durées constantes. Dans [59], Ji et Cheng ont étudié un problème d'ordonnancement sur m machines parallèles et identiques $Pm|(p(j,r), \delta_{lji}p(j,r)), \mathcal{M}_l \leq k_l, m_{li} = \beta_{li}|\sum C_j$ avec un effet d'apprentissage $p(j,r) = p_j r^{a_j}$ et au plus k_l périodes d'indisponibilité à ordonnancer sur la machine M_l où la durée de la $i^{\text{ème}}$ période sur M_l est β_{li}. L'insertion d'une période d'indisponibilité permet de modifier la performance de la machine. Ainsi, la durée de la tâche j ordonnancée après la période i sur la machine M_l a une durée d'exécution $\delta_{lji}p(j,r)$, avec $0 < \delta_{lji} \leq 1$: le taux de modification des performances de la machine après la $i^{\text{ème}}$ période d'indisponibilité. Les auteurs montrent que minimiser la charge totale de la machine (TML) peut être résolu en temps polynomial avec une complexité $\mathcal{O}(n^{m+2+\sum_{l=1}^{m} k_l})$.

Yang et Yang [125] ont étudié un problème similaire sur des machines parallèles non-liées avec plusieurs indisponibilités de durée constante β_l. La durée de la tâche T_j ordonnancée après la période i sur M_l a une durée d'exécution $\delta_{lji}p_{lj}$, avec $0 < \delta_{lji}$ et p_{lj} la durée standard de la tâche T_j sur la machine M_l. Les auteurs ont montré que le problème d'ordonnancement $Rm|\delta_{lji}p_{lj}, \mathcal{M}_l \leq k_l, m_l = \beta_l|\sum_{l=1}^{m} C_{max}^{l}$ est polynomial avec une complexité améliorée $\mathcal{O}(n^{3+\sum_{l=1}^{m} k_l})$.

Le problème d'ordonnancement avec un effet de détérioration sur des machines parallèles non-liées en présence de plusieurs périodes d'indisponibilité de durée constante β_l a été étudié dans l'article de Yang *et al.* [123]. Les auteurs considèrent deux modèles de détérioration : le premier suit une fonction exponentielle $p(l,j,r) = p_{lj}r^{a_{lj}}$ et le second modèle suit une fonction linéaire $p(l,j,r) = p_{lj}+rb_{lj}$, avec r la position de la tâche, $a_{lj} > 0$ et $b_{lj} > 0$ les facteurs de la détérioration de la tâche T_j ordonnancée sur la machine M_l. Dans ce problème, l'insertion d'une période de maintenance permet de rétablir les conditions initiales de la machine. L'objectif est de trouver la séquence optimale des tâches, les positions et le nombre de maintenances sur chaque machine qui minimisent la charge totale des machines ($TML = \sum_{l=1}^{m} C_{max}^{l}$). Yang *et al.* [123] ont montré que le problème $Rm|p(l,j,r), \mathcal{M}_l \leq k_l, m_l = \beta_l| \sum_{l=1}^{m} C_{max}^{l}$ est polynomial et ils ont proposé une méthode de résolution de complexité $\mathcal{O}(n^{m+3})$ pour les deux modèles présentés précédemment. Des cas particuliers ont aussi été étudiés dans [123] lorsque $p_{ljr} = p_{lj}r^a$ et $p_{ljr} = p_{lj} + rb$, un algorithme de résolution a été proposé avec une complexité $\mathcal{O}(n^{m+1}\, log\, n)$ pour les deux cas.

Dans le tableau 2.3, nous résumons les travaux à machines parallèles présentés précédemment. Pour chaque référence, nous indiquons : l'atelier de production, le critère d'optimisation, la variation des tâches, le nombre de maintenances et la complexité de résolution.

2.4.1.3 Problèmes flow shop

En 2010, Allaoui *et al.* [3] ont étudié un problème d'ordonnancement flow shop à deux machines avec une période de maintenance de durée constante à ordonnancer dans l'intervalle $[0, T]$. Les auteurs ont montré que le problème est NP-difficile si la période de maintenance est insérée sur la première ou la deuxième machine. Des propriétés d'optimalité et deux heuristiques de résolution ont été présentées. Nous présentons, dans le chapitre 7, une nouvelle analyse en démontrant que contrairement à ce qui est démontré dans [3], le problème lorsque la maintenance est à placer sur la deuxième machine est polynomial. Nous présentons aussi un algorithme de résolution et une extension du problème. Ces résultats ont été publiés dans l'article [31].

Tableau 2.3 – Problèmes d'ordonnancement simultané de tâches et de périodes d'indisponibilité de durées constantes sur machines parallèles

Référence		Atelier	Critère	Durée de tâches			Nombre de maintenances			Complexité
				Invariant	Position	RMA	Fixe	Borne sup.	A déterminer	
Lee et Chen	[74]	Pm	$\sum_{l=1}^{m} w_j C_j$	✓			1			*NP-difficile*
Zhao *et al.*	[139]	$P2$	$\sum_{l=1}^{m} C_j$			✓		1		$\mathcal{O}(n^6)$
Zhao *et al.*	[139]	Pm	$\sum_{l=1}^{m} C_j$			✓		1		$\mathcal{O}(n^{2m+3})$
Zhao *et al.*	[139]	Pm	$\sum_{l=1}^{m} w_j C_j$			✓		1		*NP-difficile*
Hsu *et al.*	[55]	Rm	$\sum_{l=1}^{m} C_j$			✓		1		$\mathcal{O}(n^{m+3})$
Yang *et al.*	[132]	Rm	$\sum_{l=1}^{m} C_j$			✓		1		$\mathcal{O}(n^{m+3})$
Yang et Yang	[132]	Rm	$\sum_{l=1}^{m} C_j$			✓		k_l		$\mathcal{O}(n^{3+\sum_{l=1}^{m} k_l})$
Ji et Cheng	[59]	Pm	$\sum_{l=1}^{m} C_j$		✓	✓		k_l		$\mathcal{O}(n^{2+m+\sum_{l=1}^{m} k_l})$
Ji et Cheng	[59]	Pm	$\sum_{l=1}^{m} C_{max}^{l}$		✓	✓		k_l		$\mathcal{O}(n^{2+m+\sum_{l=1}^{m} k_l})$
Yang et Yang	[125]	Rm	$\sum_{l=1}^{m} C_{max}^{l}$			✓		k_l		$\mathcal{O}(n^{3+\sum_{l=1}^{m} k_l})$
Yang *et al.*	[123]	Rm	$\sum_{l=1}^{m} C_{max}^{l}$		✓			k_l		$\mathcal{O}(n^{m+3})$
Yang *et al.*	[123]	Rm	$\sum_{l=1}^{m} C_{max}^{l}$		✓			k_l		$\mathcal{O}(n^{m+1} \log n)$

2.4.2 Périodes d'indisponibilité de durées variables

Les études présentées dans la section précédente considèrent que les périodes d'indisponibilité sont de durées constantes. En 2005, un nouveau modèle a été proposé par Kubzin et Strusevich [67, 68]. Dans cet article, la durée de la maintenance suit une fonction linéaire selon sa date de début :

$$m(t) = \alpha \times t + \beta$$

avec t la date de début de la période d'indisponibilité mesurée à partir de la date de fin de la période d'indisponibilité précédente, α et β étant deux paramètres positifs. Dans ce modèle, plus la période séparant deux maintenances est importante, plus la maintenance à réaliser est lourde et donc plus sa durée est importante.

Depuis lors, l'ordonnancement sous contraintes de disponibilité de durée variable attire de plus en plus l'attention des chercheurs.

Dans les sous-sections suivantes, nous passons en revue les problèmes d'ordonnancement en présence de périodes d'indisponibilité de durées variables.

2.4.2.1 Problèmes à une machine

a. Minimisation du C_{max}

Nous présentons dans ce qui suit, les principaux résultats des problèmes d'ordonnancement intégrant la minimisation du makespan en présence de périodes d'indisponibilité de durées variables sur une machine.

Nous commençons par un problème basique étudié par Mosheiov et Sidney [90]. Les auteurs considèrent une machine, n tâches indépendantes à ordonnancer avec b_j le temps opératoire de la tâche T_j, une période de maintenance à ordonnancer, la durée de l'activité de maintenance varie en fonction de sa date de début $m = \alpha t + \beta$. L'insertion de la maintenance améliore la performance de la machine : si la tâche T_j est ordonnancée après la maintenance le temps opératoire devient a_j $(a_j \leq b_j)$. L'objectif est la minimisation du C_{max}. La résolution de ce problème est triviale. En effet, dans l'ordonnancement optimal deux cas sont possibles ; soit, l'activité de maintenance est insérée à l'instant $t = 0$, ainsi la durée de la maintenance est minimisée $(m = \beta)$ et la somme des temps opératoires est égale à $\sum_{j=1}^{n} a_j$. Ou la maintenance n'est pas effectuée du tout. Dans les deux cas, l'ordre des travaux est sans importance. La solution optimale est donc le makespan minimum de ces deux cas. Ainsi, le problème d'ordonnancement $1|p(j) = (b_j, a_j), \mathcal{M} \leq 1, m = \alpha t + \beta|C_{max}$ peut être résolu à l'optimal avec une complexité $\mathcal{O}(n)$, avec $C^*_{max} = min\{\sum b_j; \sum a_j + \beta\}$.

Yang et Yang [134] se sont intéressés au problème à une machine avec un effet de

détérioration $1|p(j,r) = p_j r^{a_j}, \mathcal{M}, m_i = \alpha t_i + \beta|C_{max}$. Le temps opératoire de la tâche T_j est égal à $p(j,r) = p_j r^{a_j}$. L'objectif est de trouver simultanément la séquence des tâches qui minimise le C_{max}, le nombre et les positions optimales de maintenances. Les auteurs ont considéré des périodes de maintenance qui varient en fonction de leurs dates de début $m_i = \alpha t_i + \beta$, t_i étant la durée entre la $(i-1)^{éme}$ maintenance et la $i^{\text{ème}}$ maintenance. En utilisant un algorithme basé sur le problème d'affectation, les auteurs ont prouvé que le problème est polynomial avec une complexité $\mathcal{O}(n^5)$.

Lorsque la durée de la maintenance suit un effet d'apprentissage tel que $m_i = t_0 i^b$, avec $b < 0$ le facteur d'apprentissage et $t_0 > 0$ la durée élémentaire de la maintenance, les auteurs ont montré que la minimisation du makespan est polynomiale avec un algorithme de résolution de complexité $\mathcal{O}(n^4)$. Si le facteur de la détérioration est indépendant de la tâche $a_j = a$, les auteurs ont montré que le problème $1|p(j,r) = p_j r^a, \mathcal{M}, m_i = t_0 i^b|C_{max}$ peut être résolu avec une complexité $\mathcal{O}(n \log n)$.

Yang [126] a étudié un problème à une machine avec un modèle qui combine l'effet de détérioration et d'apprentissage. La durée de la tâche T_j varie en fonction de sa position et de sa date de début $p(j,r,t) = (p_j - at)r^b$, où $0 < a < 1$ et $b \geq 0$ sont, respectivement, le facteur d'apprentissage et de détérioration. Au plus, une période de maintenance doit être insérée sur la machine et qui varie en fonction de sa date de lancement $m = \alpha t + \beta$. Les auteurs ont prouvé que la résolution du problème est polynomiale avec une complexité $\mathcal{O}(n^2 logn)$. Dans [128], Yang a étudié un problème similaire au problème précédent avec plusieurs périodes de maintenance à insérer. Vu le budget limité, le nombre maximum de maintenances autorisées est égal à k_0. L'objectif du problème $1|p(j,r,t) = (p_j - at)r^b, \mathcal{M} \leq k_0, m_i = \alpha t_i + \beta|C_{max}$ est de trouver la séquence optimale qui minimise C_{max}, le nombre et les positions optimales de maintenances. Yang a développé une méthode de résolution énumérative qui résout le problème d'une façon optimale avec une complexité $\mathcal{O}(n^{k_0+1} \log n)$.

Un modèle avec un effet de détérioration général et plusieurs périodes de maintenance a été présenté par Rustogi et Strusevich [99]. Dans ce modèle avec plusieurs périodes de maintenance, les auteurs considèrent que la détérioration varie selon le rang et le groupe de tâches, autrement dit, l'efficacité de l'intervention de maintenance varie d'une maintenance à une autre. $p^{(i)}(j,r)$ est le temps opératoire de la tâche T_j ordonnancée dans le rang r du groupe i (ensemble de tâches entre la maintenance i et $(i-1)$. La durée de la $i^{\text{ème}}$ période de maintenance est $m_i = \alpha_i t_i + \beta_i$ avec α_i et β_i les paramètres de la $i^{\text{ème}}$ maintenance. L'objectif du problème $1|p^{(i)}(j,r), \mathcal{M}, m_i = \alpha_i t_i + \beta_i|C_{max}$ est de trouver le nombre optimal de maintenances et la séquence optimale de tâches qui minimisent C_{max}. Le tableau 2.4 résume les différentes formes de détérioration considérées par les auteurs, on note par la fonction g le facteur de détérioration décroissant selon r.

Tableau 2.4 – Les différents modèles étudiés dans Rustogi *et al.* [99]

Temps opératoire	Durée de la maintenance	Complexité
$p^{(i)}(j,r) = p_j g(r)$	$m_i = \alpha_i t + \beta_i$	$\mathcal{O}(n^2)$
$p^{(i)}(j,r) = p_j g(i,r)$	$m_i = \alpha_i t + \beta_i$	$\mathcal{O}(n^3)$
$p^{(i)}(j,r) = p_j g(j,r)$	$m_i = \alpha_i t + \beta_i$	$\mathcal{O}(n^4)$
$p^{(i)}(j,r) = p_j g(i,j,r)$	$m_i = \alpha_i t + \beta_i$	$\mathcal{O}(n^5)$

Nous résumons dans le tableau 2.5 les différents travaux de minimisation du C_{max} en présence de périodes de maintenance avec durées variables. La première colonne du tableau regroupe les références. La deuxième colonne précise les variations appliquées sur les tâches. Le nombre de maintenances à insérer et la complexité de résolution sont indiqués, respectivement, dans les deux dernières colonnes.

Tableau 2.5 – Problèmes d'ordonnancement simultané des tâches et des périodes d'indisponibilité de durées variables sur une machine

Référence		Durée des tâches				Nombre de maintenances			Complexité
		Paramètres de variation							
		position	temps	groupe	RMA	Fixe	Borne sup.	A déterminer	
Mosheiov et Sidney	[90]				✓		≤ 1		$\mathcal{O}(n)$
Yang et Yang	[134]	✓						✓	$\mathcal{O}(n^5)$
Yang	[126]	✓	✓				≤ 1		$\mathcal{O}(n^2 \log n)$
Yang	[128]	✓	✓				$\leq k_0$		$\mathcal{O}(n^{k_0+1} \log n)$
Rustogi et Strusevich	[99]	✓						✓	$\mathcal{O}(n^2)$
Rustogi et Strusevich	[99]	✓		✓				✓	$\mathcal{O}(n^3)$
Rustogi et Strusevich	[99]	✓						✓	$\mathcal{O}(n^4)$
Rustogi et Strusevich	[99]	✓		✓				✓	$\mathcal{O}(n^5)$

b. Minimisation de la somme des dates de fin $\sum C_j$

Mosheiov et Sidney [90] ont considéré le problème d'ordonnancement suivant : $1|p(j) = (a_j, b_j), \mathcal{M} \leq 1, m = \alpha t + \beta| \sum C_j$. Une période de maintenance variable est à insérer dans l'ordonnancement. La durée de la maintenance varie en fonction de sa date de début $m = \alpha t + \beta$. L'insertion de l'activité de maintenance permet d'améliorer la

performance de la machine, avec a_j et b_j le temps opératoire de la tâche, respectivement, avant et après la maintenance tel que $b_j \leq a_j$. Le problème a été résolu avec une complexité $\mathcal{O}(n^4)$ en utilisant un problème d'affectation.

Yang [126] a proposé une méthode de résolution basée sur un algorithme de tri de complexité $\mathcal{O}(n^2 \log n)$ pour minimiser la somme des dates de fin en présence d'une période de maintenance optionnelle de durée variable $m = \alpha t + \beta$. La planification d'une période de maintenance permet de rétablir les conditions initiales du système de production. La machine subit un effet de détérioration et d'apprentissage avec $p(j,r) = (p_j - at)r^b$ avec $0 < a < 1$ et $b \geq 0$, respectivement, les facteurs d'apprentissage et de détérioration. Notons qu'une restriction sur ces paramètres a été ajoutée par les auteurs pour avoir des temps opératoires positifs. Nous présentons le problème étudié sous la notation à trois champs par $1|p(j,r) = (p_j - at)r^b, \mathcal{M} \leq 1, m = \alpha t + \beta| \sum C_j$.

Le cas des problèmes avec plusieurs périodes de maintenance a été étudié par Yang et Yang [135]. Les auteurs considèrent le problème $1|p(j,r), \mathcal{M} \leq k_0, m_i = \alpha t_i + \beta| \sum C_j$. Yang et Yang ont étudié trois modèles de détérioration, à savoir :

- Modèle de détérioration exponentiel : $p(j,r) = p_j r^{a_j}$, où $a_j > 0$ le facteur de détérioration de la tâche T_j.
- Modèle de détérioration linéaire : $p(j,r) = p_j + a_j r$, où $a_j > 0$ le facteur de détérioration de la tâche T_j.
- Modèle de détérioration temporel et linéaire : $p(j,r) = p_j + at$, où $a > 0$ et t sont, respectivement, le facteur de détérioration commun à toutes les tâches et la date de début de la tâche.

En raison de la restriction du budget d'entretien, le nombre maximum de maintenances k_0 est supposé connu à l'avance. L'objectif est de minimiser la somme des dates de fin et de trouver le nombre et les positions optimales de maintenances. Nous résumons dans le tableau 2.6 les trois problèmes étudiés ainsi que la complexité de résolution. Notons qu'un cas particulier du modèle linéaire a été étudié avec $a_j = a$, *i.e.* un facteur de détérioration indépendant des tâches (deuxième ligne du tableau). L'ensemble des méthodes de résolution présentées dans l'article est basé sur la résolution d'un problème d'affectation.

Tableau 2.6 – Les différents modèles étudiés dans Yang et Yang [135]

Effet sur la machine	Durée de la maintenance	Complexité
$p(j,r) = p_j r^{a_j}$	$m_i = \alpha t_i + \beta$	$\mathcal{O}(n^{k_0+3})$
$p(j,r) = p_j r^a$	$m_i = \alpha t_i + \beta$	$\mathcal{O}(n^{k_0+1}\, log\, n)$
$p(j,r) = p_j + b_j r$	$m_i = \alpha t_i + \beta$	$\mathcal{O}(n^{k_0+3})$
$p(j,r) = p_j + at$	$m_i = \alpha t_i + \beta$	$\mathcal{O}(n^{k_0+1}\, log\, n)$

Dans [128], un problème similaire au précédent a été étudié avec plusieurs périodes de maintenance à ordonnancer : $1|p(j,r,t) = (p_j + \delta t)r^a, \mathcal{M} \leq k_0, m_i = \alpha t_i + \beta| \sum C_j$. Les auteurs considèrent l'effet combiné de détérioration/apprentissage suivant $p(j,r,t) = (p_j + \delta t)r^a$, avec $a \leq 0$ le facteur d'apprentissage et $\delta \geq 0$ le facteur de détérioration. Le nombre maximum de maintenances k_0 est connu d'avance. L'objectif est de trouver la séquence optimale qui minimise $\sum C_j$, le nombre et les positions optimales de maintenances. Les auteurs ont montré que lorsque le nombre de maintenances à insérer et les tailles des groupes (nombre de tâches entre deux maintenances) sont connus, le problème peut être résolu avec un algorithme de tri de complexité $\mathcal{O}(n \log n)$. Ainsi, pour résoudre le problème initial, il suffit d'énumérer toutes les tailles possibles de groupes et tous les nombres de maintenances, ils ont montré que $\mathcal{O}(n^{k_0})$ combinaisons sont possibles. Alors, la complexité totale pour résoudre le problème est de $\mathcal{O}(n^{k_0+1}logn)$.

Nous résumons dans le tableau 2.7 les différents travaux menés sur la minimisation de $\sum C_j$ en présence de périodes de maintenance avec durées variables. Dans ce tableau, plusieurs caractéristiques du problème étudié sont présentées : les paramètres de variation des tâches, le nombre de maintenances à insérer et la complexité de résolution.

Tableau 2.7 – Problèmes d'ordonnancement simultané des tâches et des périodes d'indisponibilité de durées variables sur une machine

Référence		Durée de tâches			Nombre de maintenances	Complexité
		Position	Temps	RMA		
Mosheiov et Sidney	[90]			✓	≤ 1	$\mathcal{O}(n^4)$
Yang	[126]	✓	✓		≤ 1	$\mathcal{O}(n^2 \log n)$
Yang et Yang	[135]	✓			$\leq k_0$	$\mathcal{O}(n^{k_0+3})$
Yang et Yang	[135]	✓			$\leq k_0$	$\mathcal{O}(n^{k_0+1} \log n)$
Yang et Yang	[135]		✓		$\leq k_0$	$\mathcal{O}(n^{k_0+1} \log n)$
Yang	[128]	✓	✓		$\leq k_0$	$\mathcal{O}(n^{k_0+1} \log n)$

c. Minimisation des pénalités d'avance et de retard

Dans ce paragraphe, nous nous focalisons sur les problèmes avec pour objectif la minimisation de la somme des pénalités d'avance et de retard. Cette configuration est définie comme suit : la date de fin d'exécution de chaque tâche doit être incluse dans une fenêtre de réalisation $[d; f]$. d (≥ 0) et f $(\geq d)$ sont des paramètres du problème. Si la tâche est réalisée avant cet intervalle, elle doit être stockée, une pénalité sera donc appliquée.

Une pénalité de retard est appliquée si la tâche est réalisée après l'instant f. Le critère d'optimisation est le suivant :

$$Z = \sum_{j=1}^{n} (\delta E_j + \gamma T_j + \mu d + \vartheta D)$$

$E_j = \max\{0, d - C_j\}$ et $T_j = \max\{0, C_j - f\}$ et D sont, respectivement, l'avance de la tâche j (earliness), le retard absolu (tardiness) et la taille de l'intervalle ($D = f - d$). $\delta > 0, \gamma > 0, \mu > 0$ et $\vartheta > 0$ respectivement les coûts d'avance, de retards, la date de début de l'intervalle de réalisation et la taille de l'intervalle de réalisation par unité de temps. Pour plus de détails sur les problèmes d'ordonnancement avec fenêtres de réalisations, nous renvoyons le lecteur à Gordon *et al.* [43] et à la bibliographie de l'article [131] .

Ce critère d'optimisation est important dans la gestion des systèmes industriels modernes, tels que le Juste-À-Temps (JIT) et la gestion de la chaîne logistique. En effet, dans le système de production juste-à-temps, les tâches ne doivent pas être terminées trop tôt ou trop tard, sinon, des pénalités de retard et d'avance sont appliquées.

Mosheiov et Sidney [90] se sont intéressés au problème $1|p(j) = (b_j, a_j), \mathcal{M} \leq 1, m = \alpha t + \beta|Z$ avec $Z = \sum_{j=1}^{n}(\delta E_j + \gamma T_j + \mu d)$. Dans cet article les auteurs considèrent que $d = f$. Une maintenance optionnelle de durée variable (qui varie en fonction de sa date de début) permet de modifier la performance de la machine (b_j et a_j représentent les durées d'exécution des tâches respectivement avant et après la maintenance tel que $a_j \leq b_j$). Les auteurs ont démontré, à l'aide d'un programme linéaire d'affectation, que la résolution de ce problème est polynomiale avec une complexité $\mathcal{O}(n^4)$.

Dans [24], Cheng *et al.* ont considéré le problème $1|p(j,t) = a_j + bt, \mathcal{M} \leq 1, m = \alpha t + \beta|Z$, problème à une machine avec un effet de détérioration qui varie en fonction de la date de début des tâches $p(j,t) = a_j + bt$, où $b > 0$ le facteur de détérioration. L'insertion d'une période de maintenance optionnelle permet de rétablir les conditions initiales du système. La maintenance varie en fonction de sa date de lancement $m = \alpha t + \beta$. L'objectif de cette étude est de trouver la position optimale de la maintenance, l'intervalle de réalisation et la minimisation de la fonction coût Z. Les auteurs ont proposé une méthode de résolution avec une complexité $\mathcal{O}(n^2 \log n)$ basée sur un algorithme de tri.

Dans [131], Yang *et al.* ont étudié un problème d'ordonnancement avec un effet de détérioration appliqué sur la machine. La durée de la tâche est déterminée par la fonction suivante $p(j,r) = p_j r^{a_j}$ avec $a_j \geq 0$: le facteur de détérioration. Une période de maintenance optionnelle de durée variable $m = \alpha t + \beta$ permet de réinitialiser la machine. Les auteurs ne considèrent pas un intervalle de réalisation, *i.e.* $D = 0$, mais une date de réalisation au plus tard $d_j = p(j,r) + q$ avec q une variable de décision à déterminer lors de la résolution. Yang *et al.* [131] ont prouvé, en utilisant un problème d'affectation,

que ce problème est polynomial avec une complexité $\mathcal{O}(n^4)$. Un cas particulier a été étudié avec $a_j = a$, les auteurs ont montré que dans ce cas le problème est de complexité $\mathcal{O}(n^2 \log n)$. Une extension du problème précédent ([131]) a été étudiée par les mêmes auteurs dans [133], mais avec un intervalle de réalisation tel que $[d, f]$. Yang *et al.* [133] ont démontré que la résolution du problème est aussi polynomiale avec une complexité $\mathcal{O}(n^2 \log n)$.

Dans [84], Mor et Mosheiov ont étudié le problème $1|p(j,r), \mathcal{M} \leq 1, m|Z$ avec plusieurs configurations. Les auteurs ont montré que ces problèmes (voir tableau 2.8 ci-dessous) sont polynomiaux. Des algorithmes basés sur le problème d'affectation ont été développés avec une complexité $\mathcal{O}(n^4)$.

Tableau 2.8 – Les différents modèles étudiés dans Mor *et al.* [84]

Effet sur la machine $p(j,r)$	Durée de la maintenance	Complexité
$(p(j), \theta_j p(j))$	$m = \alpha t + \beta$	$\mathcal{O}(n^4)$
$(p(j,r), \theta_j p(j,r))$	$m = \alpha t + \beta$	$\mathcal{O}(n^4)$
$(p(j), \theta_j p(j))$	$m_r = \beta_r$	$\mathcal{O}(n^4)$
$(p(j,r), \theta_j p(j,r))$	$m_r = \beta_r$	$\mathcal{O}(n^4)$

Notons que pour ces problèmes la période de maintenance améliore les performances de la machine, *i.e.* $0 \leq \theta_j \leq 1$. $m_r = \beta_r$ représente la durée de la maintenance ordonnancée avant la tâche qui se trouve dans la position r avec $\beta_r \leq \beta_{r+1}$.

Plus récemment, Xue *et al.* [122] ont considéré un problème à une machine en présence de plusieurs périodes de maintenance de durée variable $m_i = \alpha t_i + \beta$. La machine doit être maintenue une fois, chaque k tâches exécutées, pour rétablir ses conditions initiales. Un modèle de variation général a été appliqué sur les tâches : $p(j,r) = f_j(p_j, r)$. L'objectif est la minimisation de $\varepsilon C_{max} + \sum_{j=1}^{n}(\delta E_j + \gamma T_j)$ avec ε : les frais de production par unité de temps. Ils ont montré que le problème est polynomial avec une complexité de $\mathcal{O}(n^4)$.

Le tableau 2.9 résume les différents travaux menés sur la minimisation des pénalités d'avance et de retard en présence de périodes de maintenance avec durées variables.

d. Autres critères

D'autres critères d'optimisation ont été étudiés sur des problèmes à une machine. Dans [90], Mosheiov et Sidney ont considéré le problème d'ordonnancement suivant : $1|p(j) = (b_j, a_j), \mathcal{M} \leq 1, m = \alpha t + \beta|\gamma$ avec $\gamma = \{\sum U_j, L_{max}\}$, une période de maintenance avec un effet de détérioration qui varie en fonction de sa date de début $m = \alpha t + \beta$.

Tableau 2.9 – Minimisation des pénalités d'avance et de retard en présence de périodes de maintenance avec durées variables

Référence		Durée des tâches			Nombre de maintenances		Complexité
		Paramètres de variation		RMA	Borne sup.	A déterminer	
		Position	Temps				
Mosheiov et Sidney	[90]			✓	≤ 1		$\mathcal{O}(n^4)$
Cheng *et al.*	[24]		✓		≤ 1		$\mathcal{O}(n^2 \log n)$
Yang *et al.*	[131]	✓			≤ 1		$\mathcal{O}(n^4)$
Yang *et al.*	[133]	✓			≤ 1		$\mathcal{O}(n^2 \log n)$
Mor et Mosheiov	[84]	✓		✓	≤ 1		$\mathcal{O}(n^4)$
Xue *et al.*	[122]	✓		✓		✓	$\mathcal{O}(n^4)$

L'insertion de la maintenance permet d'améliorer la performance de la machine (b_j et a_j le temps opératoire de la tâche T_j avant et après la maintenance). Les auteurs ont démontré que la minimisation du nombre de tâches en retard $\sum U_j$ et la minimisation du retard maximal L_{max} peuvent être résolues en temps polynomial avec, respectivement, des complexités $\mathcal{O}(n^4 \log n)$ et $\mathcal{O}(n^2)$.

Dans [126], le problème considéré est $1|p(j,r,t) = (p_j - at)r^b, \mathcal{M} \leq 1, m = \alpha t + \beta|TADC$: un effet combiné de détérioration et d'apprentissage sur la machine, le temps opératoire défini par $p(j,r,t) = (p_j - at)r^b$, avec a et b respectivement, le facteur d'apprentissage et de détérioration, au plus une période de maintenance de durée variable est à insérer. Cette dernière permet de rétablir les conditions initiales de la machine. L'objectif est de minimiser la variation des dates de fin ($TADC$) et aussi de trouver la position optimale de la maintenance. Yang démontre que lorsque la position de la maintenance est fixée, la séquence optimale est obtenue en utilisant un algorithme de tri en $\mathcal{O}(n \log n)$. Sachant que n positions sont possibles pour insérer la maintenance, le problème peut donc être résolu avec une complexité $\mathcal{O}(n^2 \log n)$.

2.4.2.2 Problèmes à machines parallèles

Les études faites sur les problèmes à machines parallèles en présence de périodes d'indisponibilité de durées variables sont généralement menées avec deux critères d'optimisation classiques, à savoir ; la minimisation de la charge totale des machines ($TML = \sum_{l=1}^{m} C_{max}^l$) et la somme totale des dates de fin ($TC = \sum C_j$). Ces résultats seront présentés dans les deux premiers paragraphes. Nous présenterons également, dans le dernier paragraphe, des résultats sur les problèmes à machines parallèles avec d'autres critères

d'optimisation.

a. Minimisation de la charge totale des machines

Pour les problèmes avec des périodes d'indisponibilité de durée variable, Cheng *et al.* [21] ont étudié le problème $Rm|p(l,j)=(b_{lj},a_{lj}),\mathcal{M}_l\leq 1, m_l=\alpha_l t_l+\beta_l|\sum_{l=1}^{m} C_{max}^{l}$ avec une période de maintenance optionnelle [2] sur chaque machine. La durée de la maintenance sur la machine M_l varie linéairement en fonction de sa date de début $m_l=\alpha_l t_l+\beta_l$, avec t_l, α_l et β_l sont, respectivement, la date de début de la maintenance, le facteur de la détérioration de la maintenance et sa durée élémentaire sur la machine M_l. La durée de la tâche T_j ordonnancée sur la machine M_l avant la maintenance est égale à b_{lj} et si la tâche T_j est placée après la maintenance sa durée d'exécution sera a_{lj}, où $b_{lj}\geq a_{lj}$. Les auteurs démontrent que la minimisation de la charge totale des machines est polynomiale, si le nombre de machines m est fixe, avec une complexité $\mathcal{O}(n^{m+3})$.

Hsu *et al.* [56] considèrent un problème similaire avec la configuration suivante : m machines parallèles non liées avec une période de maintenance dont la durée varie linéairement en fonction de sa date de début $m_l=\alpha t_l+\beta$, où $\alpha\geq 0$ et $\beta>0$ sont, respectivement, le facteur de détérioration de la maintenance et la durée élémentaire de la maintenance. Les auteurs considèrent des périodes de maintenance optionnelles sur chaque machine. Trois modèles de détérioration ont été étudiés :

- modèle de détérioration exponentiel : la durée de la tâche T_j ordonnancée à la position r sur la machine M_l est $p(l,j,r)=p_{lj}r^{a_{lj}}$.
- modèle de détérioration linéaire en fonction de la date de début : la durée de la tâche T_j ordonnancée à la position r à l'instant t sur la machine M_l est $p(l,j,t)=p_{lj}+b_j t$.
- modèle de détérioration linéaire en fonction de la position : la durée de la tâche T_j ordonnancée à la position r sur la machine M_l est $p(l,j,r)=p_{lj}+a_{lj}r$.

où $a_{lj}>0$ et $b_j>0$ des facteurs de la détérioration.

Les auteurs ont montré que les problèmes $Rm|p(l,j,r),\mathcal{M}_l\leq 1, m_l=\alpha_l t_l+\beta_l|\sum_{l=1}^{m} C_{max}^{l}$ étudiés sont polynomiaux (pour les trois modèles de détérioration), une méthode de résolution a été proposée avec une complexité $\mathcal{O}(n^{2m+2})$.

b. Minimisation de la somme des dates de fin

Les problèmes d'ordonnancement simultané des tâches et des périodes d'indisponibilité de durée variable ont aussi été étudiés avec comme objectif la minimisation de la somme des dates de fin. Dans ce paragraphe, nous nous focalisons sur ces problèmes d'ordonnancement.

Wang *et al.* [116] ont proposé un algorithme polynomial de complexité $\mathcal{O}(n^{2m+3})$ pour

2. Au plus, une période de maintenance sur chaque machine.

résoudre le problème $Pm|p(j) = (p_j, \delta_j p_j), \mathcal{M}_l \leq 1, m_l = \alpha_l t_l + \beta_l | \sum C_j$. Les auteurs considèrent m machines parallèles identiques et au plus une période d'indisponibilité sur chaque machine. La durée de l'indisponibilité varie en fonction de sa date de début $m_l = \alpha_l t_l + \beta_l$, avec α_l et β_l les paramètres de la maintenance et t_l sa date d'insertion sur la machine M_l. La durée de la tâche T_j avant l'indisponibilité est égale à p_j et si elle est placée après l'indisponibilité sa durée d'exécution sera $\delta_j p_j$, avec $\delta_j > 0$. Notons que dans cet article, il n'y a pas de restriction sur la valeur de δ_j, autrement dit, la durée de la tâche T_j peut être diminuée après la maintenance si $0 < \delta_j \leq 1$ ou augmentée si $1 < \delta_j$.

Peu de temps après, une amélioration du problème précédent a été proposée par Wang *et al.* [117]. Les auteurs ont proposé un algorithme de résolution d'une complexité plus faible $\mathcal{O}(n^{m+3})$ sur m machines parallèles non liées (Rm).

Cheng *et al.* reprennent dans [21] le problème précédent, mais avec un temps opératoire plus général $p(l, j) = (b_{lj}, a_{lj})$, avec $b_{lj} \geq a_{lj}$ (la maintenance améliore les performances de la machine). Cette étude a montré que le problème $Rm|p(l, j) = (b_{lj}, a_{lj}), \mathcal{M}_l \leq 1, m_l = \alpha_l t_l + \beta_l | \sum C_j$ est polynomial. Un algorithme de résolution a été proposé avec une complexité $\mathcal{O}(n^{m+3})$. Plus récemment, Yang *et al.* [130] ont montré que le problème reste aussi polynomial lorsque $b_{lj} < a_{lj}$.

Hsu *et al.* [56] considèrent un problème d'ordonnancement de tâches et de maintenances variables sur m machines parallèles en présence d'un effet de détérioration. La durée de la maintenance varie linéairement en fonction de sa date de début $m_l = \alpha_l t_l + \beta$, où α_l, β et t_l sont, respectivement, le facteur de la détérioration, la durée élémentaire de la maintenance et sa date de début sur M_l. Les auteurs considèrent des périodes de maintenance optionnelles sur chaque machine. Trois modèles de détérioration ont été étudiés :

- modèle de détérioration exponentiel : $p(l, j, r) = p_{lj} r^{a_{lj}}$
- modèle de détérioration linéaire en fonction de la date de début : $p(l, j, t) = p_{lj} + b_j t$
- modèle de détérioration linéaire en fonction de la position : $p(l, j, r) = p_{lj} + a_{lj} r$

Avec $a_{lj} > 0$ et $b_j > 0$ des facteurs de la détérioration. L'insertion de la période de maintenance permet de rétablir les conditions initiales de la machine. L'objectif est la minimisation de $\sum C_j$. Les auteurs ont montré que les problèmes $Rm|p(l, j, r), \mathcal{M}_l \leq 1, m_l = \alpha_l t_l + \beta | \sum C_j$ étudiés sont polynomiaux, une méthode de résolution a été proposée avec une complexité $\mathcal{O}(n^{2m+2})$.

Le problème à machines parallèles non liées avec plusieurs maintenances variables sur chaque machine a été étudié par Yang [129]. Il considère deux scénarios :

- Un effet de détérioration qui varie en fonction de la position de la tâche :
  $p(l, j, r) = p_{lj} f_{lj}(r)$ avec p_{lj} : la durée standard de la tâche j sur la machine M_l, $f_{lj}(r)$ le facteur de détérioration de la tâche j ordonnancée sur la machine M_l à la position r, tel que $f_{lj}(r) \geq 1$ et $f_{lj}(1) = 1$. Le facteur dépend donc de la tâche, de

sa position et de la machine.

– Un effet de détérioration qui varie en fonction de la date de début de tâches : $p(l, j, t) = p_{lj} + c_r t$, avec c_r et t, respectivement, le facteur de détérioration de la tâche ordonnancée à la position r et sa date de début.

L'auteur suppose que les durées de maintenance varient linéairement en fonction de leurs dates de lancement tel que la durée de la $i^{\text{ème}}$ maintenance sur la machine M_l est $m_{li} = \alpha_l t_{li} + \beta_l$, avec t_{li} sa date de lancement à partir de la maintenance précédente. Une borne supérieure k_0 a été proposée par l'auteur pour le nombre maximal de maintenances. L'objectif est de trouver les positions et le nombre optimal de maintenances sur chaque machine et de minimiser $\sum C_j$. En utilisant un problème d'affectation, l'auteur démontre que la résolution du problème $Rm|p(l, j, r), \mathcal{M} \leq k_0, m_{li} = \alpha_l t_{li} + \beta_l| \sum C_j$ est polynomiale avec une complexité $\mathcal{O}(n^{m+k_0+2})$ pour les deux scénarios.

Nous résumons dans le tableau 2.10 les différents travaux sur les machines parallèles en présence de périodes de maintenance avec durées variables. La première colonne du tableau regroupe les références. La deuxième colonne précise l'atelier et le critère d'optimisation. Ensuite, les variations appliquées sur les tâches sont présentées dans la troisième colonne. Le nombre de maintenances à insérer et la complexité de résolution sont indiqués dans les deux dernières colonnes.

c. Autres critères

Dans la littérature, d'autres critères ont été étudiés. Wang et Wei [114] ont considéré un problème à machines parallèles. Chaque machine peut être maintenue au plus une seule fois. La durée de la maintenance varie en fonction de sa date de début $(\alpha_l t_l + \beta_l)$ avec $l = 1, \ldots, m$ et m le nombre de machines. L'insertion de cette période de maintenance permet d'améliorer la performance de la machine : $b_{lj} \geq a_{lj}$ avec b_{lj} et a_{lj}, respectivement, les durées de la tâche T_j sur la machine M_l avant et après la maintenance. Les auteurs ont étudié cette configuration avec deux objectifs : minimisation de la variation des dates de fin $(TADC)$ et la minimisation de la variation des dates de début (ou les durées d'attente) $(TADW)$. En utilisant un problème d'affectation, les auteurs ont montré que ces deux problèmes sont polynomiaux avec une complexité $\mathcal{O}(n^{2m+2})$.

Tableau 2.10 – Problèmes d'ordonnancement simultané des tâches et des périodes d'indisponibilité de durées variables à machines parallèles

Référence		Atelier	Critère	Durée des tâches			Nombre de maintenances	Complexité
				Paramètres de variation		RMA		
				Position	Temps			
Cheng *et al.*	[21]	Rm	$\sum_{l=1}^{m} C_{max}^{l}$			✓	≤ 1	$\mathcal{O}(n^{m+3})$
Hsu *et al.*	[56]	Rm	$\sum_{l=1}^{m} C_{max}^{l}$	✓			≤ 1	$\mathcal{O}(n^{2m+2})$
Hsu *et al.*	[56]	Rm	$\sum_{l=1}^{m} C_{max}^{l}$		✓		≤ 1	$\mathcal{O}(n^{2m+2})$
Wang *et al.*	[116]	Pm	$\sum_{l=1}^{m} C_j$			✓	≤ 1	$\mathcal{O}(n^{2m+3})$
Wang *et al.*	[117]	Rm	$\sum_{l=1}^{m} C_j$			✓	≤ 1	$\mathcal{O}(n^{m+3})$
Cheng *et al.*	[21]	Rm	$\sum_{l=1}^{m} C_j$			✓	≤ 1	$\mathcal{O}(n^{m+3})$
Yang *et al.*	[130]	Rm	$\sum_{l=1}^{m} C_j$			✓	≤ 1	$\mathcal{O}(n^{m+3})$
Hsu *et al.*	[56]	Rm	$\sum_{l=1}^{m} C_j$	✓			≤ 1	$\mathcal{O}(n^{2m+2})$
Hsu *et al.*	[56]	Rm	$\sum_{l=1}^{m} C_j$		✓		≤ 1	$\mathcal{O}(n^{2m+2})$
Yang	[129]	Rm	$\sum_{l=1}^{m} C_j$	✓			$\leq k_0$	$\mathcal{O}(n^{m+k_0+2})$

2.4.2.3 Problèmes flow shop et open shop

Kubzin et Strusevich [67, 68], en 2005, ont été les premiers à introduire l'indisponibilité de durée variable dans les problèmes d'ordonnancement. Dans [68], les auteurs ont considéré le problème suivant : un problème open shop à deux machines, chaque machine doit être maintenue une seule fois. La durée de la maintenance suit un effet de détérioration qui varie en fonction de sa date de début $m_l = \beta_l + f_l(t)$ la durée de maintenance sur la machine M_l avec $\beta \geq 0$ et $f_l(t)$ une fonction croissante tel que $f_l(0) = 0$. Les auteurs démontrent que minimiser le C_{max} est polynomial avec une complexité $\mathcal{O}(n)$. Pour la même configuration avec un atelier flow shop, les auteurs prouvent que la minimisation de C_{max} est NP-difficile lorsque la période de maintenance est autorisée sur la première machine. Un algorithme de programmation dynamique a été développé pour résoudre ce problème.

Dans [67], un problème flow shop à deux machines sans attente dans lequel la première machine nécessite une maintenance de durée variable $m = \beta + f(t)$ a été étudié par Kubzin et Strusevich. Les auteurs démontrent que le problème est NP-difficile.

2.5 Positionnement de notre problématique

Dans cette thèse, nous nous intéressons aux problèmes d'ordonnancement simultané de tâches et de périodes d'indisponibilité de durées variables. L'étude bibliographique réalisée dans la section précédente aide à tracer les objectifs et les motivations pour le reste de cette thèse.

Nous constatons que les problèmes à une machine et machines parallèles ont été largement étudiés dans la littérature avec différentes fonctions qui modélisent la variation de tâches et de maintenances. Également, plusieurs critères d'optimisation ont été évalués tels que : C_{max}, $\sum C_j$, $TADC$, $TADW$,etc. Notre travail a pour originalité : d'aborder ces problèmes, mais avec une vision plus générale, nous définirons ainsi des modèles génériques de variations de tâches qui englobent les modèles antérieurs et nous présenterons une approche globale capable de généraliser des résultats obtenus par ces problèmes. Nous résoudrons ainsi ces différents problèmes avec plusieurs autres critères d'optimisation. Enfin, nous améliorons certaines complexités de résolution.

La prise en compte des périodes d'indisponibilité dans les problèmes d'atelier (flow shop, job shop, open shop) a été très peu étudiée. Nous nous intéressons donc en second lieu, dans cette étude, aux problèmes de type flow shop en présence de périodes d'indisponibilité variables.

Pour conclure, les principales originalités de ce travail sont :

- Définir un modèle général pour les systèmes de production avec des indisponibilités qui englobe les modèles antérieurs sur une machine et sur machines parallèles.
- Développer une approche globale de résolution pour les problèmes à une machine et machines parallèles, ce qui permet de généraliser et améliorer les résultats antérieurs.
- Étudier les problèmes flow shop et proposer des méthodes de résolution.

2.6 Conclusion

Cette thèse se concentre sur les problèmes d'ordonnancement simultané de tâches et de périodes d'indisponibilité. Une revue de littérature sur le sujet est donc réalisée dans ce chapitre. À partir de cette étude, nous avons tracé les objectifs et les motivations de cette thèse.

Dans la partie suivante *(Partie II, Approche globale)*, nous traitons les deux premiers objectifs de la thèse ; nous définissons d'abord dans le chapitre 3 un modèle global qui généralise les modèles antérieurs. Ensuite, nous proposons dans les chapitres 4 et 5 une approche globale pour résoudre ce modèle avec un atelier à une machine et à machines parallèles. Une comparaison sera faite entre les résultats obtenus et les résultats antérieurs.

Deuxième partie

Approche globale

3

Modèle global

Résumé : Dans ce chapitre, nous proposons un modèle global de prise en compte des indisponibilités dans les systèmes de production. La nouvelle modélisation de maintenance que nous proposons englobe les modèles classiques présentés dans la première partie (Partie I). De plus, nous considérons un modèle général pour la variation des tâches, contrairement aux travaux antérieurs dans lesquels des fonctions analytiques ont été utilisées pour modéliser l'effet d'apprentissage et de détérioration. Après avoir présenté notre modèle dans le cas à une machine, nous généralisons notre approche aux ateliers à machines parallèles et aux modèles dépendants du groupe.

Sommaire

3.1 Introduction

Comme cela a déjà été précisé précédemment, un des objectifs principaux de cette thèse est la présentation d'une approche globale de résolution des problèmes d'ordonnancement en présence de périodes d'indisponibilité. Ainsi, avant d'entamer la présentation de cette approche, un modèle général doit être défini. Dans ce chapitre, nous présentons une modélisation des systèmes de production en présence de périodes de maintenance qui englobe les modèles classiques présentés dans la première partie de la thèse (Partie I). Nous présentons, dans la section 3.2, ce modèle et en particulier la variation des temps opératoires et de la durée de la maintenance. Des extensions vers un modèle à machines parallèles et un modèle dépendant du groupe sont présentées dans la section 3.3. Nous dédions la dernière section 3.4 au rappel de quelques définitions autour des problèmes d'affectation.

3.2 Modèle global

Nous nous concentrons dans cette partie sur un modèle global dépendant de la position de la tâche *"the position-dependent model"*. Nous présentons, dans ce qui suit, les différentes caractéristiques de ce modèle dans un atelier à une machine.

3.2.1 Modèle global pour les temps opératoires

Dans notre étude, nous nous intéressons aux fonctions qui varient selon la position de la tâche, avec $p(j,r)$ le temps opératoire de la tâche T_j ordonnancée au rang r sur la machine. Dans la littérature, plusieurs fonctions spécifiques ont été étudiées pour illustrer la variation du temps opératoire. Parmi les fonctions utilisées, nous pouvons citer : $p_j r^a$ ([11], [87], [88]) et $p_j \sigma^{r-1}$ ([41]), $p_j + b_j r$ (voir [7], [135] et [56]), $p_j r^{a_j}$ (voir [89], [134], [135] et [138]), $p_j \sigma_j^{r-1}$ (voir [136]). Une liste des fonctions analytiques a été présentée dans les tableaux 1.4 et 1.3.

Contrairement aux travaux antérieurs, dans notre modèle aucune fonction spécifique n'est imposée. Le temps opératoire de la tâche T_j ordonnancée au rang r est simplement défini par $p(j,r)$.

Pour modéliser un effet de détérioration, nous imposons l'inégalité suivante :

$$p(j,1) \leq p(j,2) \leq \ldots \leq p(j,n), \quad \forall\, j = 1, \ldots, n \tag{3.1}$$

Nous obtenons un effet d'apprentissage, si les temps opératoires vérifient :

$$p(j,1) \geq p(j,2) \geq \ldots \geq p(j,n), \quad \forall\, j = 1, \ldots, n \tag{3.2}$$

En effet, dans un cas réel, les durées $p(j,r)$ peuvent être déterminées à partir de l'historique du système de production. Soit D une matrice de n lignes et n colonnes qui contient les temps opératoires $p(j,r)$.

$$D = \begin{array}{c} \text{j / r} \\ T_1 \\ T_2 \\ \\ T_n \end{array} \begin{array}{c} r=1 \\ \begin{pmatrix} p(1,1) \\ p(2,1) \\ \vdots \\ p(n,1) \end{pmatrix} \end{array} \begin{array}{c} r=2 \\ \begin{pmatrix} p(1,2) \\ p(2,2) \\ \vdots \\ p(n,2) \end{pmatrix} \end{array} \begin{array}{c} \ldots \\ \begin{pmatrix} p(1,n) \\ p(2,n) \\ \vdots \\ p(n,n) \end{pmatrix} \end{array} \begin{array}{c} r=n \\ \end{array}$$

Pour étudier un modèle classique, par exemple $p_j r^{a_j}$, il suffit de remplir la matrice en utilisant la fonction de détérioration $p(j,r) = p_j r^{a_j}$.

L'effet appliqué sur la machine s'intègre dans la notation de Graham au sein du deuxième champ via les notations « $p(j,r) - det$ » ou « $p(j,r) - app$ » si l'effet de détérioration ou l'effet d'apprentissage, respectivement, sont imposés. Lorsqu'aucun effet n'est appliqué, la variation des temps opératoires est notée par « $p(j,r)$ ».

3.2.2 Modèle de maintenances pondérées

Dans ce qui suit, nous proposons un nouveau modèle de maintenance, que nous notons $\mathcal{MP}$, qui englobe et généralise les modèles de maintenances antérieures. L'idée de ce modèle est la suivante : nous considérons qu'une tâche T_j ordonnancée dans la position r sur la machine contribue à l'usure de la machine de manière proportionnelle à sa durée opératoire. Elle contribue ainsi indirectement à la durée de la maintenance avec une durée proportionnelle à son temps opératoire $w(j,r)p(j,r)$. La maintenance permet de rétablir les conditions initiales de la machine.

Dans un système de production à une machine et avec plusieurs périodes de maintenance, soit k périodes de maintenance sur la machine, nous obtenons une répartition des tâches en $k+1$ groupes $\{G_1, m_1, G_2, \ldots, G_k, m_k, G_{k+1}\}$, où G_i le groupe des tâches entre la maintenance $i-1$ et i de taille n_i telle que $\sum_{i=1}^{k+1} n_i = n$. Notons qu'après chaque période de maintenance la machine revient à son état initial, les rangs r sont donc réinitialisés après la fin de la maintenance, voir la figure 3.1.

Avec ce modèle, la durée de la $i^{\text{ème}}$ maintenance est égale à :

$$m_i = \sum_{j \in G_i} w(j,[j])p(j,[j]) + \beta_i \tag{3.3}$$

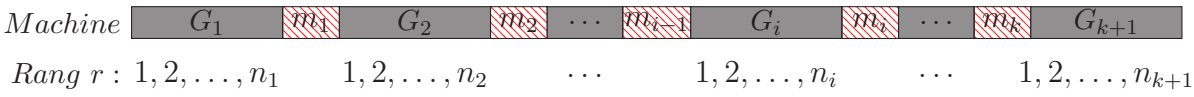

Figure 3.1 – Répartition des tâches en $k+1$ groupes

où β_i la durée élémentaire de la maintenance i et $r = [j]$ définit le rang de la tâche T_j dans le groupe G_i.

Exemple 3.1. *Afin de mieux comprendre ce modèle, nous considérons l'exemple suivant : une machine avec des temps opératoires qui varient selon la position r et deux périodes de maintenance à ordonnancer dont les durées élémentaires sont $\beta_1 = 0,75$ et $\beta_2 = 0,5$. La durée de la maintenance i est donnée par le modèle précédent (équation 3.3). Les temps opératoires $p(j,r)$ et les coefficients de maintenance $w(j,r)$ sont présentés dans les tableaux 3.1 et 3.2.*

Tableau 3.1 – Temps opératoires des tâches

	Position r				
$p(j,r)$	1	2	3	4	5
T_1	1,5	2	3	5	7
T_2	1,5	1,75	2	3	4
T_3	1	1,5	2	3	5
T_4	2	2,5	3	4	5
T_5	1	2	2	2	3

Tableau 3.2 – Coefficients de maintenance

	Position r				
$w(j,r)$	1	2	3	4	5
T_1	0,2	0,4	0,5	0,8	0,7
T_2	0,5	0,8	0,8	0,9	0,9
T_3	0,5	0,833	1,5	1,5	1,5
T_4	0,375	0,8	0,8	0,9	1
T_5	0,5	0,625	1,25	1,4	1,5

Soit la séquence suivante $\{T_2 - T_5 - m_1 - T_4 - T_3 - m_2 - T_1\}$. Après la première période de maintenance m_1 la machine revient à son état initial, ainsi le rang de la tâche T_4 est $r = 1$ et sa durée est égale à $p(4,1) = 2$, la tâche T_3 est en deuxième position dans le groupe G_2 avec une durée $p(3,2) = 1,5$.

Il est clair, d'après l'exemple, que la durée de la maintenance m_i est constituée par des portions $w(j,r)p(j,r)$ des tâches T_j exécutées dans le groupe G_i et en ajoutant une durée élémentaire β_i. Cette remarque est illustrée dans la figure 3.2.

D'après le modèle de maintenance présenté précédemment :

- *la durée de la première maintenance est*
  $m_1 = \sum_{j \in G_1} w(j,[j])p(j,[j]) + \beta_1 = w(2,1)p(2,1) + w(5,2)p(5,2) + \beta_1 = 0,5 \times 1,5 + 0,625 \times 2 + 0,75 = 2,75$
- *la durée de la deuxième maintenance est*
  $m_2 = \sum_{j \in G_2} w(j,[j])p(j,[j]) + \beta_2 = w(4,1)p(4,1) + w(3,2)p(3,2) + \beta_2 = 0,375 \times 2 +$

$0,833 \times 1,5 + 0,5 = 2,5$

La durée totale de l'ordonnancement $C_{max} = 13,75$*, voir figure 3.2.*

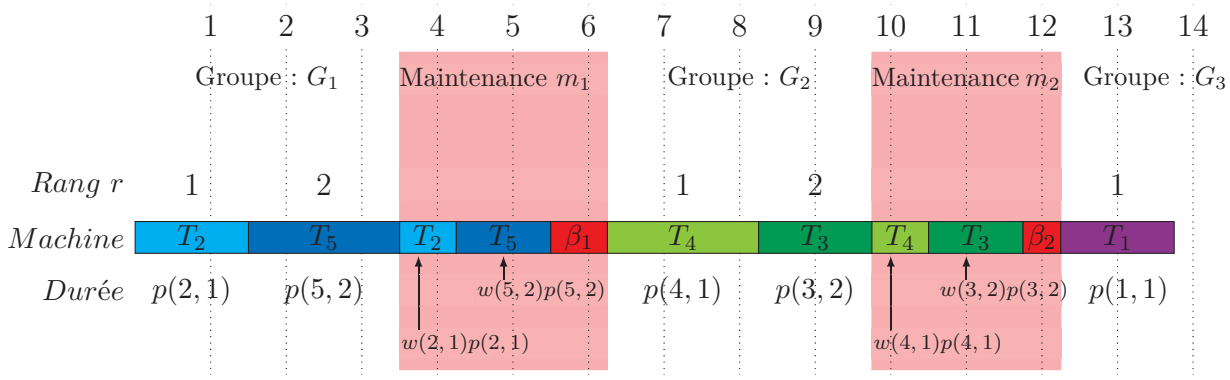

Figure 3.2 – Exemple avec une machine et deux périodes de maintenance

Ce nouveau système de maintenance pondérée s'intègre dans la notation de Graham au sein du deuxième champ. Ci-dessous, nous présentons une liste des notations utilisées dans le reste de la thèse :

- $\mathcal{MP}$: signifie que les indisponibilités suivent le modèle de maintenances pondérées et que le nombre de maintenances à insérer sur la machine est une variable de décision.
- $\mathcal{MP} = k$: les indisponibilités suivent le modèle de maintenances pondérées et le nombre de maintenances sur la machine est imposé et est égal à k.
- $\mathcal{MP} \leq k_0$: les indisponibilités suivent le modèle de maintenances pondérées et le nombre de maintenances sur la machine à une borne supérieure égale à k_0.
- $\mathcal{MP}(n_i) = k$: les indisponibilités suivent le modèle de maintenances pondérées, le nombre de maintenances et la taille de chaque groupe G_i sont connus d'avance. Le terme « $= k$ » est facultatif, car si les tailles des groupes sont connues, cela implique que le nombre de maintenances est une donnée.

Ainsi, $1|p(j,r), \mathcal{MP}|\gamma$ désigne un problème d'ordonnancement de tâches à durées variables (en fonction de la position r) et de maintenances qui suivent le modèle général $\mathcal{MP}$, dans un atelier à une machine. L'objectif est de trouver la séquence de tâches qui minimise le critère d'optimisation γ et de déterminer le nombre et les positions optimales de maintenance.

On peut toujours se ramener aux modèles classiques présentés dans la section 1.4 :

- maintenance avec une durée constante $m_i = \beta$ en fixant $w(j,r) = 0$ et $\beta_i = \beta$.
- maintenance avec une durée linéaire $m_i = \alpha t_i + \beta$ en fixant $w(j,r) = \alpha$ et $\beta_i = \beta$.

Remarque 3.1. *L'insertion d'une période de maintenance au début de l'ordonnancement (à l'instant t=0) ou à la fin de l'ordonnancement est possible d'un point de vue*

mathématique, mais non intéressante pour la minimisation du critère d'optimisation. En effet, la période de maintenance insérée au début de l'ordonnancement n'a pas d'impact sur les durées des tâches ordonnancées après, car la machine est déjà considérée dans son état initial. Ainsi, l'insertion de cette période de maintenance ne peut que retarder l'ordonnancement avec une durée égale à la durée de maintenance β_0, voir figure 3.3. De même, l'insertion d'une période d'indisponibilité à la fin de l'ordonnancement n'a pas d'impact sur les tâches puisque toutes les tâches ont été ordonnancées avant. Notons que, dans certains cas, la maintenance à la fin de l'ordonnancement peut être imposée.

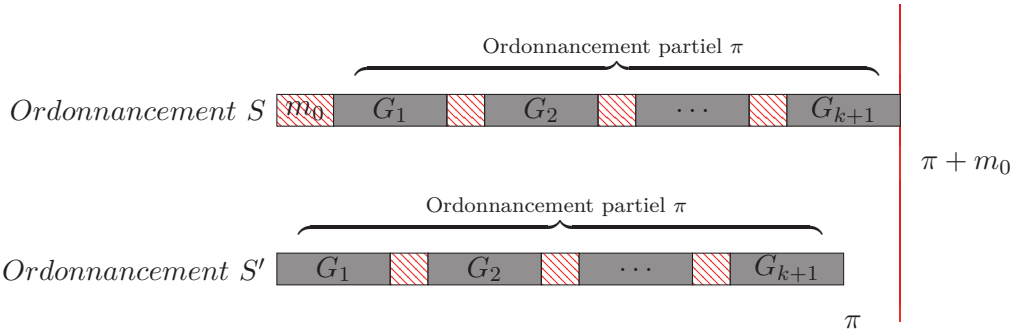

Figure 3.3 – Ordonnancement S avec une période de maintenance à $t = 0$ et ordonnancement S' sans période de maintenance

3.3 Extensions

Le modèle étudié précédemment peut être étendu à d'autres modèles. Nous présentons dans cette section deux extensions, à savoir, une extension vers les machines parallèles et une extension vers les ordonnancements par groupe.

3.3.1 Ateliers à machines parallèles

Précédemment, nous avons présenté un modèle global d'un système de production en présence de périodes d'indisponibilité dans un atelier à une machine. Néanmoins, l'atelier à machines parallèles est plus intéressant et proche des problèmes réels rencontrés dans la pratique. Nous pouvons facilement étendre notre modèle aux machines parallèles. Nous considérons, m machines parallèles et un ensemble de n tâches à ordonnancer sur ces machines. Le temps opératoire de la tâche T_j ordonnancée dans la position r sur une machine M_l est défini par $p(l, j, r)$. La machine M_l suit :

– Un effet de détérioration si le temps opératoire d'une tâche T_j augmente avec r :

$$p(l, j, 1) \leq p(l, j, 2) \leq \ldots \leq p(l, j, n), \quad \forall\, j = 1, \ldots, n, \quad l = 1, \ldots, m$$

– Un effet d'apprentissage si le temps opératoire d'une tâche T_j diminue avec r :

$$p(l,j,1) \geq p(l,j,2) \geq \ldots \geq p(l,j,n), \quad \forall\, j = 1,\ldots,n, \quad l = 1,\ldots,m$$

Des périodes de maintenance peuvent être insérées sur les machines. Nous étendons ici le modèle de maintenances pondérées $\mathcal{MP}$ présenté dans la section 3.2.2 aux machines parallèles. La durée des périodes de maintenance est pondérée par des coefficients $w(l,j,r)$. Ainsi, la durée de la $i^{ème}$ période de maintenance sur la machine M_l est définie par :

$$m_{li} = \sum_{j \in G_i^l} w(l,j,[j]) p(l,j,[j]) + \beta_{li} \tag{3.4}$$

avec G_i^l l'ensemble de tâches entre la maintenance $i-1$ et la maintenance i sur la machine M_l, G_1^l l'ensemble de tâches entre l'instant 0 et la première période de maintenance, voir figure 3.4 et $r = [j]$ définit le rang de la tâche T_j dans le groupe G_i^l et β_{li} la durée élémentaire de la maintenance i sur la machine M_l.

Figure 3.4 – Répartition des tâches sur la machine M_l avec k_l périodes de maintenance

3.3.2 Ordonnancement par groupe

Dans de nombreux procédés de fabrication, l'efficacité de production peut être augmentée en regroupant plusieurs éléments et produits avec des motifs semblables ou processus de production. Ce phénomène est connu en tant que technologie de groupe dans la littérature. De nombreux avantages ont été présentés par plusieurs applications de cette technologie, tels que [52], [58], [76] et [127].

Dans cette section, nous étendons notre modèle présenté précédemment aux problèmes d'ordonnancement par groupe.

Dans la réalité industrielle, plusieurs équipes de maintenance peuvent être amenées à intervenir durant la réalisation des tâches, voir figure 3.5. L'efficacité de ces interventions de maintenance varie d'une équipe à l'autre selon l'expérience des opérateurs. Ainsi, même après une activité de maintenance, il peut rester des usures sur la machine et donc une possibilité d'utiliser la machine avec des conditions initiales dégradées après chaque intervention de maintenance. Nous considérons ainsi des temps opératoires qui dépendent du groupe $p^{(i)}(j,r)$.

Exemple 3.2. *Dans l'exemple illustré dans la figure 3.5, trois interventions sont planifiées sur la machine. Chaque période de maintenance est assurée par une équipe différente.*

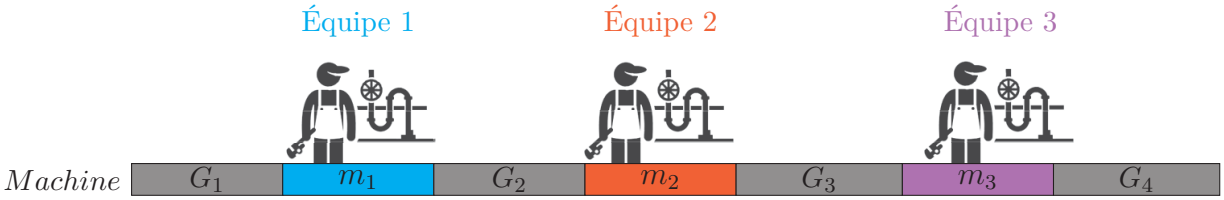

Figure 3.5 – Ordonnancement avec plusieurs équipes de maintenance

Notons que Rustogi et Strusevich ont étudié, dans [98, 99], un problème d'ordonnancement de tâches et de maintenances dépendant du groupe. Le modèle présenté par Rustogi et Strusevich [98, 99] est le suivant ; les temps opératoires des tâches dans les groupes sont de la forme $p_j g_j^{[i]}(r)$ et les facteurs $g_j^{[i]}(r)$ augmentent avec l'augmentation du rang r :

$$g_j^{[i]}(1) \leq g_j^{[i]}(2) \leq \ldots \leq g_j^{[i]}(n) \tag{3.5}$$

Il est également supposé que les facteurs sont croissants avec les groupes G_i :

$$g_j^{[1]}(r) \leq g_j^{[2]}(r) \leq \ldots \leq g_j^{[k+1]}(r) \tag{3.6}$$

La durée de la maintenance est donnée sous la forme[1] $m_i = \alpha_i t_i + \beta_i$, avec $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_k$.

Dans notre étude, nous considérons un modèle plus général. On définit des temps opératoires dépendants du groupe $p^{(i)}(j,r)$ qui augmentent avec les rangs pour chaque groupe G_i. En fait, et contrairement aux travaux de Rustogi et Strusevich [98, 99], il n'y a aucune corrélation imposée entre les temps opératoires dans les différents groupes. Nous avons juste un environnement de réalisation différent pour chaque groupe, suite par exemple à la qualité variable de la maintenance. Nous étendons aussi le modèle de maintenance $\mathcal{MP}$ comme suit : la tâche T_j ordonnancée en position r dans le groupe G_i augmente la durée de la maintenance réalisée par l'équipe i avec une valeur $w^{(i)}(j,r)p^{(i)}(j,r)$. Nous obtenons le modèle de Rustogi [98, 99] avec $w^{(i)}(j,r) = \alpha_i$. On note ce nouveau modèle de maintenance dépendant du groupe : $\mathcal{MP}^{(i)}$.

1. Le modèle classique introduit par Kubzin en 2005 [67, 68].

3.4 Les problèmes d'affectation linéaires

Avant d'entamer l'analyse et la résolution du modèle dans le chapitre suivant, concentrons-nous d'abord sur le problème d'affectation linéaire qui sera le modèle sous-jacent principal pour développer notre approche de résolution. Nous dédions cette section au rappel de quelques définitions autour de ce problème.

3.4.1 Problème d'affectation linéaire

Un problème d'affectation $\mathcal{PA}$ est défini comme le problème qui consiste à associer chaque élément d'un ensemble U de n items à un seul élément d'un autre ensemble V de n items avec un coût minimal. Soit une matrice de taille $n \times n$ contenant les coûts c_{ij} et le problème d'affectation consiste à résoudre le programme linéaire suivant :

Programme linéaire en nombres entiers 1 (Problème d'affectation linéaire $\mathcal{PA}$)**.**

$$
\begin{aligned}
min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & & \\
s.c. \quad & \sum_{j=1}^{n} x_{ij} = 1 & i = 1,\dots,n & \quad (3.7)\\
& \sum_{i=1}^{n} x_{ij} = 1 & j = 1,\dots,n & \quad (3.8)\\
& x_{ij} \in \{0,1\} & i = 1,\dots,n,\ j = 1,\dots,n &
\end{aligned}
$$

Les premières contraintes 3.7 expriment que chaque item i de U doit être affecté à un seul item de l'ensemble V. Les secondes contraintes 3.8 assurent que chaque item j de V doit être associé à un seul item de l'ensemble U. Grâce au fait que la matrice du programme linéaire est totalement unimodulaire (voir Schrijver [103]), l'optimum de ce PLNE-1 est égal à l'optimum de sa relaxation, et ce PLNE est en particulier résolu en temps polynomial.

Le problème d'affectation peut aussi être défini en théorie des graphes comme un problème de couplage parfait de coût minimal dans un graphe biparti complet $G = (U \cup V, E)$, avec $|U| = |V| = n$. On associe à chaque arête $(i,j) \in E$ le coût c_{ij}. Nous représentons, dans la figure 3.6, une instance de problème d'affectation.

Le problème d'affectation linéaire $\mathcal{PA}$ peut être résolu en temps polynomial $\mathcal{O}(n^3)$, voir Dinic et Kronrod [26].

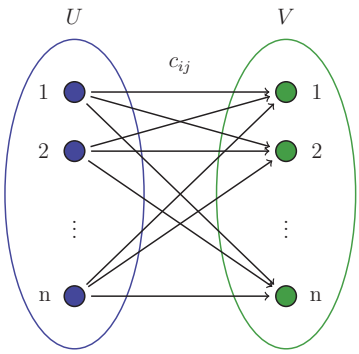

Figure 3.6 – Problème d'affectation sous forme d'un graphe biparti

3.4.2 Problème d'affectation rectangulaire

Une extension du problème d'affectation classique a été utilisée à plusieurs reprises dans les problèmes d'ordonnancement. Cette extension est connue sous le nom de problème d'affectation rectangulaire $\mathcal{PAR}$. Dans ce problème, une matrice de coûts $n \times q$, avec $n \leq q$, est donnée, où l'on sélectionne un seul élément dans chaque ligne et au plus un élément dans chaque colonne de sorte que le coût total est minimisé.

Programme linéaire en nombres entiers 2 (Problème d'affectation rectangulaire $\mathcal{PAR}$).

$$
\begin{aligned}
min \quad & \sum_{i=1}^{n}\sum_{j=1}^{q} c_{ij}x_{ij} & & & \\
s.c. \quad & \sum_{j=1}^{q} x_{ij} & = 1 \quad & i = 1,\ldots,n & (3.9)\\
& \sum_{i=1}^{n} x_{ij} & \leq 1 \quad & j = 1,\ldots,q & (3.10)\\
& x_{ij} & \in \{0,1\} \quad & i = 1,\ldots,n,\ j = 1,\ldots,q &
\end{aligned}
$$

Les contraintes 3.9 assurent que chaque item i de l'ensemble U, avec $|U| = n$, est affecté à un élément de l'ensemble V. Les contraintes 3.10 expriment que chaque élément de V, avec $|V| = q$, est affecté à au plus un élément de l'ensemble U. Ainsi, pour une solution réalisable, seulement n éléments de V sont sélectionnés et $q - n$ éléments ne sont pas affectés.

Dans la littérature, deux algorithmes ont été développés pour résoudre le problème d'affectation rectangulaire $\mathcal{PAR}$. L'approche de Brukard et *al.* [15] a une complexité de résolution de $\mathcal{O}(qn^2)$. Le second algorithme est présenté initialement par Dinic [25], aussi que par Milo et *al.* [82] avec une complexité $\mathcal{O}(n^3 + nq)$.

3.5 Conclusion

Dans ce chapitre, un modèle général d'ordonnancement avec la prise en compte des indisponibilités dans un atelier à une machine a été présenté. Ce modèle comporte un système de maintenances pondérées $\mathcal{MP}$ qui englobe les modèles antérieurs présentés dans la littérature. De plus, aucune fonction spécifique n'est imposée pour la variation des temps opératoires contrairement aux modèles classiques. Ensuite, nous avons étendu ce modèle aux ateliers à machines parallèles et aux ordonnancements dépendants de groupe.

Nous présentons, dans le tableau 3.3, une comparaison entre le modèle global proposé dans ce chapitre et les modèles antérieurs. Dans la première partie du tableau, nous nous intéressons aux temps opératoires et ensuite aux modèles de maintenances.

Tableau 3.3 – Comparaison entre le modèle proposé et les modèles antérieurs

Temps opératoires

Modèle de l'approche globale	Modèles antérieurs			
	Fonction	Effet de détérioration	Effet d'apprentissage	Réf.
$p(l,j,r)$: aucune fonction spécifique - Effet de détérioration si $p(l,j,1) \leq \ldots \leq p(l,j,n)$ - Effet d'apprentissage si $p(l,j,1) \geq \ldots \geq p(l,j,n)$	$p_j r^a$	$a > 0$	$a < 0$	[11, 87, 88]
	$p_j r^{a_j}$	$a_j > 0$	$a_j < 0$	[89, 134, 138]
	$p_j + b_j r$	$b_j > 0$	$b_j < 0$	[7, 56, 135]
	$p_j \sigma^{r-1}$	$\sigma > 1$	$0 < \sigma < 1$	[41]
	$p_j \sigma_j^{r-1}$	$\sigma_j > 1$	$0 < \sigma_j < 1$	[136]
	$p_j g(r)$	$g(r)$ fonction croissante	$g(r)$ fonction décroissante	[98, 99]
	$p_j g_j(r)$	$g_j(r)$ fonction croissante	$g_j(r)$ fonction décroissante	[98, 99]

Maintenances

Modèle de l'approche globale	Modèles antérieurs	
	Modèles	Réf.
Durée de la $i^{\text{ème}}$ période de maintenance sur la machine M_l : $m_{li} = \sum_{j \in G_i^l} w(l,j,[j]) p(l,j,[j]) + \beta_{li}$	Durée de maintenance constante : $m_{li} = \beta$	[71]
	Durée de maintenance constante : $m_{li} = \beta_l$	[123, 132]
	Durée de maintenance constante : $m_{li} = \beta_{li}$	[59]
	Durée de maintenance variable : $m_{li} = \alpha t_i + \beta_{li}$	[129]

Dans les trois prochains chapitres, on abordera la résolution du modèle en utilisant plusieurs critères d'optimisation. Des extensions et des cas particuliers seront également étudiés.

4

Problèmes d'ordonnancement à une machine avec des périodes d'indisponibilité

Résumé : Dans ce chapitre, nous développons une approche globale de résolution des problèmes d'ordonnancement simultané des tâches et des indisponibilités sur une machine. En utilisant le modèle présenté dans le chapitre 3, cette approche généralise et améliore plusieurs résultats antérieurs menés sur les ateliers à une machine.

Sommaire

Ces travaux ont été réalisés en collaboration avec Gerd Finke, Vincent Jost et Julien Moncel. Les résultats développés dans ce chapitre ont été présentés dans les articles suivants :

[30] G. Finke, A. Gara-Ali, M.-L. Espinouse, V. Jost, et J. Moncel. *Unified matrix approach to solve production-maintenance problems on a single machine*. Omega (2016).

[29] G. Finke, A. Gara-Ali, M.-L. Espinouse, V. Jost, et J. Moncel. *Single-machine production and maintenance : a unified approach*. Cahiers Leibniz (205) (2013).

4.1 Introduction

Comme l'indique l'état de l'art, figurant dans la première partie, plusieurs fonctions qui dépendent de la position ont été utilisées, dans la littérature, pour illustrer des effets appliqués sur la machine (tels que l'effet d'apprentissage et l'effet de détérioration). Outre cela, le fait que l'insertion des périodes de maintenance dans le système de production permette de rétablir les conditions initiales de la machine est une hypothèse classique. Les durées de ces périodes de maintenance peuvent être constantes ou intégrer un effet d'apprentissage (l'équipe de maintenance pouvant apprendre de leurs activités de maintenance précédentes) ou intégrer un effet de détérioration (si la période de maintenance est reportée, sa durée peut être allongée).

Un modèle global a été défini, dans le chapitre précédent, qui englobe les problèmes d'ordonnancement avec prise en compte de périodes d'indisponibilité présentés dans l'état de l'art (chapitre 2). Dans ce chapitre, nous proposons une approche de résolution globale de ce modèle dans un atelier à une machine. Plusieurs critères d'optimisation seront étudiés.

Nous commençons par une présentation détaillée du problème étudié, dans la section suivante (section 4.2). La section 4.3 de ce chapitre est consacrée à l'étude du problème avec l'insertion de maintenances sous plusieurs critères d'optimisation. En particulier, nous dédions la section 4.4, à l'étude du problème avec la minimisation du C_{max}. Nous généralisons encore plus notre modèle, dans la section 4.5, avec une configuration dépendante du groupe (cf. section 3.3.2), dans lequel la période de maintenance rétablit partiellement les conditions initiales de la machine. Une conclusion et des perspectives propres à cette étude sont exposées dans la section 4.6.

4.2 Présentation du problème

Dans ce chapitre, nous étudions un problème d'ordonnancement dans un atelier de production à une machine. Soit $J = \{T_1, T_2, \ldots, T_n\}$ un ensemble de tâches à ordonnancer sur la machine. La machine est disponible à l'instant zéro et ne peut réaliser qu'une seule opération à la fois. Les tâches sont toujours disponibles et la préemption n'est pas permise. Les durées des tâches suivent le modèle présenté dans la section 3.2, *i.e.* la durée de la tâche T_j varie selon la position de la tâche dans la séquence et est simplement définie par $p(j, r)$. Pour la présentation du problème, les notations suivantes seront utilisées :

$p(j, r)$ temps opératoire de la tâche T_j ordonnancée en position r,
C_j date d'achèvement de la tâche T_j,
$C_{[r]}$ date d'achèvement de la tâche ordonnancée en position r,

W_j durée d'attente de la tâche T_j,
$W_{[r]}$ durée d'attente de la tâche ordonnancée en position r.

Rappel : Dans un ordonnancement π donné, nous utilisons la notation suivante : l'indice $[r]$ définit la tâche ordonnancée en position r sur la machine et, de même, l'indice $[j]$ définit le rang de la tâche T_j ordonnancée sur la machine.

Des activités de maintenance peuvent être insérées sur la machine pour rétablir les conditions initiales de la machine. Ainsi, les rangs r sont réinitialisés après chaque période de maintenance. Nous supposons que plusieurs périodes de maintenances peuvent être insérées sur la machine. Si k périodes de maintenance sont placées dans un ordonnancement donné, nous obtenons une répartition des tâches en $k+1$ groupes :

$$\{G_1, m_1, G_2, \ldots, G_{k-1}, m_{k-1}, G_k, m_k, G_{k+1}\}$$

avec G_i le groupe des tâches entre les maintenances i et $i-1$, voir figure 4.1.

Figure 4.1 – Maintenances et groupes sur un problème à une machine

La durée de chaque période de maintenance suit le modèle général $\mathcal{MP}$ présenté dans le chapitre précédent (cf. section 3.2.2). Ainsi, la durée de la $i^{\text{ème}}$ période est :

$$m_i = \sum_{j \in G_i} w(j,[j]) p(j,[j]) + \beta_i$$

avec G_i l'ensemble des tâches entre les maintenances $i-1$ et i, G_1 l'ensemble des tâches entre l'instant zéro et la première période de maintenance et β_i la durée élémentaire de la maintenance i et $[j] = r$ définit le rang de la tâche T_j dans le groupe G_i.

L'objectif de ce problème est de trouver simultanément la séquence de tâches, le nombre de maintenances et les positions optimales des maintenances qui minimisent le critère d'optimisation. Nous étudions plusieurs fonctions objectifs ; minimisation du makespan (C_{max}), la somme des dates de fin ($TC = \sum C_j$), la somme des durées d'attente ($TW = \sum W_j$), la variation des dates de fin ($TADC$) et la variation des durées d'attente ($TADW$).

Nous proposons la notation suivante pour le problème d'ordonnancement étudié dans ce chapitre : $1|p(j,r), \mathcal{MP}|\gamma$, où $\gamma = \{C_{max}; \sum C_j; TW; TADC; TADW\}$, $\mathcal{MP}$ le système de maintenances pondérées appliqué sur la machine et $p(j,r)$ le temps opératoire général qui varie selon le rang r de la tâche.

4.3 Approche globale

Nous développons dans cette section une approche générale permettant de résoudre le modèle présenté dans la section précédente.

L'objectif de cette approche est de résoudre le problème $1|p(j,r),\mathcal{MP}|\gamma$, sachant que le nombre de maintenances est une variable de décision à déterminer. Lorsque k périodes de maintenance sont insérées sur la machine, nous obtenons une répartition des tâches en $k+1$ groupes sur la machine $S_k = \{G_1, m_1, G_2, \ldots, G_k, m_k, G_{k+1}\}$ avec G_i le groupe des tâches entre les maintenances i et $i-1$. Soit $\mathcal{N}(n,k)= \{n_i\}_{1\leq i\leq k+1}$ le vecteur qui contient la taille de chaque groupe, avec n_i le nombre de tâches dans G_i $(1 \leq i \leq k+1)$, voir figure 4.1, il est clair que $\sum_{i=1}^{k+1} n_i = n$.

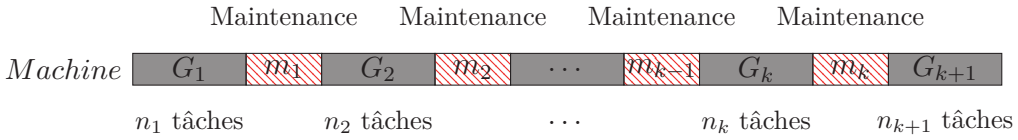

Figure 4.2 – Problème d'ordonnancement à une machine avec k périodes d'indisponibilité

Afin de résoudre le problème, nous considérons d'abord une configuration réduite du problème : le nombre de maintenances k et le vecteur $\mathcal{N}(n,k)$ sont connus d'avance. Ce problème est noté $1|p(j,r),\mathcal{MP}(n_i) = k|\gamma$ ou $1|p(j,r),\mathcal{MP}(n_i)|\gamma$ (le terme « $= k$ » est facultatif, car le nombre de maintenances peut être déduit du vecteur $\mathcal{N}(n,k)$).

Pour une séquence de tâches S_k donnée, on définit une position par le couple :

$$(i : \text{groupe}, r : \text{rang dans le groupe})$$

On note aussi que la permutation est donnée par : $j \to (i,r) = [j]$ et $(i,r) \to j = [i,r]$. Soit $[i,r] = j$ la tâche ordonnancée au rang r $(1 \leq r \leq n_i)$ du groupe G_i $(1 \leq i \leq k+1)$. Ainsi, $p([i,r],r)$ est la durée de la tâche ordonnancée dans le rang r du groupe G_i et $C_{[i,r]}$ sa date d'achèvement, voir figure 4.3.

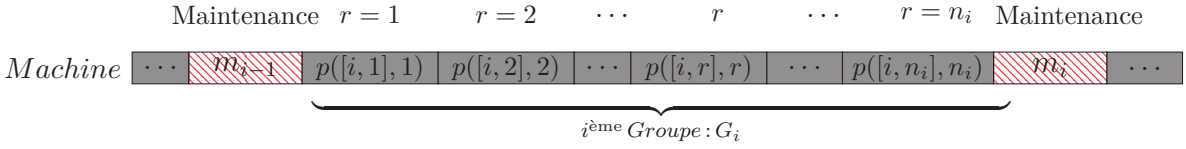

Figure 4.3 – Séquence des tâches dans le groupe G_i

Selon le modèle de la maintenance décrit précédemment, la durée de la $i^{\text{ème}}$ période de maintenance est :

$$m_i = \sum_{j \in G_i} w(j,[j])p(j,[j]) + \beta_i = \sum_{r=1}^{n_i} w([i,r],r)p([i,r],r) + \beta_i \tag{4.1}$$

Dans cette section, nous étudions plusieurs critères d'optimisation, à savoir : la durée totale de l'ordonnancement (Makespan), la somme des dates de fin (TC), la somme des durées d'attente (TW), la variation des dates de fin (*Total absolute deviation of job completion times* $TADC$) et la variation des durées d'attente (*Total absolute deviation of job waiting times* $TADW$).

Nous démontrons, dans ce qui suit, que pour le problème $1|p(j,r), \mathcal{MP}(n_i)|\gamma$, tous les critères d'optimisation γ considérés peuvent être écrits sous la forme suivante :

$$\gamma = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} V_{[i,r]}(i,r) + \Phi(n,k) \tag{4.2}$$

Avec $\Phi(n,k)$ une constante et $V_j(i,r)$ le coût d'affection de la tâche T_j à la position (i,r), *i.e.* le coût d'affectation de la tâche T_j au rang r dans le groupe G_i. Notons que le coût d'affectation de chaque tâche intègre aussi la durée de la maintenance qui lui est associée.

Minimisation du C_{max}

La durée totale de l'ordonnancement C_{max} peut être exprimée de la façon suivante :

$$\begin{aligned} C_{max} &= \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} p([i,r],r) + \sum_{i=1}^{k} m_i = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} p([i,r],r) + \sum_{i=1}^{k} \left(\sum_{r=1}^{n_i} w([i,r],r)p([i,r],r) + \beta_i \right) \\ &= \sum_{i=1}^{k} \sum_{r=1}^{n_i} \left(1 + w([i,r],r)\right) p([i,r],r) + \sum_{r=1}^{n_{k+1}} p([k+1,r],r) + \sum_{i=1}^{k} \beta_i \end{aligned} \tag{4.3}$$

On peut définir $V_j(i,r) = (1 + \lambda_i w(j,r))p(j,r)$ avec $\lambda_i = \begin{cases} 1 & \text{si } i \leq k \\ 0 & \text{si } i = k+1 \end{cases}$ et $\Phi(n,k) = \sum_{i=1}^{k} \beta_i$.

Donc,

$$C_{max} = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} V_{[i,r]}(i,r) + \Phi(n,k)$$

C'est le critère le plus simple, en effet, nous pouvons facilement intégrer la durée de maintenance dans les temps opératoires par l'augmentation du temps opératoire $p(j,r)$ de la tâche T_j, ordonnancée au rang r du groupe G_i, par la durée de maintenance $\lambda_i w(j,r)p(j,r)$ associée à cette tâche, nécessaire dans la période de maintenance suivante,

i.e. :

$$V_j(i,r) = \underbrace{p(j,r)}_{\substack{\text{coût d'affectation classique} \\ \text{pour un problème } C_{max}}} + \underbrace{\lambda_i w(j,r) p(j,r)}_{\substack{\text{coût d'affectation} \\ \text{lié à la maintenance}}}$$

Minimisation de la somme des dates de fin (TC)

La minimisation de la somme des dates de fin est donnée par la formule [1] :

$$\begin{aligned} TC = \sum C_j &= \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} C_{[i,r]} \\ &= \sum_{i=1}^{k} \sum_{h=i+1}^{k+1} n_h \, \beta_i + \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} \left(n - \sum_{h=0}^{i-1} n_h - r + 1 + \sum_{h=i+1}^{k+1} n_h \, w([i,r],r) \right) p([i,r],r) \end{aligned} \tag{4.4}$$

Soit $s_{ir} = r + \sum_{h=0}^{i-1} n_h$, avec $n_0 = 0$, s_{ir} représente le rang global de la tâche sur la machine (r définit le rang dans le groupe) et soit $\lambda_i = \sum_{h=i+1}^{k+1} n_h$, avec $\lambda_{k+1} = 0$. Nous obtenons, ainsi :

$$\sum C_j = \sum_{i=1}^{k} \lambda_i \, \beta_i + \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} \left(n - s_{ir} + 1 + \lambda_i w([i,r],r) \right) p([i,r],r) \tag{4.5}$$

On peut définir $V_j(i,r) = \left(n - s_{ir} + 1 + \lambda_i w(j,r) \right) p(j,r)$ et $\Phi(n,k) = \sum_{i=1}^{k} \lambda_i \, \beta_i$.

Donc, nous pouvons écrire la somme des dates de fin sous la forme :

$$\sum C_j = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} V_{[i,r]}(i,r) + \Phi(n,k)$$

De nouveau, le coût d'affectation $V_j(i,r)$ d'une tâche T_j au rang r dans le groupe G_i intègre la durée de la maintenance qui lui est associée $\lambda_i w(j,r) p(j,r)$, *i.e.* :

$$V_j(i,r) = \underbrace{\left(n - s_{ir} + 1 \right) p(j,r)}_{\substack{\text{coût d'affectation} \\ \text{classique pour un problème } \sum C_j}} + \underbrace{\lambda_i w(j,r) p(j,r)}_{\substack{\text{coût d'affectation} \\ \text{lié à la maintenance}}}$$

Notons aussi que les durées fixes des maintenances β_i sont intégrées dans le terme $\Phi(n,k)$.

1. Voir l'équation détaillée dans l'annexe B.1.

Minimisation de la somme des durées d'attente (TW)

La minimisation de la somme des dates d'attente est donnée par :

$$TW = \sum W_j = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} W_{[i,r]} = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} (C_{[i,r]} - p([i,r],r))$$

Or, en utilisant l'expression 4.4, nous pouvons exprimer la somme des dates d'attentes par

$$\sum W_j = \sum_{i=1}^{k} \sum_{h=i+1}^{k+1} n_h \, \beta_i + \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} \left(n - \sum_{h=0}^{i-1} n_h - r + \sum_{h=i+1}^{k+1} n_h w([i,r],r) \right) p([i,r],r)$$

Soit $s_{ir} = r + \sum_{h=0}^{i-1} n_h$, avec $n_0 = 0$, s_{ir} le rang global de la tâche ordonnancée dans le rang r du groupe G_i et $\lambda_i = \sum_{h=i+1}^{k+1} n_h$, avec $\lambda_{k+1} = 0$. Nous obtenons donc :

$$\sum W_j = \sum_{i=1}^{k} \lambda_i \beta_i + \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} (n - s_{ir} + \lambda_i w([i,r],r)) \, p([i,r],r)$$

On peut définir $V_j(i,r) = (n - s_{ir} + \lambda_i w(j,r)) \, p(j,r)$ et $\Phi(n,k) = \sum_{i=1}^{k} \lambda_i \beta_i$.

Donc, nous pouvons écrire la somme des durées d'attente sous la forme :

$$\sum W_j = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} V_{[i,r]}(i,r) + \Phi(n,k)$$

De nouveau, le coût d'affectation $V_j(i,r)$ d'une tâche T_j au rang r dans le groupe G_i intègre la durée de la maintenance qui lui est associée $\lambda_i w(j,r) p(j,r)$, *i.e.* :

$$V_j(i,r) = \underbrace{(n - s_{ir}) \, p(j,r)}_{\substack{\text{coût d'affectation} \\ \text{classique pour un problème } \sum W_j}} + \underbrace{\lambda_i w(j,r) p(j,r)}_{\substack{\text{coût d'affectation} \\ \text{lié à la maintenance}}}$$

Notons aussi que les durées fixes des maintenances β_i sont intégrées dans le terme $\Phi(n,k)$.

Minimisation de la variation des dates de fin

La minimisation de la variation des dates de fin est exprimée par [2] :

$$\begin{aligned} TADC &= \sum_{h=1}^{n}\sum_{j=h}^{n}|C_j - C_h| \\ &= \sum_{i=1}^{k+1}\sum_{r=1}^{n_i}\left[(s_{ir}-1)(n-s_{ir}+1) + (\sum_{h=1}^{i} n_h)(\sum_{h=i+1}^{k+1} n_h)w([i,r],r)\right] p([i,r],r) \\ &+ \sum_{i=1}^{k}(\sum_{h=1}^{i} n_h)(\sum_{h=i+1}^{k+1} n_h)\beta_i \end{aligned} \tag{4.6}$$

où $s_{ir} = r + \sum_{h=0}^{i-1} n_h$, avec $n_0 = 0$, s_{ir} le rang global de la tâche ordonnancée dans le rang r du groupe G_i.

Soit $\lambda_i = (\sum_{h=1}^{i} n_h)(\sum_{h=i+1}^{k+1} n_h)$, avec $\lambda_{k+1} = 0$. Nous obtenons donc :

$$TADC = \sum_{i=1}^{k+1}\sum_{r=1}^{n_i}\left[(s_{ir}-1)(n-s_{ir}+1) + \lambda_i w([i,r],r)\right] p([i,r],r) + \sum_{i=1}^{k}\lambda_i\beta_i \tag{4.7}$$

On peut donc définir $V_j(i,r) = \left[(s_{ir}-1)(n-s_{ir}+1) + \lambda_i w(j,r)\right] p(j,r)$
et $\Phi(n,k) = \sum_{i=1}^{k}\lambda_i\beta_i$.

Donc, nous pouvons écrire la variation des dates de fin sous la forme :

$$TADC = \sum_{i=1}^{k+1}\sum_{r=1}^{n_i} V_{[i,r]}(i,r) + \Phi(n,k)$$

De nouveau, le coût d'affectation $V_j(i,r)$ d'une tâche T_j au rang r dans le groupe G_i intègre la durée de la maintenance qui lui est associée $\lambda_i w(j,r)p(j,r)$, *i.e.* :

$$V_j(i,r) = \underbrace{(s_{ir}-1)(n-s_{ir}+1)p(j,r)}_{\substack{\text{coût d'affectation}\\ \text{classique pour un problème } TADC}} + \underbrace{\lambda_i w(j,r)p(j,r)}_{\substack{\text{coût d'affectation}\\ \text{lié à la maintenance}}}$$

Notons aussi que les durées fixes des maintenances β_i sont intégrées dans le terme $\Phi(n,k)$.

2. Voir l'équation détaillée dans l'annexe B.2.

Minimisation de la variation des durées d'attente

La minimisation de la variation des durées d'attente est exprimée par [3] :

$$\begin{aligned} TADW &= \sum_{h=1}^{n}\sum_{j=h}^{n}|W_j - W_h| \\ &= \sum_{i=1}^{k+1}\sum_{r=1}^{n_i}\left[s_{ir}(n-s_{ir}) + (\sum_{h=1}^{i} n_h)(\sum_{h=i+1}^{k+1} n_h)w([i,r],r)\right]p([i,r],r) \\ &+ \sum_{i=1}^{k}(\sum_{h=1}^{i} n_h)(\sum_{h=i+1}^{k+1} n_h)\beta_i \end{aligned} \tag{4.8}$$

où $s_{ir} = r + \sum_{h=0}^{i-1} n_h$, avec $n_0 = 0$, s_{ir} le rang global de la tâche ordonnancée dans le rang r du groupe G_i.

Soit $\lambda_i = \sum_{h=1}^{i} n_h \sum_{h=i+1}^{k+1} n_h$, avec $\lambda_{k+1} = 0$. Nous obtenons donc :

$$TADW = \sum_{i=1}^{k+1}\sum_{r=1}^{n_i}\left[s_{ir}(n-s_{ir}) + \lambda_i w([i,r],r)\right]p([i,r],r) + \sum_{i=1}^{k}\lambda_i\beta_i \tag{4.9}$$

On peut donc définir $V_j(i,r) = \left[s_{ir}(n-s_{ir}) + \lambda_i w(j,r)\right]p(j,r)$ et $\Phi(n,k) = \sum_{i=1}^{k}\lambda_i\beta_i$.

Donc, nous pouvons écrire la variation des durées d'attente sous la forme :

$$TADW = \sum_{i=1}^{k+1}\sum_{r=1}^{n_i} V_{[i,r]}(i,r) + \Phi(n,k)$$

De nouveau, le coût d'affectation $V_j(i,r)$ d'une tâche T_j au rang r dans le groupe G_i intègre la durée de la maintenance qui lui est associée $\lambda_i w(j,r)p(j,r)$, *i.e.* :

$$V_j(i,r) = \underbrace{s_{ir}(n-s_{ir})p(j,r)}_{\substack{\text{coût d'affectation} \\ \text{classique pour un problème } TADW}} + \underbrace{\lambda_i w(j,r)p(j,r)}_{\substack{\text{coût d'affectation} \\ \text{lié à la maintenance}}}$$

Notons que les durées fixes des maintenances β_i sont intégrées dans le terme $\Phi(n,k)$.

Remarque 4.1. *Nous pouvons facilement vérifier que lorsqu'aucune période de maintenance n'est insérée sur la machine ($k = 0$), nous obtenons les coûts d'affectation $V_j(i,r)$ classiques d'un problème d'ordonnancement sans maintenance. En effet, pour $k = 0$, nous obtenons un seul groupe des tâches G_1 sur la machine avec $\lambda_1 = 0$ (car $\lambda_{k+1} = 0$). Ainsi, le coût d'affectation lié à la maintenance $\lambda_1 w(j,r)p(j,r)$ est nul.*

Exemple : Dans un problème d'ordonnancement à une machine et sans maintenance ($k = 0$) avec comme objectif la minimisation de la somme des dates de fin, nous avons

3. Voir l'équation détaillée dans l'annexe B.3.

un seul groupe de tâches G_1 et le coût d'affectation d'une tâche au rang r dans ce groupe est :

$$V_j(1,r) = \underbrace{(n - s_{ir} + 1)\, p(j,r)}_{\substack{\text{coût d'affectation} \\ \text{classique pour un problème } \sum C_j}} + \underbrace{\lambda_1 w(j,r) p(j,r)}_{= 0 \text{ car } \lambda_1 = 0} = (n - s_{ir} + 1)\, p(j,r)$$

Ci-dessous, dans le tableau 4.1, nous représentons les coûts d'affectation de chaque critère pour les deux cas, avec et sans maintenance.

Tableau 4.1 – Coût d'affectation de chaque critère, avec et sans maintenance

Critère	Coût d'affectation d'une tâche T_j dans la position (i,r)	
	avec maintenance $(k > 0)$	sans maintenance $(k = 0)$
C_{max}	$(1 + \lambda_i w(j,r))p(j,r)$	$p(j,r)$
$\sum C_j$	$(n - s_{ir} + 1 + \lambda_i w(j,r))p(j,r)$	$(n - s_{ir} + 1)p(j,r)$
$\sum W_j$	$(n - s_{ir} + \lambda_i w(j,r))p(j,r)$	$(n - s_{ir})p(j,r)$
$TADC$	$((s_{ir} - 1)(n - s_{ir} + 1) + \lambda_i w(j,r))p(j,r)$	$(s_{ir} - 1)(n - s_{ir} + 1)p(j,r)$
$TADW$	$(s_{ir}(n - s_{ir}) + \lambda_i w(j,r))p(j,r)$	$s_{ir}(n - s_{ir})p(j,r)$

s_{ir} est le rang global de la tâche ordonnancée au rang r du groupe G_i, si $k = 0$, $s_{ir} = r$.

Pour un nombre de maintenances k donné et un vecteur $\mathcal{N}(n,k) = \{n_i\}_{1 \le i \le k+1}$ connu d'avance, $\Phi(n,k)$ est une constante. Donc, la minimisation de l'équation (4.2) est équivalente à la minimisation de :

$$\sum_{i=1}^{k+1} \sum_{r=1}^{n_i} V_{[i,r]}(i,r) \tag{4.10}$$

Ainsi, le problème $1|p(j,r), \mathcal{MP}(n_i) = k|\gamma$ peut être modélisé sous forme d'un problème d'affectation linéaire $\mathcal{PA}$ de n lignes et n colonnes (cf. section 3.4.1), les lignes correspondent aux tâches et chaque colonne définit une position (i,r) avec $V_j(i,r)$ le coût d'affectation de la tâche T_j au rang r du groupe G_i, voir figure (4.4).

Soit $x_{j,i,r}$ des variables entières exprimant l'affectation des tâches aux positions avec :

$$x_{j,i,r} = \begin{cases} 1 & \text{si la tâche } T_j \text{ est ordonnancée en position } r \text{ dans le groupe } G_i \\ 0 & \text{sinon} \end{cases}$$

Le programme linéaire peut alors s'écrire comme suit (PLNE.3) :

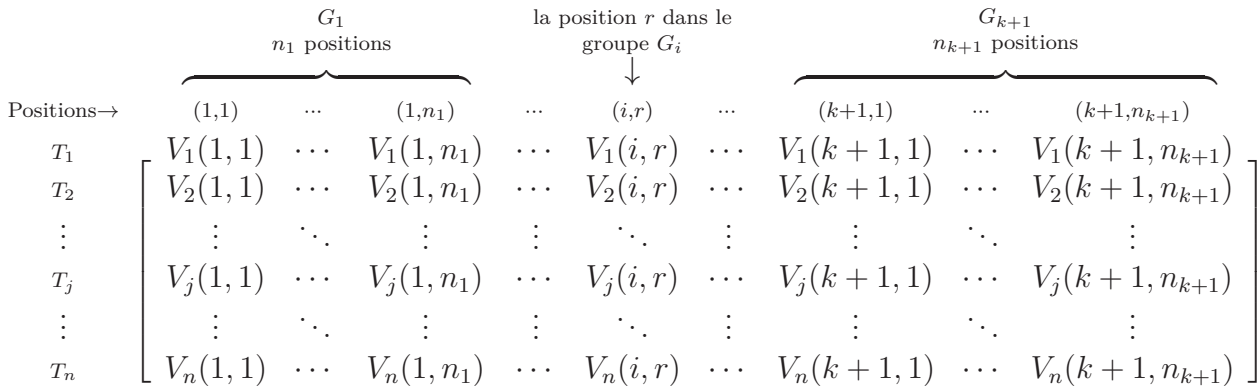

Figure 4.4 – La matrice d'affectation M du problème $1|p(j,r), \mathcal{MP}(n_i) = k|\gamma$

Programme linéaire en nombres entiers 3 (PLNE.3).

$$min \quad \sum_{j=1}^{n} \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} V_j(i,r) x_{j,i,r}$$

$$s.c. \quad \sum_{j=1}^{n} x_{j,i,r} = 1 \quad i = 1,\ldots,k+1, \quad r = 1,\ldots,n_i \tag{4.11}$$

$$\sum_{i=1}^{k+1} \sum_{r=1}^{n_i} x_{j,i,r} = 1 \quad j = 1,\ldots,n \tag{4.12}$$

$$x_{j,i,r} \in \{0,1\} \quad j = 1,\ldots,n, \quad i = 1,\ldots,k+1, \quad r = 1,\ldots,n$$

L'ensemble de contraintes (4.11) exprime le fait que chaque position n'est associée qu'à une seule tâche et l'ensemble de contraintes (4.12) assure que chaque tâche est affectée à une seule et unique position. La résolution d'un problème d'affectation de $n - lignes$ et $n - colonnes$ est polynomiale avec une complexité $\mathcal{O}(n^3)$, voir section 3.4.1.

Donc, nous obtenons le résultat suivant :

Théorème 4.1. *Le problème d'ordonnancement :*
$1|p(j,r), \mathcal{MP}(n_i) = k|\gamma$, *avec* $\gamma = \{C_{max}, \sum C_j, TW, TADC, TADW\}$ *est résolu en temps polynomial avec une complexité* $\mathcal{O}(n^3)$.

Exemple 4.1. *Soit* $n = 5$ *tâches à ordonnancer sur une machine avec* $k = 2$ *périodes de maintenance avec* $\beta_1 = \beta_2 = 1$. *Les tailles des groupes sont* $n_1 = 2$, $n_2 = 2$ *et* $n_3 = 1$. *Nous considérons le critère d'optimisation* C_{max}. *Dans les tableaux ci-dessous, nous présentons les temps opératoires* $p(j,r)$ *et les facteurs de maintenance* $w(j,r)$. *Rappelons que la durée de la maintenance* i *est donnée par l'expression 4.1.*

Tableau 4.2 – Temps opératoires des tâches

	Position r				
$p(j,r)$	1	2	3	4	5
T_1	1	2	3	5	7
T_2	2	4	4,5	6	6
T_3	1	2	2,5	4	5
T_4	1	1,5	2	4	6
T_5	1	1	2	3	4

Tableau 4.3 – Coefficients de maintenance

	Position r				
$w(j,r)$	1	2	3	4	5
T_1	0,2	0,4	0,5	0,8	0,7
T_2	0,5	0,6	0,8	0,9	0,9
T_3	0,4	0,5	0,6	0,7	0,9
T_4	0,5	0,6	0,8	0,9	1
T_5	0,9	1	1,2	1,4	1,5

Minimisation du C_{max} : *En utilisant les coûts d'affectation :* $V_j(i,r) = (1+\lambda_i w(j,r))p(j,r)$ *avec* $\lambda_i = \begin{cases} 1 & si\ i \leq k \\ 0 & si\ i = k+1 \end{cases}$, *nous définissons la matrice d'affectation de taille* 5×5 *suivante :*

$$M = \begin{array}{c} \\ \text{Position (r)}\rightarrow \\ T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{array} \begin{array}{c} \overbrace{}^{G_1}\ \overbrace{}^{G_2}\ \overbrace{}^{G_3} \\ \begin{array}{ccccc} 1 & 2 & 1 & 2 & 1 \end{array} \\ \begin{bmatrix} \boxed{1,2} & 2,8 & 1,2 & 2,8 & 1 \\ 3 & 6,4 & 3 & 6,4 & \boxed{2} \\ 1,4 & 3 & \boxed{1,4} & 3 & 1 \\ 1,5 & \boxed{2,4} & 1,5 & 2,4 & 1 \\ 1,9 & 2 & 1,9 & \boxed{2} & 1 \end{bmatrix} \end{array}$$

La résolution de ce problème d'affectation donne une solution optimale : $V^* = 1,2 + 2,4 + 1,4 + 2 + 2 = 9$, *en ajoutant le terme constant* $\Phi(n,2) = \beta_1 + \beta_2 = 2$, *on obtient* $C^*_{max} = 11$.

L'ordonnancement optimal est le suivant : $\{T_1, T_4, m_1, T_3, T_5, m_2, T_2\}$, *voir le diagramme de Gannt ci-dessous :*

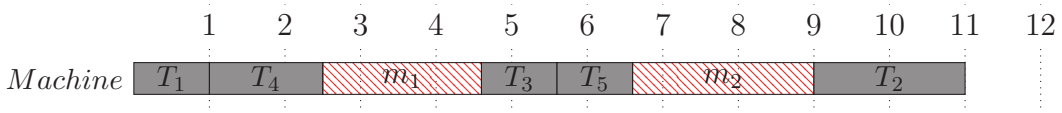

Figure 4.5 – Exemple avec deux périodes de maintenance

Dans la suite, nous considérons que $\gamma = \{C_{max}, \sum C_j, TW, TADC, TADW\}$. On définit par γ_{cost} un critère d'optimisation général généré à partir d'une combinaison linéaire γ tel que $\gamma_{cost} = \mu_1 C_{max} + \mu_2 \sum C_j + \mu_3 \sum TW + \mu_4 TADC + \mu_5 TADW$, avec $\mu_1 \geq 0$, $\mu_2 \geq 0$, $\mu_3 \geq 0$, $\mu_4 \geq 0$, $\mu_5 \geq 0$ des paramètres constants. Par exemple, Mosheiov [86] a étudié un problème d'ordonnancement à une machine avec la minimisation de $\mu_1 C_{max} + (1-\mu_1)TADC$.

Théorème 4.2. *Le problème* $1|p(j,r), \mathcal{MP}(n_i) = k|\gamma_{cost}$ *est résolu polynomialement avec une complexité* $\mathcal{O}(n^3)$.

Démonstration. La preuve est identique à celle du théorème (4.1). □

Jusqu'à présent, nous avons une répartition connue d'avance du nombre de tâches par groupe (n_i tâches pour chaque groupe G_i $(1 \leq i \leq k+1)$), *i.e.* le vecteur $\mathscr{N}(n,k) = \{n_i\}_{1 \leq i \leq k+1}$ est fixé. Lorsque les tailles des groupes sont inconnues, nous générons toutes les combinaisons possibles de $\mathscr{N}(n,k)$ telles que $n_1 + n_2 + \ldots + n_{k+1} = n$. En utilisant le lemme 4.1, le nombre de vecteurs possibles $\mathscr{N}(n,k) = \{n_i\}_{1 \leq i \leq k+1}$ est au plus égal à $\mathcal{O}(n^k)$.

Lemme 4.1. *Le nombre de solutions entières non négatives de l'équation :*

$$x_1 + x_2 + \ldots + x_m = n$$

a une borne supérieure égale à $\frac{(2n)^{m-1}}{(m-1)!} \leq \mathcal{O}(n^{m-1})$.

Démonstration. Le nombre total de partitionnements en nombres entiers non négatifs $x_1 + x_2 + \ldots + x_m = n$ est donné par le coefficient binomial $C(n+m-1, m-1)$, voir Mott et *al.* [91]. Or, il est clair que :

$$C(n+m-1,m-1) = \frac{(n+m-1)(n+m-2)\ldots(n+1)}{(m-1)!} \leq \frac{(n+m-1)^{m-1}}{(m-1)!} \leq \frac{(2n)^{m-1}}{(m-1)!}$$

Or $\frac{(2)^{m-1}}{(m-1)!} \leq 2$, alors :

$$\frac{(2n)^{m-1}}{(m-1)!} \leq \mathcal{O}(n^{m-1})$$

□

Donc, si seulement le nombre de maintenances k sur la machine est connu, nous pouvons résoudre le problème d'ordonnancement $1|p(j,r), \mathcal{MP} = k|\gamma$, avec l'énumération de tous les vecteurs $\mathscr{N}(n,k) = \{n_i\}_{1 \leq i \leq k+1}$ possibles, ce qui nous amène à l'algorithme 1.

Théorème 4.3. *Le problème* $1|p(j,r), \mathcal{MP} = k|\gamma, \gamma_{cost}$ *avec* k *périodes de maintenance est résolu polynomialement avec une complexité* $\mathcal{O}(n^{k+3})$.

Démonstration. Dans l'algorithme 1, l'étape 1.2 consiste à résoudre le problème d'affectation $PLNE.3$, cette étape est résolue avec une complexité $\mathcal{O}(n^3)$. L'étape 1 est exécutée au plus $\frac{(2n)^k}{k!}$ fois, or $\frac{2^k}{k!} \leq 2$, donc l'étape 1 est exécutée $\mathcal{O}(n^k)$ fois (voir lemme 4.1), d'où la complexité totale de l'algorithme : $\mathcal{O}(n^{k+3})$. □

Algorithme 1 : Problème à une machine avec k maintenances $1|p(j,r), \mathcal{MP} = k|\gamma$

Données :
- k le nombre de maintenances à insérer
- $\mathscr{N}_k$ ensemble de vecteurs $\mathscr{N}(n,k)$ possibles tels que $n_1 + n_2 + \ldots + n_{k+1} = n$
- $p(j,r)$ temps opératoire de la tâche T_j $(1 \leq j \leq n)$ ordonnancée à la position r
- $w(j,r)$ coefficients de maintenance
- β_i durée élémentaire de la maintenance

Résultat : $\gamma^*(k)$ la solution optimale

début

1 **pour chaque** $\mathscr{N}(n,k) \in \mathscr{N}_k$ **faire**

`/* Résoudre` $1|p(j,r), \mathcal{MP}(n_i) = k|\gamma$ `*/`

1.1 Calculer les facteurs λ_i. Définir la matrice des coûts M de taille $n \times n$.

1.2 $V^\gamma \longleftarrow$ *Résoudre* $\mathcal{PA}$ (PLNE.3)

1.3 $\gamma^*(\mathscr{N}(n,k)) \longleftarrow V^\gamma + \Phi(n,k)$, avec $\Phi(n,k) = \sum_{i=1}^{k} \lambda_i \beta_i$

fin pour

2 **retourner** $\gamma^*(k) \longleftarrow \min_{\mathscr{N}(n,k) \in \mathscr{N}_k} \{\gamma^*(\mathscr{N}(n,k))\}$

fin

Remarque 4.2. *Pour des raisons budgétaires, une borne supérieure sur le nombre de maintenances peut être ajoutée. On note k_0 le nombre de maintenances maximal à insérer sur la machine et on considère le problème $1|p(j,r), \mathcal{MP} \leq k_0|\gamma$. En utilisant l'algorithme ci-dessous, basé sur le théorème 4.3, nous pouvons résoudre le problème avec une complexité $\mathcal{O}(n^{k_0+3})$.*

début

1 **pour chaque** $k = 1, \ldots, k_0$ **faire**

1.1 $\gamma^*(k) \longleftarrow$ *Résoudre* $1|p(j,r), \mathcal{MP} = k|\gamma$

fin pour

2 **retourner** $\gamma^* \longleftarrow \min_{k=1,\ldots,k_0} \{\gamma^*(k)\}$

fin

Les résultats évoqués dans le théorème 4.3 et dans la remarque 4.2 généralisent des résultats antérieurs [119, 135]. Par exemple, Yang et Yang [135] ont étudié un problème avec une configuration plus restrictive : deux modèles de temps opératoires ont été étudiés $p(j,r) = p_j r^{a_j}$ et $p(j,r) = p_j + a_j r$, où $a_j > 0$, avec une durée de maintenance linéaire $(w(j,r) = \alpha$ et $\beta_i = \beta)$. L'objectif est de trouver simultanément la séquence des tâches qui minimise la somme des dates de fin $\sum C_j$. Le tableau 4.4 fournit une comparaison entre les résultats obtenus dans cette section et les résultats antérieurs.

Tableau 4.4 – Comparaison avec les résultats antérieurs

Notre modèle	Modèle et complexité de la littérature		
	Réf.	Modèle	Complexité
∘ Critère d'optimisation : γ ∘ Temps opératoires : $p(j,r)$ ∘ Système de maintenance $\mathcal{MP}$: $m_i = \sum_{j \in G_i} w(j,[j])p(j,[j]) + \beta_i$ ∘ k périodes de maintenance ⇒ Complexité : $\mathcal{O}(n^{k+3})$ (théorème 4.3) ⇒ Si k_0 le nombre maximum de maintenances, la complexité est : $\mathcal{O}(n^{k_0+3})$ (remarque 4.2)	[135]	∘ $\sum_{j=1}^{n} C_j$ ∘ $p(j,r) = p_j r^{a_j}$, avec $a_j > 0$ ∘ $w(j,r) = \alpha$ et $\beta_i = \beta$ ∘ k_0 nombre maximum de maintenances	$\mathcal{O}(n^{k_0+3})$
	[135]	∘ $\sum_{j=1}^{n} C_j$ ∘ $p(j,r) = p_j + a_j r$, avec $a_j > 0$ ∘ $w(j,r) = \alpha$ et $\beta_i = \beta$ ∘ k_0 nombre maximum de maintenances	$\mathcal{O}(n^{k_0+3})$
	[119]	∘ C_{max} et $\sum_{j=1}^{n} C_j$ ∘ $p(j,r) = (1+a_j)^r p_j$, avec $a_j \geq 0$ ∘ $w(j,r) = 0$ ∘ k_0 nombre maximum de maintenances	$\mathcal{O}(n^{k_0+3})$

4.4 Minimisation du C_{max}

Dans la section précédente, une approche générale a été développée pour résoudre le problème d'ordonnancement de tâches et de périodes de maintenance sous plusieurs critères d'optimisation. Nous nous intéressons, dans cette section, à la minimisation du C_{max} sous un effet de détérioration :

$$p(j,1) \leq p(j,2) \leq \ldots \leq p(j,n), \quad \forall\, j = 1, \ldots, n \tag{4.13}$$

De plus, nous considérons que les coefficients $w(j,r)$ du système de maintenance $\mathcal{MP}$, défini dans le chapitre 3, sont croissants selon r :

$$w(j,1) \leq w(j,2) \leq \ldots \leq w(j,n), \quad \forall\, j = 1, \ldots, n \tag{4.14}$$

Nous étudions donc le problème $1|p(j,r) - det, \mathcal{MP}|C_{max}$, dont la complexité peut être considérablement améliorée. Précédemment, et pour un nombre de maintenance k donné, nous avons montré, dans le théorème 4.1, que la minimisation du C_{max} est polynomiale lorsque le vecteur $\mathcal{N}(n,k) = \{n_i\}_{1 \leq i \leq k+1}$ est connu d'avance, *i.e.* la taille de chaque groupe est une donnée. Dans cette section, nous ne considérons plus le vecteur $\mathcal{N}(n,k) = \{n_i\}_{1 \leq i \leq k+1}$ et nous étendons la taille de chaque groupe à n emplacements. Alors, pour k périodes de maintenance, le problème peut être modélisé sous forme d'un programme linéaire qui consiste à affecter n tâches à $n(k+1)$ positions ($k+1$ groupes de

taille n chacun), voir figure 4.6.

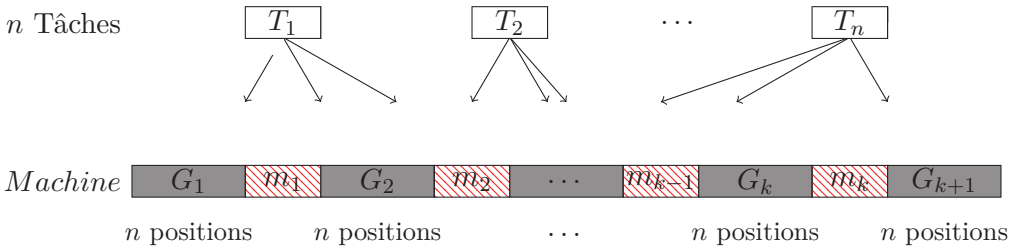

Figure 4.6 – Maintenances et groupes dans un problème à une machine

Il est clair que le nombre de positions disponibles sur la machine $n(k+1)$ est supérieur au nombre de tâches n à affecter. Le coût d'affectation d'une tâche j à une position donnée (i : groupe G_i, r : position) est :

$$V_j(i,r) = (1 + \lambda_i w(j,r))p(j,r), \text{ avec } \lambda_i = \begin{cases} 1 & \text{si } i \leq k \\ 0 & \text{si } i = k+1 \end{cases}$$

Le programme linéaire peut alors s'écrire comme suit :

Programme linéaire en nombres entiers 4 (PLNE.4)**.**

$$min \quad \sum_{j=1}^{n} \sum_{i=1}^{k+1} \sum_{r=1}^{n} V_j(i,r) x_{j,i,r}$$

$$s.c. \quad \sum_{i=1}^{k+1} \sum_{r=1}^{n} x_{j,i,r} = 1 \quad j = 1, \ldots, n \tag{4.15}$$

$$\sum_{j=1}^{n} x_{j,i,r} \leq 1 \quad i = 1, \ldots, k+1, \quad r = 1, \ldots, n \tag{4.16}$$

$$\sum_{j=1}^{n} x_{j,i,1} = 1 \quad i = 1, \ldots, k+1 \tag{4.17}$$

$$x_{j,i,1} \geq x_{j,i,2} \geq \ldots \geq x_{j,i,n} \quad j = 1, \ldots, n,\ i = 1, \ldots, k+1 \tag{4.18}$$

$$x_{j,i,r} \in \{0,1\} \quad j = 1, \ldots, n,\ i = 1, \ldots, k+1,\ r = 1, \ldots, n$$

L'ensemble de contraintes (4.15) assure que chaque tâche est affectée à une seule et unique position et l'ensemble de contraintes (4.16) exprime le fait que chaque position est associée au plus à une seule tâche. L'ensemble de contraintes (4.17) garantit que la première position de chaque groupe est occupée par une tâche. Les contraintes (4.18)

assurent que les positions occupées sont au début du groupe et les positions non occupées sont à la fin du groupe, voir figure 4.7.

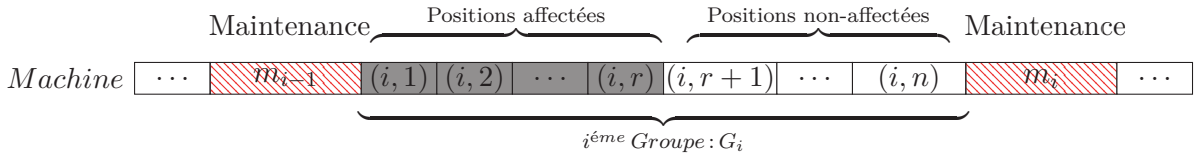

Figure 4.7 – Affectation des tâches dans le groupe G_i

Or, pour tout $j = 1, 2, \ldots, n$ on a :

$$p(j,1) \leq p(j,2) \leq \ldots \leq p(j,n) \qquad et \qquad w(j,1) \leq w(j,2) \leq \ldots \leq w(j,n)$$

Ainsi, les coûts d'affectation $V_j(i,r)$ dans un groupe G_i, sont croissants avec la position r :

$$V_j(i,1) \leq V_j(i,2) \leq \ldots \leq V_j(i,n), \qquad \forall\, j = 1, \ldots, n \; et \; i = 1, \ldots, k+1$$

Donc, il est clair que dans l'ordonnancement optimal, les tâches sont affectées à partir de $r = 1$ et les tâches sont sélectionnées dans des positions adjacentes, *i.e.* les positions affectées se trouvent au début du groupe et les positions non affectées sont à la fin du groupe, voir figure 4.7. Donc, les contraintes (4.18) sont redondantes et peuvent être retirées du programme linéaire. Le problème est reformulé sous forme d'un problème d'affectation rectangulaire $\mathcal{PAR}$ (cf. section 3.4.2) :

Programme linéaire en nombres entiers 5 (PLNE.5).

$$min \quad \sum_{j=1}^{n} \sum_{i=1}^{k+1} \sum_{r=1}^{n} V_j(i,r) x_{j,i,r}$$

$$s.c. \quad \sum_{i=1}^{k+1} \sum_{r=1}^{n} x_{j,i,r} = 1 \quad j = 1, \ldots, n \tag{4.19}$$

$$\sum_{j=1}^{n} x_{j,i,r} \leq 1 \quad i = 1, \ldots, k+1, \quad r = 1, \ldots, n \tag{4.20}$$

$$x_{j,i,r} \in \{0,1\} \qquad j = 1, \ldots, n,\; i = 1, \ldots, k+1,\; r = 1, \ldots, n$$

Notons que dans la solution obtenue pour ce problème d'affectation $\mathcal{PAR}$ certains groupes peuvent être vides, à cause de la suppression des contraintes (4.17). Alors, la solution du $\mathcal{PAR}$ doit être modifiée, en supprimant les groupes vides et en décalant le

dernier groupe, on obtient une solution faisable et optimale[4]. Soit $V^{C_{max}}$ la solution obtenue par le problème d'affectation $\mathcal{PAR}$. La solution optimale C^*_{max} est donc égale à :

$$C^*_{max} = V^{C_{max}} + \Phi(n,k)$$

avec $\Phi(n,k) = \sum_{i=1}^{k'} \beta_i$, où k' est le nouveau nombre de maintenances après la suppression des groupes vides.

Ainsi, en utilisant l'algorithme proposé par Milo et *al.* [82], nous pouvons résoudre le problème d'affectation rectangulaire de taille $n \times (k+1)n$ avec une complexité $\mathcal{O}(n^3 + n^2(k+1))$ soit $\mathcal{O}(n^3)$, pour tout $k < n$ (cf. section 3.4.2).

Théorème 4.4. *Si le nombre de maintenances à insérer k est donné, la résolution du problème $1|p(j,r) - det, \mathcal{MP} = k|C_{max}$ est polynomiale avec une complexité $\mathcal{O}(n^3)$.*

Lorsque le nombre de maintenances k est une variable de décision, le problème d'ordonnancement $1|p(j,r) - det, \mathcal{MP}|C_{max}$ peut être résolu en temps polynomial, avec l'énumération de tous les nombres k possibles avec k variant entre 1 et $n-1$ (la maintenance est interdite au début et à la fin de l'ordonnancement). Ainsi, le problème peut être résolu avec une complexité $\mathcal{O}(n^4)$ en utilisant l'algorithme 2 suivant :

Algorithme 2 : Minimisation du C_{max} sur une machine : $1|p(j,r) - det, \mathcal{MP}|C_{max}$

Données :
- $p(j,r)$ temps opératoire de la tâche T_j ordonnancée au rang r
- $w(j,r)$ coefficients de maintenance et β_i durée élémentaire de la maintenance i

Résultat : C^*_{max} la solution optimale

début

1 **pour chaque** $k = \{1, 2, \ldots, n-1\}$ **faire**

`/* Résoudre` $1|p(j,r) - det, \mathcal{MP} = k|C_{max}$ `*/`

1.1 $V^{C_{max}} \longleftarrow$ *Résoudre* $\mathcal{PAR}$ (PLNE.5) et supprimer les groupes vides

1.2 Calculer $\Phi(n,k)$ avec la bonne valeur de k après la suppression des groupes vides

1.3 $C^*_{max}(k) \longleftarrow V^{C_{max}} + \Phi(n,k)$

fin pour

2 **retourner** $C^*_{max} \longleftarrow \min_{1 \leq k \leq n-1}\{C^*_{max}(k)\}$

fin

Théorème 4.5. *La résolution du problème $1|p(j,r) - det, \mathcal{MP}|C_{max}$ est polynomiale avec une complexité $\mathcal{O}(n^4)$.*

4. Notons qu'en supprimant et décalant les groupes, nous obtenons un ordonnancement de qualité égale ou encore mieux avec un nombre de maintenances plus petit que k.

Remarque 4.3. *On note que le Théorème 4.5 ne se limite pas à la minimisation de C_{max}, il est vrai aussi pour tous critères d'optimisation qui vérifient la propriété :*

$$V_j(i,1) \leq V_j(i,2) \leq \ldots \leq V_j(i,n), \qquad \forall\, j = 1,\ldots,n \text{ et } i = 1,\ldots,k+1$$

On précise que les critères $\{\sum C_j, TW, TADC, TADW\}$ ne vérifient pas cette propriété.

Afin de mieux comprendre ce modèle, nous considérons l'exemple suivant pour illustrer la méthode de résolution.

Exemple 4.2. *Soit $n = 4$ tâches à ordonnancer sur une machine avec $k = 2$ périodes de maintenance avec $\beta_1 = \beta_2 = 1$. Notre objectif est de résoudre le problème d'ordonnancement $1|p(j,r) - det, \mathcal{MP} = 2|C_{max}$. Dans les tableaux ci-dessous, nous présentons les temps opératoires $p(j,r)$ et les facteurs de maintenance $w(j,r)$.*

Tableau 4.5 – Temps opératoires des tâches

	Position r			
$p(j,r)$	1	2	3	4
T_1	1	1,2	1,3	5
T_2	2	4	4,5	6
T_3	1	2	2,5	4
T_4	1	1	2	3

Tableau 4.6 – Coefficients de maintenance

	Position r			
$w(j,r)$	1	2	3	4
T_1	0,4	0,6	0,7	0,8
T_2	0,7	0,8	0,9	1
T_3	0,5	0,6	0,6	0,7
T_4	0,9	1	1,2	1,4

Pour résoudre le problème :

1. *Tout d'abord, nous construisons la matrice des coûts d'affectation $M = \{V_j(i,r)\}$, de taille $4 \times (4 \cdot 3) = 4 \times 12$.*

$$M = \begin{array}{c} \\ \text{Position (r)}\rightarrow \\ T_1 \\ T_2 \\ T_3 \\ T_4 \end{array} \begin{array}{c} \overbrace{}^{G_1} \overbrace{}^{G_2} \overbrace{}^{G_3} \\ \begin{array}{cccc|cccc|cccc} 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 \end{array} \\ \left[\begin{array}{cccc|cccc|cccc} 1,4 & 1,92 & 2,21 & 9 & 1,4 & 1,92 & 2,21 & 9 & 1 & 1,2 & \boxed{1,3} & 5 \\ 3,4 & 7,2 & 8,55 & 12 & 3,4 & 7,2 & 8,55 & 12 & \boxed{2} & 4 & 4,5 & 6 \\ \boxed{1,5} & 3,2 & 4 & 6,8 & 1,5 & 3,2 & 4 & 6,8 & 1 & 2 & 2,5 & 4 \\ 1,9 & 2 & 4,4 & 7,2 & 1,9 & 2 & 4,4 & 7,2 & 1 & \boxed{1} & 2 & 3 \end{array}\right] \end{array}$$

2. *En utilisant le problème d'affectation rectangulaire $\mathcal{PAR}(PLNE.5)$, la solution $V^{C_{max}} = 5,8$ est obtenue par l'ordonnancement suivant :*

$$(\underbrace{T_3,\ vide, vide, vide}_{G_1}; m_1; \underbrace{vide, vide, vide, vide}_{G_2}; m_2; \underbrace{T_2,\ T_4,\ T_1,\ vide}_{G_3})$$

Comme cela a été précisé avant, les tâches dans chaque groupe sont affectées au début du groupe, car les coûts d'affectation sont décroissants selon r. Le groupe G_2 est vide, ainsi l'ordonnancement est faisable en supprimant le groupe G_2 et en décalant le groupe G_3. L'ordonnancement faisable et optimal est donc donné par :

$$\textit{Séquence optimale} = (T_3; m_1; T_2, T_4, T_1) \textit{ avec une seule période de maintenance}$$

3. *En ajoutant la partie constante de la maintenance* $\Phi(n,k) = \sum_{i=1}^{1} \beta_1 = 1$, *nous obtenons la solution optimale :*

$$C^*_{max} = V^{C_{max}} + \Phi(n,k) = 5,8 + 1 = 6,8$$

Les résultats obtenus dans cette section généralisent et améliorent des travaux antérieurs. Le tableau 4.7 fournit une comparaison entre les résultats obtenus dans cette section et les résultats antérieurs.

Tableau 4.7 – Comparaison avec les résultats antérieurs

Notre modèle	Modèle et complexité de la littérature		
	Réf.	Modèle	Complexité
∘ Minimisation du C_{max} ∘ Temps opératoires avec détérioration : $p(j,1) \leq p(j,2) \leq \ldots \leq p(j,n)$ ∘ Système de maintenance $\mathcal{MP}$: $m_i = \sum_{j \in G_i} w(j,[j])p(j,[j]) + \beta_i$ avec $w(j,1) \leq \ldots \leq w(j,n)$ ⇒ Complexité : $\mathcal{O}(n^4)$ (théorème 4.5)	[138]	∘ Minimisation du C_{max} ∘ $p(j,r) = p_j r^{a_j}$, avec $a_j \geq 0$ ∘ $w(j,r) = 0$ et $\beta_i = \beta$	$\mathcal{O}(n^4)$
	[134]	∘ Minimisation du C_{max} ∘ $p(j,r) = p_j r^{a_j}$, avec $a_j \geq 0$ ∘ $w(j,r) = 0$ et β_i croissant selon i	$\mathcal{O}(n^4)$
	[134]	∘ Minimisation du C_{max} ∘ $p(j,r) = p_j r^{a_j}$, avec $a_j \geq 0$ ∘ $w(j,r) = \alpha$ et $\beta_i = \beta$	$\mathcal{O}(n^5)$

4.5 Ordonnancement dépendant du groupe

Dans cette section, nous étudions une version étendue du modèle principal (cf. section 3.3.2). Dans ce modèle, on définit un environnement de réalisation différent pour chaque groupe. Ainsi, le problème étudié est le suivant : Soit $J = \{T_1, T_2, \ldots, T_n\}$ un ensemble de n tâches à ordonnancer sur une machine. La machine est disponible à l'instant zéro et ne peut réaliser qu'une seule opération à la fois. Les tâches sont toujours dispo-

nibles et la préemption n'est pas permise. Soit $p^{(i)}(j,r)$ le temps opératoire de la tâche T_j ordonnancée en position r dans le groupe G_i. Des périodes de maintenance peuvent être insérées sur la machine. Ces dernières suivent le système de maintenance $\mathcal{MP}^{(i)}$, déjà défini dans la section 3.3.2 : la tâche T_j ordonnancée en position r dans le groupe G_i augmente la durée de la maintenance réalisée par l'équipe i avec une valeur $w^{(i)}(j,r)p^{(i)}(j,r)$.

En utilisant la nouvelle configuration du problème (temps opératoires $p^{(i)}(j,r)$ et coefficients de maintenance $w^{(i)}(j,r)$ dépendants du groupe), et pour k et $\mathcal{N}(n,k)=\{n_i\}_{1\leq i\leq k+1}$ donnés nous pouvons toujours réécrire le critère d'optimisation γ sous la forme :

$$\gamma = \sum_{i=1}^{k+1}\sum_{r=1}^{n_i} V_{[i,r]}(i,r) + \Phi(n,k) \tag{4.21}$$

Rappelons que l'indice $[i,r]=j$ indique que la tâche T_j est ordonnancée au rang r du groupe G_i et $V_j(i,r)$ le coût d'affectation de la tâche T_j à la position : (groupe G_i, rang r) et $\Phi(n,k)$ une constante. En utilisant le même principe de la section 4.3, nous obtenons :

Minimisation du C_{max}

La durée totale de l'ordonnancement C_{max} peut être exprimée sous la forme de l'expression 4.21 avec : $V_j(i,r) = (1+\lambda_i w^{(i)}(j,r))p^{(i)}(j,r)$ avec $\lambda_i = \begin{cases} 1 & \text{si } i \leq k \\ 0 & \text{si } i = k+1 \end{cases}$ et $\Phi(n,k) = \sum_{i=1}^{k}\beta_i$

Minimisation de la somme des dates de fin

La minimisation de la somme des dates de fin donnée par $\sum C_j = \sum_{i=1}^{k+1}\sum_{r=1}^{n_i} C_{[i,r]}$ peut être exprimée sous la forme de l'expression 4.21 avec :

$V_j(i,r) = \left(n - s_{ir} + 1 + \lambda_i w^{(i)}(j,r)\right)p^{(i)}(j,r)$ et $\Phi(n,k) = \sum_{i=1}^{k}\lambda_i\,\beta_i$.

où $s_{ir} = r + \sum_{h=0}^{i-1} n_h$, avec $n_0 = 0$, s_{ir} représente le rang global de la tâche sur la machine (r définit le rang dans le groupe) et $\lambda_i = \sum_{h=i+1}^{k+1} n_h$, avec $\lambda_{k+1} = 0$.

Minimisation de la somme des durées d'attente

De même, la minimisation de la somme des durées d'attente est exprimée sous la forme de l'expression 4.21 avec :

$V_j(i,r) = \left(n - s_{ir} + \lambda_i w^{(i)}(j,r)\right)p^{(i)}(j,r)$ et $\Phi(n,k) = \sum_{i=1}^{k}\lambda_i\beta_i$.

où $s_{ir} = r + \sum_{h=0}^{i-1} n_h$, avec $n_0 = 0$, s_{ir} représente le rang global de la tâche sur la machine (r définit le rang dans le groupe) et $\lambda_i = \sum_{h=i+1}^{k+1} n_h$, avec $\lambda_{k+1} = 0$.

Minimisation de la variation des dates de fin

La variation des dates de fin $TADC$ peut être exprimée sous la forme de l'expression 4.21

avec :

$V_j(i,r) = \Big[(s_{ir}-1)(n-s_{ir}+1) + \lambda_i w^{(i)}(j,r)\Big]\, p^{(i)}(j,r)$ et $\Phi(n,k) = \sum_{i=1}^{k} \lambda_i \beta_i$.

où $s_{ir} = r + \sum_{h=0}^{i-1} n_h$, avec $n_0 = 0$, s_{ir} le rang global de la tâche ordonnancée au rang r du groupe G_i et $\lambda_i = \sum_{h=1}^{i} n_h \sum_{h=i+1}^{k+1} n_h$, avec $\lambda_{k+1} = 0$.

Minimisation de la variation des durées d'attente

La variation des durées d'attente $TADW$ peut être exprimée sous la forme de l'expression 4.21 avec :

$V_j(i,r) = \Big[s_{ir}(n-s_{ir}) + \lambda_i w^{(i)}(j,r)\Big]\, p^{(i)}(j,r)$ et $\Phi(n,k) = \sum_{i=1}^{k} \lambda_i \beta_i$.

où $s_{ir} = r + \sum_{h=0}^{i-1} n_h$, avec $n_0 = 0$, s_{ir} le rang global de la tâche ordonnancée dans le rang r du groupe G_i et $\lambda_i = \sum_{h=1}^{i} n_h \sum_{h=i+1}^{k+1} n_h$, avec $\lambda_{k+1} = 0$.

De même que précédemment, lorsque le nombre de maintenances k et la taille de chaque groupe $\mathscr{N}(n,k) = \{n_i\}_{1\leq i\leq k+1}$ sont connus d'avance, nous pouvons résoudre le problème d'ordonnancement $1|p^{(i)}(j,r), \mathcal{MP}^{(i)}(n_i) = k|\gamma$ en utilisant le problème d'affectation linéaire $\mathcal{PA}$ (P.L.3), défini dans la section 4.3.

Les contraintes (4.11) et (4.12) du P.L.3 représentent les contraintes classiques d'un problème d'affectation, *i.e.* chaque tâche est ordonnancée une seule fois et chaque position est occupée par une seule tâche.

Ce problème d'affectation $\mathcal{PA}$ peut être résolu en temps polynomial avec une complexité $\mathcal{O}(n^3)$, voir Dinic et Kronrod [26] (cf. section 3.4.1). Soit V^γ la solution obtenue par P.L.3, alors la solution optimale du problème est : $\gamma^*(\mathscr{N}(n,k)) = V^\gamma + \Phi(n,k)$.

Théorème 4.6. *Si le nombre de maintenances et les tailles des groupes sont connus d'avance, le problème* $1|p^{(i)}(j,r), \mathcal{MP}^{(i)}(n_i) = k|\gamma$, $\gamma = \{C_{max}, \sum C_j, TW, TADC, TADW\}$ *peut être résolu en temps polynomial avec une complexité* $\mathcal{O}(n^3)$.

Pour trouver la solution optimale du problème $1|p^{(i)}(j,r), \mathcal{MP}^{(i)} = k|\gamma$, *i.e.* lorsque seulement le nombre de périodes de maintenance k est connu d'avance, il suffit d'énumérer tous les vecteurs $\mathscr{N}(n,k) = \{n_i\}_{1\leq i\leq k+1}$ possibles, tel que $n_1 + \ldots + n_{k+1} = n$, et de résoudre à chaque fois le problème d'affectation (PLNE.3). D'après le lemme 4.1, le nombre de vecteurs possible est $\mathcal{O}(n^k)$ avec k le nombre de maintenances à insérer. D'où la complexité totale de résolution $\mathcal{O}(n^{k+3})$, voir Algorithme 1.

Théorème 4.7. *La résolution du problème* $1|p^{(i)}(j,r), \mathcal{MP}^{(i)} = k|\gamma$, *avec k périodes de maintenance à insérer, est polynomiale avec une complexité* $\mathcal{O}(n^{k+3})$, *en utilisant l'algorithme 1.*

Remarque 4.4. *En utilisant la même analyse que celle de la section 4.4, nous pouvons montrer que le Théorème 4.5 est toujours vrai pour la minimisation du C_{max} avec une*

configuration dépendante du groupe ($w^{(i)}(j,r)$ *et* $p^{(i)}(j,r)$). *Cependant, deux conditions supplémentaires doivent être ajoutées :*

$$p^{(1)}(j,r) \leq p^{(2)}(j,r) \leq \ldots \leq p^{(n)}(j,r), \quad \forall\, j \text{ et } r \tag{4.22}$$

$$w^{(1)}(j,r) \leq w^{(2)}(j,r) \leq \ldots \leq w^{(n)}(j,r), \quad \forall\, j \text{ et } r \tag{4.23}$$

Ces deux conditions assurent que la solution obtenue par le $\mathcal{PAR}$ *(PLNE.5) est faisable. Avec les conditions (4.22) et (4.23) les groupes non vides précèdent les groupes vides (sauf le dernier groupe* G_{k+1}*). En supprimant les blocs vides et en décalant le dernier bloc (groupe* G_{k_l+1}*), on obtient la solution optimale*[5]*. Ainsi, nous obtenons l'extension suivante du théorème 4.5 :*

Théorème 4.8. *La résolution du problème* $1|p^{(i)}(j,r) - det, \mathcal{MP}^{(i)}|C_{max}$ *est polynomiale avec une complexité* $\mathcal{O}(n^4)$.

Ce résultat généralise et améliore les résultats obtenus dans les travaux de Rustogi et Strusevich [99]. Les auteurs ont étudié un modèle réduit avec $w^{(i)}(j,r) = \alpha_i$ *et* $p^{(i)}(j,r) = p_j g(j,i,r)$*, ils ont montré que le problème est polynomial en proposant une approche de résolution de complexité* $\mathcal{O}(n^5)$*. Ainsi, le Théorème 4.8, généralise et améliore le résultat de [99].*

Tableau 4.8 – Comparaison avec les résultats de Rustogi et Strusevich [99]

Notre modèle	Modèle et complexité de la littérature		
	Réf.	Modèle	Complexité
∘ Minimisation du C_{max} ∘ Temps opératoires avec détérioration : $p^{(i)}(j,1) \leq \ldots \leq p^{(i)}(j,n)$ ∘ Système de maintenance $\mathcal{MP}$: $m_i = \sum_{j \in G_i} w^{(i)}(j,[j])p^{(i)}(j,[j]) + \beta_i$ avec $w^{(i)}(j,1) \leq \ldots \leq w^{(i)}(j,n)$ ∘ Deux conditions : (4.22) et (4.23) ⇒ Complexité : $\mathcal{O}(n^4)$, voir théorème 4.8	[99]	∘ Minimisation du C_{max} ∘ $p^{(i)}(j,r) = p_j g(j,i,r)$ ∘ $w^{(i)}(j,r) = 0$	$\mathcal{O}(n^4)$
	[99]	∘ Minimisation du C_{max} ∘ $p^{(i)}(j,r) = p_j g(j,r)$ ∘ $w^{(i)}(j,r) = \alpha_i$	$\mathcal{O}(n^4)$
	[99]	∘ Minimisation du C_{max} ∘ $p^{(i)}(j,r) = p_j g(j,i,r)$ ∘ $w^{(i)}(j,r) = \alpha_i$	$\mathcal{O}(n^5)$

5. Remarquons que cette modification de la solution ne peut qu'améliorer la solution obtenue par le $\mathcal{PAR}$: supprimer les groupes vides pour obtenir $k' \leq k$ périodes de maintenance (amélioration) et décaler le dernier groupe au groupe $k'+1$ (amélioration supplémentaire, car $w^{(i)}(j,r)$ et $p^{(i)}(j,r)$ sont croissants avec i).

4.6 Conclusion

Dans ce chapitre, nous nous sommes intéressés au problème d'ordonnancement de tâches et de maintenances sur une machine. Les temps opératoires sont dépendants de la position $p(j,r)$ et contrairement à la littérature aucune fonction spécifique n'est imposée pour ces temps opératoires. Un nouveau modèle de maintenance général $\mathcal{MP}$ a été utilisé qui englobe les différents modèles de la littérature. L'objectif est de trouver la séquence de tâches optimale et les positions de maintenances qui permettent de minimiser un critère d'optimisation γ.

Grâce au nouveau modèle de maintenance utilisé, nous avons proposé une approche globale pour résoudre ce problème. Dans le cas le plus général, il suffit de résoudre $\mathcal{O}(n^k)$ fois le problème d'affectation et de trouver ainsi la meilleure solution. Notons ici que cette approche est vraie pour toutes les fonctions $p(j,r)$, *i.e.* aucune tendance n'est imposée, par exemple la propriété est vraie pour les machines avec un effet de détérioration et/ou un effet d'apprentissage.

Nous avons montré ensuite, que le nombre de problèmes d'affectation à résoudre peut être diminué à un seul problème si le coût d'affectation d'une tâche dans un groupe G_i donné, vérifie l'expression : $V_j(i,1) \leq V_j(i,2) \leq \ldots \leq V_j(i,n), \quad \forall\ j\ et\ i$, en particulier cette propriété est vraie pour le critère C_{max}.

Ensuite, dans la section 4.5, nous avons généralisé l'idée de Rustogi et Strusevich [98, 99] (cf. section 3.3.2) dont la maintenance ne permet pas de restaurer totalement les conditions initiales de la machine vu les différentes qualités d'intervention des équipes de maintenances. De nouveau, nous avons montré que le problème reste polynomial.

Nous présentons, dans le tableau 4.9, un résumé de nos résultats. Notons qu'une comparaison, entre les résultats obtenus dans ce chapitre et les résultats antérieurs, a été fournie dans les tableaux : 4.8, 4.7 et 4.4.

Tableau 4.9 – Tableau récapitulatif des résultats

Problèmes	**Approche globale**	
	Complexité	Références
$1\|p(j,r), \mathcal{MP} = k\|\gamma, \gamma_{cost}$	$\mathcal{O}(n^{k+3})$	Théorème 4.3
$1\|p(j,r), \mathcal{MP} \leq k_0\|\gamma, \gamma_{cost}$	$\mathcal{O}(n^{k_0+3})$	Remarque 4.2
$1\|p(j,r) - det, \mathcal{MP} = k\|C_{max}$	$\mathcal{O}(n^3)$	Théorème 4.4
$1\|p(j,r) - det, \mathcal{MP}\|C_{max}$	$\mathcal{O}(n^4)$	Théorème 4.5
$1\|p^{(i)}(j,r), \mathcal{MP}^{(i)} = k\|\gamma, \gamma_{cost}$	$\mathcal{O}(n^{k+3})$	Théorème 4.7
$1\|p^{(i)}(j,r) - det, \mathcal{MP}^{(i)}\|C_{max}$	$\mathcal{O}(n^4)$	Théorème 4.8

Dans le chapitre suivant, nous allons présenter une généralisation de cette approche sur les problèmes d'ordonnancement avec des machines parallèles.

5

Problèmes d'ordonnancement à machines parallèles avec des périodes d'indisponibilité

Résumé : Les problèmes d'ordonnancement dans un atelier à une machine ont été largement étudiés. Néanmoins, l'atelier à machines parallèles est plus intéressant et proche des problèmes réels rencontrés dans la pratique. Dans ce chapitre, nous étendons l'approche présentée dans le chapitre précédent aux machines parallèles. Ces résultats permettent de résoudre des problèmes non traités avant et de généraliser et améliorer plusieurs travaux antérieurs.

Sommaire

Ces travaux ont été réalisés en collaboration avec Gerd Finke. Les résultats développés dans ce chapitre ont été présentés dans les articles suivants :

[37] A. Gara-Ali, G. Finke, et M.-L. Espinouse. *Parallel-machine scheduling with maintenance : Praising the assignment problem.* European Journal of Operational Research **252**, 90 – 97 (2016).

[34] A. Gara-Ali, M.-L. Espinouse, et G. Finke. *Unrelated parallel-machine scheduling with deterioration effects and multi-maintenance activities.* In *CIE-44 - 44th International conference on Computers & Industrial Engineering*, Istanbul, Turkey (2014).

[36] A. Gara-Ali, M.-L. Espinouse, et G. Finke. *Problèmes d'ordonnancement sur des machines parallèles avec des périodes de maintenance et un effet de détérioration : Minimisation de la charge totale des machines.* In *16ème conférence ROADEF Société Française de Recherche Opérationnelle et Aide à la Décision*, Marseille, France (2015).

5.1 Introduction

Dans le chapitre précédent, nous avons proposé une approche globale pour résoudre les problèmes d'ordonnancement à une machine en présence de périodes de maintenance variables et de tâches dépendantes de la position $p(j,r)$. Plusieurs critères d'optimisation ont été étudiés, à savoir : la minimisation de la durée totale de l'ordonnancement, la somme des dates de fin, la somme des durées d'attente, la somme de la variation des dates de fin [1] et la somme de la variation des durées d'attente [2]. Dans ce chapitre, nous continuons à explorer les problèmes d'ordonnancement simultané de tâches et de maintenances sur des machines parallèles.

Tout d'abord, nous commençons par une présentation du problème étudié. Ensuite, nous étudions deux versions du problème : le modèle global présenté dans la section 3.2 et une extension dont les paramètres sont dépendants du groupe (cf. section 3.3.2). Pour les deux cas, nous présentons une méthode de résolution basée sur un problème d'affectation. En particulier, nous montrons que la minimisation de la charge totale des machines ($\sum_{l=1}^{m} C_{max}^{l}$) peut être résolue avec une complexité améliorée par rapport aux autres critères. Les résultats présentés dans ce chapitre permettent de résoudre des problèmes non traités avant et généraliser plusieurs résultats antérieurs.

5.2 Présentation du problème

Dans ce chapitre, nous considérons n tâches T_j $(j = 1, \ldots, n)$ à ordonnancer sur m machines parallèles indépendantes $\{M_1, M_2, \ldots, M_m\}$. Les machines sont disponibles à l'instant zéro et ne peuvent réaliser qu'une seule opération à la fois. Les tâches sont toujours disponibles et la préemption n'est pas autorisée. Les durées des tâches suivent le modèle général présenté dans la section 3.3.1, *i.e.* la durée de la tâche T_j varie selon sa position et est définie par $p(l,j,r)$.

Des activités de maintenance peuvent être insérées sur les machines pour rétablir les conditions initiales de machine, *i.e.* les rangs r sont réinitialisés après chaque période de maintenance. Dans ce chapitre, on considère le système de maintenances pondérées $\mathcal{MP}$ présenté dans la section 3.2.2. La durée de la $i^{\text{ème}}$ période de maintenance sur la machine M_l est donnée par :

$$m_{li} = \sum_{j \in G_i^l} w(l,j,[j]) p(l,j,[j]) + \beta_{li}$$

Dans un ordonnancement π donné, la position d'une tâche T_j est définie par le triplet

1. Appelé aussi *Total absolute deviation of job completion times* ($TADC$)
2. Appelé aussi *Total absolute deviation of job waiting times* ($TADW$)

(l : machine, i : groupe, r : rang dans le groupe). Ainsi, la permutation est donnée par : $j \rightarrow (l,i,r) = [j]$ et $(l,i,r) \rightarrow j = [l,i,r]$. On note $[l,i,r] = j$ la tâche ordonnancée à la position r du groupe G_i sur la machine M_l.

Dans ce chapitre, nous étudions des critères d'optimisation dits décomposables par machine. γ est décomposable s'il est de forme :

$$\gamma = \sum_{l=1}^{m} \gamma^l \tag{5.1}$$

avec γ^l un critère d'optimisation liée à la machine M_l.

Par définition, ces critères d'optimisation sont décomposables :

- la charge totale des machines : $TML = \sum_{l=1}^{m} C_{max}^l$
- la somme des dates de fin : $TC = \sum C_j = \sum_{l=1}^{m} TC^l$
- la somme des durées d'attente : $TW = \sum W_j = \sum_{l=1}^{m} TW^l$
- la somme de la variation absolue des dates de fin : $TADC = \sum_{l=1}^{m} TADC^l$
- la somme de la variation absolue des durées d'attente : $TADW = \sum_{l=1}^{m} TADW^l$

5.3 Approche globale

Nous étendons, dans cette section, l'approche étudiée précédemment aux machines parallèles. L'objectif de cette approche est de résoudre le problème $Rm|p(l,j,r),\mathcal{MP}|\gamma$.

Soit n_l le nombre de tâches ordonnancées sur la machine M_l, tel que $\sum_{l=1}^{m} n_l = n$. Chaque machine M_l ($l = 1,\ldots,m$) peut être maintenue plusieurs fois. Avec k_l maintenances sur la machine M_l, nous obtenons $k_l + 1$ groupes de tâches formés sur la machine, on note par G_i^l ($l = 1,\ldots,m$ et $i = 1,\ldots,k_l+1$) le $i^{\text{ème}}$ groupe de tâches sur la machine M_l et u_i^l sa taille, avec $\sum_{i=1}^{k_l+1} u_i^l = n_l$ (cf. figure 5.1). Soit k le nombre total de maintenances à insérer sur l'ensemble des machines, avec $k = \sum_{l=1}^{m} k_l$. Ainsi, le nombre total de groupes formés sur toutes les machines est $\sum_{l=1}^{m}(k_l+1) = \sum_{l=1}^{m} k_l + m = k + m$.

Lorsque le nombre de maintenances total k est une donnée, nous définissons le problème d'ordonnancement par $Rm|p(l,j,r),\mathcal{MP} = k|\gamma$.

Les critères d'optimisation décomposables par machine peuvent être écrits sous la forme :

$$\gamma = \sum_{l=1}^{m} \sum_{i=1}^{k_l+1} \sum_{r=1}^{u_i^l} V_{[l,i,r]}(l,i,r) + \Phi(n,k) \tag{5.2}$$

Avec $\Phi(n,k)$ une constante et $V_j(l,i,r)$ le coût d'affectation d'une tâche T_j en position (l,i,r), *i.e.* au rang r du $i^{\text{ème}}$ groupe de la machine M_l (groupe G_i^l).

En utilisant les coûts d'affectation dans un atelier à une machine (4.5)-(4.9) et en se basant sur le fait que les critères d'optimisation sont décomposables par machine (5.1),

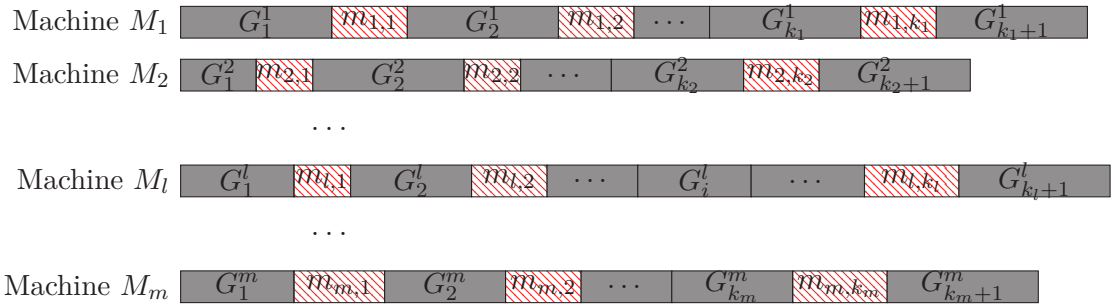

Figure 5.1 – Répartition des tâches sur les machines

nous rapportons, dans ce qui suit, les différents coûts d'affectation dans un atelier à machines parallèles :

La charge totale des machines

$$TML: \quad V_j(l,i,r) = (1+\lambda_i^l w(l,j,r))p(l,j,r), \text{ avec } \lambda_i^l = \begin{cases} 1 & \text{si } i \leq k_l \\ 0 & \text{si } i = k_l+1 \end{cases} \tag{5.3}$$

La somme des dates de fin

$$\begin{aligned} TC: \quad & V_j(l,i,r) = \Big(n_l - s_{ir}^l + 1 + \lambda_i^l w(l,j,r)\Big)\, p(l,j,r), \\ & \text{avec } \lambda_i^l = \textstyle\sum_{h=i+1}^{k_l+1} u_h^l \text{ et } \lambda_{k_l+1}^l = 0, \ \forall\, i,l. \end{aligned} \tag{5.4}$$

La somme des durées d'attente

$$\begin{aligned} TW: \quad & V_j(l,i,r) = \Big(n_l - s_{ir}^l + \lambda_i^l w(l,j,r)\Big)\, p(l,j,r), \\ & \text{avec } \lambda_i^l = \textstyle\sum_{h=i+1}^{k_l+1} u_h^l \text{ et } \lambda_{k_l+1}^l = 0, \ \forall\, i,l. \end{aligned} \tag{5.5}$$

La variation des dates de fin

$$\begin{aligned} TADC: \quad & V_j(l,i,r) = \Big[(s_{ir}^l - 1)(n_l - s_{ir}^l + 1) + \lambda_i^l w(l,j,r)\Big]\, p(l,j,r), \\ & \text{avec } \lambda_i^l = (\textstyle\sum_{h=1}^{i} u_h^l)(\sum_{h=i+1}^{k_l+1} u_h^l), \text{ avec } \lambda_{k_l+1}^l = 0, \ \forall\, i,l. \end{aligned} \tag{5.6}$$

La variation des durées d'attente

$$\begin{aligned} TADW: \quad & V_j(l,i,r) = \Big[(s_{ir}^l)(n_l - s_{ir}^l) + \lambda_i^l w(l,j,r)\Big]\, p(l,j,r), \\ & \text{avec } \lambda_i^l = (\textstyle\sum_{h=1}^{i} u_h^l)(\sum_{h=i+1}^{k_l+1} u_h^l), \text{ avec } \lambda_{k_l+1}^l = 0, \ \forall\, i,l. \end{aligned} \tag{5.7}$$

où $s_{ir}^l = r + \sum_{h=0}^{i-1} u_h^l$, avec $u_0^l = 0$, $\forall\, i \, et \, l$, s_{ir}^l représente le rang global de la tâche sur la machine M_l (r définit le rang dans le groupe) et $\Phi(n,k) = \sum_{l=1}^{m} \sum_{i=1}^{k_l} \lambda_i^l \beta_{li}$.

Notons que pour tous ces critères d'optimisation, le coût d'affectation $V_j(l,i,r)$ intègre deux sous-coûts :

- le coût classique d'affectation d'une tâche en position (l,i,r) dans un problème sans maintenance.
- $\lambda_i^l w(l,j,r)p(l,j,r)$ le coût de la maintenance associée à la tâche T_j dans la position (l,i,r)

Les coûts élémentaires des maintenances β_{li} sont intégrés dans le terme $\Phi(n,k)$.

Dans un premier temps, nous considérons que le nombre de maintenances $(k_1,k_2,\ldots,k_m)$ sur chaque machine M_l et les tailles de chaque groupe sont connus d'avance. Notre problème d'ordonnancement est donc le suivant : $R|p(l,j,r),\mathcal{MP}(n_l,k_l,u^l)=k|\gamma$, avec $\gamma=\{\sum_{l=1}^m C_{max}^l, \sum C_j, TW, TADC, TADW\}$. Nous notons que $\mathcal{MP}(n_l,k_l,u^l)=k$ indique que le nombre de tâches n_l, le nombre de maintenances k_l sur M_l $(l=1,\ldots,m)$ et le nombre de tâches u_i^l dans chaque groupe G_i^l sont connus d'avance et que le nombre total de maintenances $\sum_{l=1}^m k_l$ est égal à k. Notons aussi que le deuxième terme de la notation « $=k$ » est facultatif, car le vecteur $(k_1,k_2,\ldots,k_m)$ est une donnée.

Lorsque le vecteur $(k_1,k_2,\ldots,k_m)$ et la taille u_i^l de chaque groupe G_i^l sont connus d'avance, le terme $\Phi(n,k)$ est une constante, ainsi la minimisation de la fonction objectif (5.2) est équivalente à la minimisation de :

$$\sum_{l=1}^{m}\sum_{i=1}^{k_l+1}\sum_{r=1}^{u_i^l} V_{[l,i,r]}(l,i,r) \tag{5.8}$$

Le problème peut être modélisé sous forme d'un problème d'affectation $\mathcal{PA}$ de taille $n\times n$. Les lignes correspondent aux n tâches à affecter, chaque colonne représente une position (l,i,r) et $V_j(l,i,r)$ les entrées de la matrice d'affectation, *i.e.* le coût d'affectation d'une tâche T_j à la position (l,i,r).

Soit les variables binaires :

$$x_{j,l,i,r}=\begin{cases}1 & \text{si la tâche } T_j \text{ est ordonnancée dans le rang } r \text{ du groupe } G_i^l \text{ sur } M_l\\ 0 & \text{sinon}\end{cases}$$

Nous pouvons alors exprimer le problème d'affectation de la façon suivante :

Programme linéaire en nombres entiers 6 (PLNE.6).

$$min \quad \sum_{j=1}^{n}\sum_{l=1}^{m}\sum_{i=1}^{k_l+1}\sum_{r=1}^{u_i^l} V_j(l,i,r)x_{j,l,i,r}$$

$$s.c. \quad \sum_{j=1}^{n} x_{j,l,i,r} = 1 \quad l=1,\ldots,m \quad i=1,\ldots,k_l+1 \quad r=1,\ldots,u_i^l \tag{5.9}$$

$$\sum_{l=1}^{m}\sum_{i=1}^{k_l+1}\sum_{r=1}^{u_i^l} x_{j,l,i,r} = 1 \quad j=1,\ldots,n \tag{5.10}$$

$$x_{j,l,i,r} \in \{0,1\} \quad j=1,\ldots,n,\ l=1,\ldots,m,\ i=1,\ldots,k_l+1,\ r=1,\ldots,u_i^l$$

Les contraintes (5.9) et (5.10) représentent les contraintes classiques d'un problème d'affectation, *i.e.* chaque position est associée à une seule et unique tâche et chaque tâche est affectée à une seule position.

La résolution de ce problème d'affectation permet donc de trouver la séquence de tâches et les positions de maintenances optimales, soit V^γ la valeur obtenue par le programme d'affectation $\mathcal{PA}$ (P.L.6). Alors la solution optimale de l'ordonnancement γ est donnée par $V^\gamma + \Phi(n,k)$.

Théorème 5.1. *Les problèmes d'ordonnancement à machines parallèles :*
$R|p(l,j,r),\mathcal{MP}(n_l,k_l,u^l)|\gamma$ *avec* $\gamma = \{\sum_{l=1}^{m} C_{max}^l, \sum C_j, TW, TADC, TADW\}$
peuvent être résolus en temps polynomial $\mathcal{O}(n^3)$ *(cf. section 3.4.1), en utilisant le programme linéaire (PLNE.6).*

Pour bien comprendre le principe, nous allons résoudre l'exemple suivant.

Exemple 5.1. *Nous considérons* $n = 10$ *tâches à ordonnancer sur* $m = 2$ *machines avec* $k = 3$ *périodes de maintenance. On suppose que* $k_1 = 1$ *et* $k_2 = 2$. *Les tailles des groupes sur les machines sont :*

- *sur la première machine* $n_1 = 3$ *:* $G_1^1 \rightarrow u_1^1 = 2$ *;* $G_2^1 \rightarrow u_2^1 = 1$
- *sur la deuxième machine* $n_2 = 7$ *:* $G_1^2 \rightarrow u_1^2 = 3$ *;* $G_2^2 \rightarrow u_2^2 = 2$ *;* $G_3^2 \rightarrow u_3^2 = 2$

L'objectif est la minimisation de la somme des dates de fin ($\sum C_j$). *Les temps opératoires sont présentés sur le tableau 5.1.*

Les facteurs de détérioration pour les maintenances sont $w(1,j,r) = 0,2$ *pour tous* j *et* r *;* $w(2,j,r) = 0,1$ *pour tous* r *et* $j = 1,2,3,4,5,6$ *;* $w(2,j,r) = 0.3$ *pour tous* r *et* $j = 7,8,9,10$. *Les durées constantes* $\beta_{1,1} = 1$, $\beta_{2,1} = 2$ *et* $\beta_{2,2} = 3$.

Tableau 5.1 – Les temps opératoires $\forall\ j = 1, \ldots, n$

	Positions (r)									
	1	2	3	4	5	6	7	8	9	10
$p(1, j, r)$	1	2	3	3	3	3	3	3	3	3
$p(2, j, r)$	2	3	4	4	4	4	4	4	4	4

En utilisant les entrées $V_j(l, i, r)$ de la matrice d'affectation, voir l'expression 5.4, on construit la matrice M suivante :

$$M = \overbrace{\begin{pmatrix} \overbrace{3,2 \quad 4,4}^{G_1^1} & \overbrace{1}^{G_2^1} & \overbrace{14,8 \quad 19,2 \quad 21,6}^{G_1^2} & \overbrace{8,4 \quad 9,6}^{G_2^2} & \overbrace{4 \quad 3}^{G_3^2} \\ 3,2 \quad 4,4 & 1 & 14,8 \quad 19,2 \quad 21,6 & 8,4 \quad \boxed{9,6} & 4 \quad 3 \\ 3,2 \quad 4,4 & \boxed{1} & 14,8 \quad 19,2 \quad 21,6 & 8,4 \quad 9,6 & 4 \quad 3 \\ 3,2 \quad 4,4 & 1 & \boxed{14,8} \quad 19,2 \quad 21,6 & 8,4 \quad 9,6 & 4 \quad 3 \\ 3,2 \quad 4,4 & 1 & 14,8 \quad \boxed{19,2} \quad 21,6 & 8,4 \quad 9,6 & 4 \quad 3 \\ 3,2 \quad 4,4 & 1 & 14,8 \quad 19,2 \quad \boxed{21,6} & 8,4 \quad 9,6 & 4 \quad 3 \\ 3,2 \quad 4,4 & 1 & 16,4 \quad 21,6 \quad 24,8 & 9,2 \quad 10,8 & \boxed{4} \quad 3 \\ 3,2 \quad 4,4 & 1 & 16,4 \quad 21,6 \quad 24,8 & 9,2 \quad 10,8 & 4 \quad \boxed{3} \\ \boxed{3,2} \quad 4,4 & 1 & 16,4 \quad 21,6 \quad 24,8 & 9,2 \quad 10,8 & 4 \quad 3 \\ 3,2 \quad \boxed{4,4} & 1 & 16,4 \quad 21,6 \quad 24,8 & 9,2 \quad 10,8 & 4 \quad 3 \end{pmatrix}}^{10\,\text{tâches}}$$

Machine M_1 Machine M_2

La résolution du problème d'affectation linéaire permet de trouver une solution optimale $V^{\sum C_j} = 89, 2$. Le terme constant $\Phi(n, k)$ est $\sum_{l=1}^{m} \sum_{i=1}^{k_l} \lambda_i^l \beta_{li} = 15$, avec $\lambda_1^1 = 1$, $\lambda_1^2 = 4$ et $\lambda_2^2 = 2$. Alors, la solution optimale de la somme des dates de fin :
$TC = \sum C_j = V^{\sum C_j} + \Phi(n, k) = 89, 2 + 15 = 104, 2$.

La séquence optimale est présentée dans le diagramme de Gantt suivant :

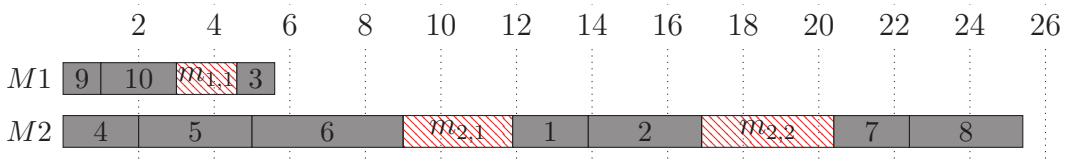

Figure 5.2 – Minimisation de $\sum C_j$

Les durées des maintenance sont : $m_{1,1} = 1, 6$, $m_{2,1} = 2, 9$ et $m_{2,2} = 3, 5$.

Pour trouver la solution globale du problème $R|p(l, j, r), \mathcal{MP} = k|\gamma$, *i.e.* lorsque seulement le nombre total de maintenances k est donné, nous devons résoudre le problème d'ordonnancement précédent, *i.e.* $R|p(l, j, r), \mathcal{M}(n_l, k_l, u^l) = k|\gamma$, pour toutes les combinaisons possibles. Tout d'abord, nous avons $O(n^{m-1})$ combinaisons pour partitionner les n tâches sur les m machines telle que $n_1 + n_2 + \ldots + n_m = n$, voir lemme 4.1. Ensuite, nous

attribuons un certain nombre de maintenances k_l pour chaque machine M_l $(l = 1, \ldots, m)$ tel que $k_1 + k_2 + \ldots + k_m = k$ avec k le nombre total de maintenances à insérer. Il est évident que le nombre de maintenances total k est inférieur ou égal au nombre de tâches n, $k \leq n$. D'après, Mott et *al.* [91], le nombre total de partitionnements en nombres entiers non négatifs k_l est donné par le coefficient binomial $C(k + m - 1, m - 1)$. Nous avons :

$$C(k+m-1, m-1) \leq \frac{(k+m-1)^{m-1}}{(m-1)!} \leq \frac{[2max\{k,m\}]^{m-1}}{(m-1)!}$$

Or $\frac{2^{m-1}}{(m-1)!} \leq 2$, donc nous obtenons :

$$C(k+m-1, m-1) \leq O([max\{k,m\}]^{m-1}) \tag{5.11}$$

D'autre part, nous avons $O(n_l^{k_l})$ combinaisons possibles pour répartir les tailles des groupes G_i^l sur une machine M_l $(l = 1, \ldots, m)$ telle que $u_1^l + u_2^l + \ldots + u_{k_l+1}^l = n_l$. Ainsi, le nombre de combinaisons $(u_1^l, u_2^l, \ldots, u_n^l)$ pour l'ensemble des machines est :

$$\prod_{l=1}^{m} O(n_l^{k_l}) \leq O(n^{\sum_{l=1}^{m} k_l}) \leq O(n^k) \tag{5.12}$$

Nous devons donc résoudre $O([max\{k,m\}]^{m-1} n^{m+k-1})$ problèmes d'affectation dont la complexité de résolution est $\mathcal{O}(n^3)$ (cf. section 3.4.1). En se basant, sur l'analyse ci-dessus nous développons l'algorithme 3. L'étape 1 est exécutée $\mathcal{O}(n^{m-1})$ fois, l'étape 1.1 et l'étape 1.1.1 sont exécutées en $\mathcal{O}([max\{k,m\}]^{m-1})$ et $\mathcal{O}(n^k)$, respectivement.

Théorème 5.2. *Les problèmes d'ordonnancement à machines parallèles avec k périodes de maintenance à ordonnancer : $R|p(l,j,r), \mathcal{MP} = k|\gamma$*
avec $\gamma = \{\sum_{l=1}^{m} C_{max}^l, \sum C_j, \sum W_j, TADC, TADW\}$, peuvent être résolus avec une complexité $\mathcal{O}([max\{k,m\}]^{m-1} n^{m+k+2})$, en utilisant l'algorithme 3.

Si le nombre de maintenances k et le nombre de machines m sont constants, le problème $Rm|p(l,j,r), \mathcal{MP} = k|\gamma$ peut être résolu avec une complexité polynomiale $\mathcal{O}(n^{m+k+2})$.

Remarque 5.1. *On peut aussi étendre le théorème 5.2 à des fonctions plus générales. En effet, le résultat reste vrai pour toutes les combinaisons linéaires des critères étudiés précédemment de forme : $\gamma_{cost} = \mu_1 C_{max}^l + \mu_2 \sum C_j + \mu_3 \sum W_j + \mu_4 TADC + \mu_5 TADW$, avec $\mu_1 \geq 0$, $\mu_2 \geq 0$, $\mu_3 \geq 0$, $\mu_4 \geq$, $0\mu_5 \geq 0$ des paramètres constants.*

Remarque 5.2. *Pour des raisons budgétaires, une borne supérieure de nombre de maintenances peut être ajoutée. On note k_0 le nombre de maintenances maximal à insérer sur l'ensemble de machines. Pour résoudre le problème $Rm|p(l,j,r), \mathcal{MP} \leq k_0|\gamma, \gamma_{cost}$, il suffit d'exécuter l'algorithme 3 pour $k = 1, \ldots, k_0$ et de renvoyer la meilleure solution. Ainsi, la complexité finale de cette résolution est : $\mathcal{O}([max\{k_0,m\}]^{m-1} n^{m+k_0+2})$.*

Algorithme 3 : Problèmes à machines parallèles $R|p(l,j,r),\mathcal{MP}=k|\gamma$

Données :
- $p(l,j,r)$ temps opératoire de chaque tâche T_j au rang r sur la machine M_l
- $w(l,j,r)$ coefficient de la maintenance due à l'ordonnancement de la tâche T_j au rang r sur la machine M_l
- β_{li} durée élémentaire de la $i^{\text{ème}}$ maintenance sur la machine M_l
- $\mathscr{N}$ ensemble de vecteurs $(n_1, n_2, \ldots, n_m)$ possibles tels que $n_1 + n_2 + \ldots + n_m = n$
- $\mathscr{K}$ ensemble de vecteurs $(k_1, k_2, \ldots, k_m)$ possibles tels que $k_1 + k_2 + \ldots + k_m = k$
- $\mathscr{U}$ ensemble de vecteurs $(u_1^l, u_2^l, \ldots, u_{k_l+1}^l)$ possibles, avec $(l = 1, \ldots, m)$, tels que $u_1^l + u_2^l + \ldots + u_{k_l+1}^l = n_l$

Résultat : la solution optimale : γ^*

début

1 **pour chaque** $(n_1, n_2, \ldots, n_m) \in \mathscr{N}$ **faire**

1.1 **pour chaque** $(k_1, k_2, \ldots, k_m) \in \mathscr{K}$ **faire**

1.1.1 **pour chaque** $(u_1^l, u_2^l, \ldots, u_{k_l+1}^l) \in \mathscr{U}$ **faire**

/* Résoudre $\left(R|p(l,j,r),\mathcal{MP}(n_l,k_l,u^l)|\gamma\right)$ */

1.1.1.1 Calculer les facteurs λ_i^l et définir la matrice des coûts M de taille $n \times n$

1.1.1.2 $V^\gamma \longleftarrow$ *Résoudre* $\mathcal{PA}$ (PLNE.6)

1.1.1.3 $\gamma^*(n_l, k_l, u^l) \longleftarrow V^\gamma + \Phi(n,k)$, avec $\Phi(n,k) = \sum_{l=1}^{m} \sum_{i=1}^{k_l} \lambda_i^l \beta_{li}$

fin pour

fin pour

fin pour

2 **retourner** $\gamma^* \longleftarrow \min\{\gamma^*(n_l, k_l, u^l)\}$

fin

Dans [129], Yang a étudié un modèle plus restrictif avec un nombre de machines m fixe, $w(l,j,r) = \alpha_l$ et $\beta_{li} = \beta_l$, et seule la minimisation de la somme des dates de fin a été étudiée avec au plus k_0 périodes de maintenance à insérer sur les machines. Les auteurs ont résolu le problème avec une complexité $\mathcal{O}(n^{m+k_0+2})$, voir tableau 5.2.

5.4 Ordonnancement dépendant du groupe

Dans cette section, nous présentons une extension du problème étudié dans la section 5.3 avec la prise en compte d'une configuration dépendante du groupe (cf. section 3.3.2).

Nous définissons notre problème comme suit : n tâches $J = \{T_1, T_2, \ldots, T_n\}$ à ordonnancer sur m machines parallèles $M = \{M_1, M_2, \ldots, M_m\}$ et k périodes de maintenance à insérer avec k_l le nombre de maintenances sur la machine M_l tel que $\sum_{l=1}^{m} k_l = k$. Nous considérons que les temps opératoires sont dépendants du rang r de la tâche, de la ma-

Tableau 5.2 – Comparaison avec les résultats antérieurs

Notre modèle	Modèle et complexité de la littérature		
	Réf.	Modèle	Complexité
∘ m machines parallèles ∘ Critère d'optimisation : γ ∘ Temps opératoires : $p(l,j,r)$ ∘ Système de maintenance $\mathcal{MP}$: $m_{li} = \sum_{j \in G_i^l} w(l,j,[j])p(l,j,[j]) + \beta_{li}$ ∘ k périodes de maintenance à insérer ⇒ Complexité : $\mathcal{O}([max\{k,m\}]^{m-1}n^{m+k+2})$ (théorème 5.2) ⇒ Si m et k sont fixes, la complexité est : $\mathcal{O}(n^{m+k+2})$ ⇒ Si m est fixe et k_0 une borne supérieure, la complexité est : $\mathcal{O}(n^{m+k_0+2})$ (remarque 5.2).	[129]	∘ m machines parallèles fixes ∘ $\sum_{j=1}^{n} C_j$ ∘ $p(l,j,r) = p_{lj}g_{lj}(r)$ ∘ $w(l,j,r) = \alpha_l$ et $\beta_{li} = \beta_l$ ∘ k_0 nombre maximum de maintenances	$\mathcal{O}(n^{m+k_0+2})$

chine M_l et aussi du groupe i soit $p^{(i)}(l,j,r)$ le temps opératoire de la tâche j ordonnancée au rang r du groupe G_i^l : groupe i sur la machine M_l. Nous considérons le système de maintenance $\mathcal{MP}^{(i)}$, dont les coefficients des maintenances $w^{(i)}(l,j,r)$ sont dépendants du groupe. La durée de la $i^{\text{ème}}$ maintenance sur la machine M_l est donnée par :

$$m_{li} = \sum_{r=1}^{u_i^l} w^{(i)}(l,[l,i,r],r)p^{(i)}(l,[l,i,r],r) + \beta_{li} \tag{5.13}$$

où $p^{(i)}(l,j,r)$ est le temps opératoire de la tâche T_j ordonnancée au rang r du groupe G_i^l et β_{li} la durée élémentaire de la maintenance. La position dans l'ordonnancement est représentée par le triplet (l,i,r) *(machine, groupe, rang dans le groupe)*. Nous rappelons que u_i^l est le nombre de tâches ordonnancées entre la maintenance $i^{éme} - 1$ et la $i^{\text{ème}}$ maintenance sur la machine M_l, *i.e.* le nombre de tâches du groupe G_i^l (voir figure 5.1), et que $[l,i,r]$ désigne la tâche T_j ordonnancée à la position (l,i,r). Nous étudions donc le problème $R|p^{(i)}(l,j,r), \mathcal{MP}^{(i)} = k|\gamma$, avec $\gamma = \{\sum_{l=1}^{m} C_{max}^l, \sum C_j, TW, TADC, TADW\}$.

Lorsque le nombre de tâches sur les machines n_l $(l = 1,\ldots,m)$, le nombre de maintenances k_l et les tailles de groupes sur les machines sont connus d'avance, nous pouvons remarquer que, de nouveau, les critères $\gamma = \{\sum_{l=1}^{m} C_{max}^l, \sum C_j, \sum W_j, TADC, TADW\}$ sont sous la forme :

$$\gamma = \sum_{l=1}^{m}\sum_{i=1}^{k+1}\sum_{r=1}^{u_i^l} V_{[l,i,r]}(l,i,r) + \Phi(n,k) \tag{5.14}$$

où $V_j(l,i,r)$ est le coût d'affectation de la tâche T_j à la position (l,i,r), *i.e.* au rang r dans le groupe G_i^l et $\Phi(n,k)$ est une constante. Ci-dessous nous rapportons les coûts

d'affectation des différents critères d'optimisation étudiés :

La charge totale des machines

$$TML: \quad V_j(l,i,r) = (1+\lambda_i^l w^{(i)}(l,j,r))p^{(i)}(l,j,r), \text{ avec } \lambda_i^l = \begin{cases} 1 & \text{si } i \leq k_l \\ 0 & \text{si } i = k_l+1 \end{cases} \tag{5.15}$$

La somme des dates de fin

$$\begin{aligned} TC: \quad & V_j(l,i,r) = \left(n_l - s_{ir}^l + 1 + \lambda_i^l w^{(i)}(l,j,r)\right) p^{(i)}(l,j,r), \\ & \text{avec } \lambda_i^l = \textstyle\sum_{h=i+1}^{k_l+1} u_h^l \text{ et } \lambda_{k_l+1}^l = 0, \forall\, i,l. \end{aligned} \tag{5.16}$$

La somme des durées d'attente

$$\begin{aligned} TW: \quad & V_j(l,i,r) = \left(n_l - s_{ir}^l + \lambda_i^l w^{(i)}(l,j,r)\right) p^{(i)}(l,j,r), \\ & \text{avec } \lambda_i^l = \textstyle\sum_{h=i+1}^{k_l+1} u_h^l \text{ et } \lambda_{k_l+1}^l = 0, \forall\, i,l. \end{aligned} \tag{5.17}$$

La variation des dates de fin

$$\begin{aligned} TADC: \quad & V_j(l,i,r) = \left[(s_{ir}^l - 1)(n_l - s_{ir}^l + 1) + \lambda_i^l w^{(i)}(l,j,r)\right] p^{(i)}(l,j,r), \\ & \text{avec } \lambda_i^l = (\textstyle\sum_{h=1}^{i} u_h^l)(\sum_{h=i+1}^{k_l+1} u_h^l), \text{ avec } \lambda_{k_l+1}^l = 0, \forall\, i,l. \end{aligned} \tag{5.18}$$

La variation des durées d'attente

$$\begin{aligned} TADW: \quad & V_j(l,i,r) = \left[(s_{ir}^l)(n_l - s_{ir}^l) + \lambda_i^l w^{(i)}(l,j,r)\right] p^{(i)}(l,j,r), \\ & \text{avec } \lambda_i^l = (\textstyle\sum_{h=1}^{i} u_h^l)(\sum_{h=i+1}^{k_l+1} u_h^l), \text{ avec } \lambda_{k_l+1}^l = 0, \forall\, i,l. \end{aligned} \tag{5.19}$$

avec $s_{ir}^l = r + \sum_{h=0}^{i-1} u_h^l$, tel que $u_0^l = 0, \forall\, i \, et \, l$, s_{ir}^l représente le rang global de la tâche sur la machine M_l (r définit le rang dans le groupe) et $\Phi(n,k) = \sum_{l=1}^{m} \sum_{i=1}^{k_l} \lambda_i^l \beta_{li}$.

Lorsque les vecteurs $(n_1, n_2, \ldots, n_m)$, $(k_1, k_2, \ldots, k_m)$ et $(u_1^l, u_2^l, \ldots, u_{k_l+1}^l), \forall\, l = 1, \ldots, m$, sont connus d'avance, le terme $\Phi(n,k)$ de l'équation (5.14) est constant. Ainsi, la minimisation de (5.14) est équivalente à la minimisation de l'expression $\sum_{l=1}^{m} \sum_{i=1}^{k+1} \sum_{r=1}^{u_i^l} V_{[l,i,r]}(l,i,r)$.

De même que précédemment, le problème $R|p^{(i)}(l,j,r), \mathcal{MP}^{(i)}(n_i,k_l,u^l)|\gamma$ peut être aussi modélisé sous forme d'un problème d'affectation avec n tâches à affecter à n positions (l,i,r) et $V_j(l,i,r)$ est le coût d'affectation de la tâche T_j à une position (l,i,r), voir problème d'affectation (PLNE.6).

Théorème 5.3. *Les problèmes d'ordonnancement à machines parallèles* $R|p^{(i)}(l,j,r), \mathcal{MP}^{(i)}(n_l,k_l,u^l)|\gamma$ *avec* $\gamma = \{\sum_{l=1}^{m} C_{max}^l, \sum C_j, \sum W_j, TADC, TADW\}$ *peuvent être résolus en temps polynomial* $\mathcal{O}(n^3)$*, en utilisant le problème d'affectation (PLNE.6).*

Afin de résoudre le problème $R|p^{(i)}(l,j,r), \mathcal{MP}^{(i)} = k|\gamma$, et en utilisant la même analyse que celle de la section précédente, le nombre de problèmes d'affectation (PLNE.6) à résoudre est égal à $\mathcal{O}([max\{k,m\}]^{m-1}n^{m+k-1})$. Nous obtenons ainsi le résultat suivant :

Théorème 5.4. *Les problèmes d'ordonnancement à machines parallèles* $R|p^{(i)}(l,j,r), \mathcal{MP}^{(i)} = k|\gamma$ *avec* $\gamma = \{\sum_{l=1}^{m} C_{max}^l, \sum C_j, \sum W_j, TADC, TADW\}$ *peuvent être résolus avec une complexité* $\mathcal{O}([max\{k,m\}]^{m-1}n^{m+k+2})$.

Notons que si k et m sont constants, le problème $Rm|p^{(i)}(l,j,r), \mathcal{MP}^{(i)} = k|\gamma$ est polynomial avec une complexité $\mathcal{O}(n^{m+k+2})$.

Lorsqu'une borne supérieure k_0 est imposée pour le nombre total de maintenances, l'objectif du problème $R|p^{(i)}(l,j,r), \mathcal{MP}^{(i)} \leq k_0|\gamma$ consiste à trouver la séquence de tâches, le nombre de maintenances $k \leq k_0$ et ses positions, afin de minimiser le critère d'optimisation γ. Pour résoudre le problème, il suffit d'exécuter le problème $R|p^{(i)}(l,j,r), \mathcal{MP}^{(i)} = k|\gamma$ pour $k = 1, \ldots, k_0$ et en gardant la meilleure solution. D'où le résultat suivant :

Théorème 5.5. *Les problèmes d'ordonnancement à machines parallèles* $R|p^{(i)}(l,j,r), \mathcal{MP}^{(i)} \leq k_0|\gamma$*, où* k_0 *est la borne supérieure du nombre de maintenances, peuvent être résolus avec une complexité* $\mathcal{O}([max\{k_0,m\}]^{m-1}n^{m+k_0+2})$.
Si le nombre de machines m *est fixe, le problème est polynomial avec une complexité* $\mathcal{O}(n^{m+k_0+2})$.

Remarque 5.3. *Les théorèmes 5.3, 5.4 et 5.5 sont applicables aussi pour le critère* γ_{cost}*. Ce critère est généré à partir d'une combinaison linéaire* γ *telle que :* $\gamma_{cost} = \mu_1 \sum_{l=1}^{m} C_{max}^l + \mu_2 \sum C_j + \mu_3 \sum TW + \mu_4 TADC + \mu_5 TADW$*, avec* $\mu_1 \geq 0$*,* $\mu_2 \geq 0$*,* $\mu_3 \geq 0$*,* $\mu_4 \geq 0$*,* $\mu_5 \geq 0$ *des paramètres constants.*

Le théorème 5.4 généralise le résultat de Rustogi et Strusevich [98], dans lequel, les auteurs ont étudié un modèle plus restrictif avec $p^{(i)}(l,j,r) = p_{lj}g(l,i,r)$, $w^{(i)}(l,j,r) = \alpha_i$ et $\beta_{li} = \beta_i$ et avec comme objectif la minimisation de la somme des dates de fin. Le problème a été résolu avec la même complexité $\mathcal{O}(n^{m+k+2})$.

Ji et Cheng [59] ont étudié le modèle restrictif suivant : m machines parallèles et identiques Pm. Le temps opératoire de la tâche T_j ordonnancée au rang r du groupe G_i^l est donné par $\gamma_{lji}p_j r^{a_j}$, avec $0 < \gamma_{lji} \leq 1$ et $a_j \leq 0$. Les durées des maintenances sont constantes ($w^{(i)}(l,j,r) = 0$). Pour les deux critères d'optimisation $\sum_{l=1}^{m} C_{max}^l$ et $\sum C_j$ les

auteurs ont montré que le problème est polynomial avec une complexité $\mathcal{O}(n^{m+k+2})$. Le tableau 5.3 représente une comparaison entre les résultats obtenus dans cette section et les travaux antérieurs.

Tableau 5.3 – Comparaison avec les résultats antérieurs

Notre modèle	Modèle et complexité de la littérature		
	Réf.	Modèle	Complexité
∘ m machines parallèles ∘ Critère d'optimisation : γ ∘ Temps opératoires : $p^{(i)}(l,j,r)$ ∘ Système de maintenance $\mathcal{MP}^{(i)}$: $m_{li} = \sum_{j \in G_i^l} w^{(i)}(l,j,[j])p^{(i)}(l,j,[j]) + \beta_{li}$ ∘ k périodes de maintenance à insérer ⇒ Complexité : $\mathcal{O}([max\{k,m\}]^{m-1}n^{m+k+2})$ voir théorème 5.4 ⇒ Si m et k sont fixes, la complexité est : $\mathcal{O}(n^{m+k+2})$ ⇒ Si m est fixe et k_0 une borne supérieure, la complexité est : $\mathcal{O}(n^{m+k_0+2})$, (théorème 5.5).	[98]	∘ Atelier Rm ∘ $\sum_{j=1}^{n} C_j$ ∘ $p^{(i)}(l,j,r) = p_{lj}g(l,i,r)$ ∘ $w^{(i)}(l,j,r) = \alpha_i$ et $\beta_{li} = \beta_i$ ∘ k nombre de maintenances	$\mathcal{O}(n^{m+k+2})$
	[59]	∘ Atelier Pm ∘ $\sum_{l=1}^{m} C_{max}^l$ et $\sum C_j$ ∘ $p(l,j,r) = \gamma_{lji}p_j r^{a_j}$, avec $0 < \gamma_{lji} \leq 1$ et $a_j \leq 0$ ∘ $w^{(i)}(l,j,r) = 0$ ∘ k nombre de maintenances	$\mathcal{O}(n^{m+k+2})$

5.5 Minimisation de la charge totale des machines

Dans cette section, nous nous concentrons sur le critère d'optimisation $\sum_{l=1}^{m} C_{max}^l$ (la charge totale des machines). Nous considérons la version la plus globale du problème dont les paramètres sont dépendants du groupe (cf. section 3.3.2). Rappelons que la charge totale des machines est donnée par :

$$\sum_{l=1}^{m} C_{max}^l = \sum_{l=1}^{m} \sum_{i=1}^{k_l+1} \sum_{r=1}^{u_i^l} V_{[l,i,r]}(l,i,r) + \Phi(n,k) \tag{5.20}$$

où $V_j(l,i,r) = (1 + \lambda_i^l w^{(i)}(l,j,r))p^{(i)}(l,j,r)$, avec $\lambda_i^l = \begin{cases} 1 & \text{si } i \leq k_l \\ 0 & \text{si } i = k_l + 1 \end{cases}$

et $\Phi(n,k) = \sum_{l=1}^{m} \sum_{i=1}^{k_l} \beta_{li}$.

Nous allons montrer que la complexité de la résolution peut être améliorée considérablement par rapport aux autres critères d'optimisation, si nous considérons un effet de détérioration appliqué sur les machines et que les coefficients $w^{(i)}(l,j,r)$ sont croissants se-

lon r. Notre problème est le suivant $R|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)} = k| \sum_{l=1}^{m} C^l_{max}$, *i.e.* seulement le nombre de maintenances total est donné.

Nous avons donc, pour tout l, j et i :

$$p^{(i)}(l,j,1) \leq p^{(i)}(l,j,2) \leq \ldots \leq p^{(i)}(l,j,n) \tag{5.21}$$

$$w^{(i)}(l,j,1) \leq w^{(i)}(l,j,2) \leq \ldots \leq w^{(i)}(l,j,n) \tag{5.22}$$

Nous ajoutons les conditions suivantes pour tout l, j et r :

$$p^{(1)}(l,j,r) \leq p^{(2)}(l,j,r) \leq \ldots \leq p^{(n)}(l,j,r) \tag{5.23}$$

$$w^{(1)}(l,j,r) \leq w^{(2)}(l,j,r) \leq \ldots \leq w^{(n)}(l,j,r) \tag{5.24}$$

Afin de résoudre le problème, nous étudions en premier lieu le sous-problème d'or-

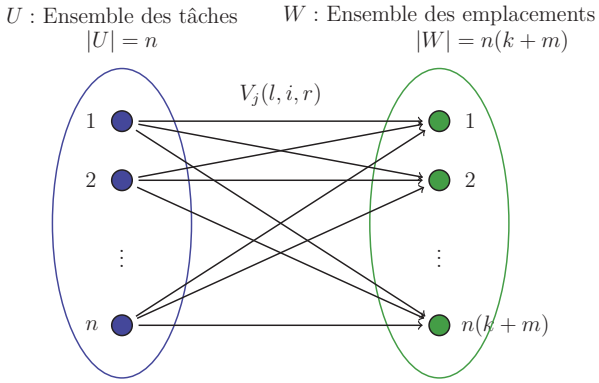

Figure 5.3 – Graphe biparti du problème étudié

donnancement $R|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)}(k_l)| \sum_{l=1}^{m} C^l_{max}$, dont le nombre de maintenances sur chaque machine k_l est connu d'avance, tel que $\sum_{l=1}^{m} k_l = k$. Nous ne considérons plus le nombre de tâches n_l et le vecteur $(u^l_1, u^l_2, \ldots, u^l_{k_l+1})$, $l = 1, \ldots, m$. Ensuite, nous étendons le nombre d'emplacements à n positions dans chaque groupe G^l_i au lieu de u^l_i ($\forall\ l = 1, \ldots, m\ et\ i = 1, \ldots, k_l + 1$). Nous ne savons pas à l'avance combien de tâches seront affectées à chaque groupe G^l_i. Ainsi, notre problème consiste à affecter n tâches à $n \sum_{l=1}^{m}(k_l + 1) = n(k + m)$ positions, voir figure 5.3. Il est clair qu'on a $nk + nm - n$ positions non occupées.

Le problème peut donc être modélisé sous forme d'un problème d'affectation rectangulaire $\mathcal{PAR}$ (cf. section 3.4.2).

Soient les variables binaires $x_{j,l,i,r} = \begin{cases} 1 & \text{si la tâche } T_j \text{ est dans la position } (l,i,r) \\ 0 & \text{sinon} \end{cases}$

et $V_j(l,i,r)$ le coût d'affectation d'une tâche T_j à une position (l,i,r).

Le programme linéaire en nombres entiers peut alors s'écrire comme suit :

Programme linéaire en nombres entiers 7 (PLNE.7).

$$min \quad \sum_{j=1}^{n}\sum_{l=1}^{m}\sum_{i=1}^{k_l+1}\sum_{r=1}^{n} V_j(l,i,r)x_{j,l,i,r}$$

$$s.c. \quad \sum_{l=1}^{m}\sum_{i=1}^{k_l+1}\sum_{r=1}^{n} x_{j,l,i,r} = 1 \quad j = 1,\ldots,n \tag{5.25}$$

$$\sum_{j=1}^{n} x_{j,l,i,r} \leq 1 \quad l = 1,\ldots,m, \quad i = 1,\ldots,k_l+1, \quad r = 1,\ldots,n \tag{5.26}$$

$$x_{j,l,i,r} \in \{0,1\} \quad j = 1,\ldots,n, \quad l = 1,\ldots,m, \quad i = 1,\ldots,k_l+1, \quad r = 1,\ldots,n$$

avec $V_j(l,i,r) = (1+\lambda_i^l w^{(i)}(l,j,r))p^{(i)}(l,j,r)$ et $\lambda_i^l = \begin{cases} 1 & \text{si } i \leq k_l \\ 0 & \text{si } i = k_l+1 \end{cases}$ (5.27)

Les contraintes (5.25) assurent que chaque tâche T_j est ordonnancée une seule fois. Les contraintes (5.26) garantissent que chaque position (l,i,r) est occupée par au plus une seule tâche.

Vu l'effet de détérioration imposé (5.21) et les facteurs de la détérioration de maintenances (5.22) et d'après (5.27), nous avons :

$$V_j(l,i,1) \leq V_j(l,i,2) \leq \ldots \leq V_j(l,i,n), \quad \forall\, l = 1,\ldots,m,\ i = 1,\ldots,k_{l+1} \tag{5.28}$$

Il est donc clair que les tâches seront affectées dans les groupes dans des positions adjacentes à partir du rang $r = 1$.

L'utilisation de ce problème d'affectation rectangulaire peut donner une solution avec des groupes vides à cause de la présence des coûts d'affectation élevés dans certains groupes. Avec des groupes vides, la solution est faisable, mais sous-optimale.

De plus, en utilisant les conditions 5.23 et 5.24, les groupes vides (s'ils existent) suivent les groupes non vides, sauf le dernier groupe $G^l_{k_l+1}$. Nous pouvons construire à partir de cette solution sous-optimale (solution avec des groupes vides) une solution

réalisable, en supprimant les blocs vides et en décalant le dernier bloc (groupe $G^l_{k_l+1}$), on obtient la solution optimale[3]. Soit $V^{\sum C_{max}}$ la valeur optimale obtenue par la résolution du problème d'affectation rectangulaire $\mathcal{PAR}$. La solution optimale du problème $R|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)}(k_l)|\sum_{l=1}^{m} C^l_{max}$ est donnée par :

$$\sum_{l=1}^{m} C^l_{max} = V^{\sum C_{max}} + \Phi(n,k)$$

avec $\Phi(n,k) = \sum_{l=1}^{m}\sum_{i=1}^{k'_l}\beta_{li}$, la partie constante de la fonction objectif 5.20. Notons que k'_l est le nouveau nombre de maintenances sur la machine M_l après la suppression des groupes vides ($k'_l \leq k_l$).

Selon Dinic [25] et Milo et *al.* [82] (cf. section 3.4.2), le problème d'affectation rectangulaire peut être résolu en temps polynomial avec une complexité $\mathcal{O}(n^3+n^2(k+m))$ pour un vecteur $(k_1, k_2, \ldots, k_m)$ donné.

Théorème 5.6. *Lorsque le nombre de maintenances k_l sur chaque machine M_l est connu d'avance, le problème d'ordonnancement à machines parallèles*

$$R|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)}(k_l)|\sum_{l=1}^{m} C^l_{max}$$

peut être résolu en temps polynomial $O(n^3 + n^2(k+m))$, où $k = \sum_{l=1}^{m} k_l$.

Comme cela a été précisé dans la section précédente, le nombre de vecteurs $(k_1, k_2, \ldots, k_m)$ tel que $k_1+k_2+\ldots+k_m = k$ est $O([max\{k,m\}]^{m-1})$, voir équation (5.11), il faut donc résoudre le problème d'affectation rectangulaire $O([max\{k,m\}]^{m-1})$ fois. Nous développons l'algorithme 4 pour résoudre le problème $R|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)} = k|\sum_{l=1}^{m} C^l_{max}$.

Théorème 5.7. *Pour un nombre de maintenances k donné, le problème d'ordonnancement à machines parallèles*

$$R|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)} = k|\sum_{l=1}^{m} C^l_{max}$$

peut être résolu en temps polynomial $\mathcal{O}\left([max\{k,m\}]^{m-1}(n^3 + n^2(k+m))\right)$.

Maintenant, nous supposons que $k, m \leq n$ et k varie entre 1 et n. Nous avons donc pour chaque k comme borne supérieure de complexité de résolution $\mathcal{O}(n^{m+2})$. D'où le résultat suivant :

3. Cette modification de la solution ne peut qu'améliorer la solution obtenue par le $\mathcal{PAR}$: supprimer les groupes vides pour obtenir $k'_l \leq k_l$ périodes de maintenance (amélioration) et décaler le dernier groupe au groupe $k'_l + 1$ (amélioration supplémentaire, car $w^{(i)}(l,j,r)$ et $p^{(i)}(l,j,r)$ sont croissants avec i).

Algorithme 4 : Problèmes d'ordonnancement $R|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)} = k| \sum_{l=1}^{m} C_{max}^{l}$

Données :
- $p^{(i)}(l,j,r)$ temps opératoire de chaque tâche T_j en position (l,i,r)
- $w^{(i)}(l,j,r)$ coefficient de détérioration de la maintenance
- β_{li} durée élémentaire de la maintenance
- $\mathscr{K}$ ensemble de vecteurs $(k_1, k_2, \ldots, k_m)$ possibles tels que $k_1 + k_2 + \ldots + k_m = k$

Résultat : TML^* la solution optimale

début

1 **pour chaque** $(k_1, k_2, \ldots, k_m) \in \mathscr{K}$ **faire**

/* Résoudre $\left(R|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)}(k_l)| \sum_{l=1}^{m} C_{max}^{l}\right)$ */

1.1 Calculer les facteurs λ_i^l et définir la matrice des coûts M de taille $n \times n(k+m)$.

1.2 $V^{\sum C_{max}} \longleftarrow$ *Résoudre* $\mathcal{PAR}$ (PLNE.7)

1.3 $\sum C_{max}^*(k_l) \longleftarrow V^{\sum C_{max}} + \Phi(n,k)$, avec $\Phi(n,k) = \sum_{l=1}^{m} \sum_{i=1}^{k_l} \lambda_i^l \beta_i$

fin pour

2 **retourner** $TML^* \longleftarrow \min_{(k_1,k_2,\ldots,k_m)\in\mathscr{K}} \{\sum C_{max}^*(k_l)\}$

fin

Théorème 5.8. *Le problème d'ordonnancement à machines parallèles*

$$R|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)}| \sum_{l=1}^{m} C_{max}^{l}$$

dont l'objectif est la minimisation de la charge totale des machines et de trouver le nombre de maintenances k optimal, peut être résolu avec une complexité $\mathcal{O}(n^{m+3})$, avec $k, m \leq n$. Si le nombre de maintenances est fixe (atelier Rm), le problème est polynomial avec une complexité $\mathcal{O}(n^{m+3})$.

Yang et *al.* [123] ont étudié un problème plus restrictif : des périodes de maintenance constantes à insérer sur les machines, deux modèles de détérioration indépendants du groupe ont été considérés $p^{(i)}(l,j,r) = p_{lj} r^{a_{lj}}$ et $p^{(i)}(l,j,r) = p_{lj} + a_{lj} r$. La résolution qu'ils proposent a une complexité $\mathcal{O}(n^{m+3})$, et est donc polynomiale si le nombre de machines m est fixe. Ainsi le théorème 5.8 généralise le problème étudié dans [123] en gardant toujours la même complexité, voir tableau 5.4.

5.6 Conclusion

Dans ce chapitre, nous avons étudié les problèmes d'ordonnancement simultané de tâches et de périodes de maintenance sur des machines parallèles. Tout d'abord, une approche générale a été présentée pour résoudre le problème en présence d'une politique de maintenance qui consiste à insérer plusieurs périodes de maintenance. Dans la section 5.4,

Tableau 5.4 – Comparaison avec les résultats antérieurs

Notre modèle	Modèle et complexité de la littérature		
	Réf.	Modèle	Complexité
∘ m machines parallèles ∘ Minimisation de la charge totale ∘ Temps opératoires avec détérioration : $p^{(i)}(l,j,r)$ avec $p^{(i)}(l,j,1) \leq \ldots \leq p^{(i)}(l,j,n)$ ∘ Système de maintenance $\mathcal{MP}^{(i)}$: $m_{li} = \sum_{j \in G_i^l} w^{(i)}(l,j,[j]) p^{(i)}(l,j,[j]) + \beta_{li}$ avec $w^{(i)}(l,j,1) \leq \ldots \leq w^{(i)}(l,j,n)$ ∘ Les conditions (5.23) et (5.24) sont vérifiées ∘ k périodes de maintenance à insérer ⇒ Comp. : $\mathcal{O}\left([max\{k,m\}]^{m-1}(n^3 + n^2(k+m))\right)$ voir théorème 5.7 ⇒ Si m est fixe et k une variable de décision, la complexité est : $\mathcal{O}(n^{m+3})$	[123]	∘ Atelier Rm ∘ $\sum_{l=1}^{m} C_{max}^{l}$ ∘ $p^{(i)}(l,j,r) = p_{lj} r^{a_{lj}}$ et $p^{(i)}(l,j,r) = p_{lj} + a_{lj} r$ avec $a_{lj} > 0$ ∘ $w^{(i)}(l,j,r) = 0$ et $\beta_{li} = \beta_l$ ∘ k périodes de maintenance	$\mathcal{O}(n^{m+3})$

nous étendons l'approche présentée précédemment afin de résoudre le problème dans un environnement qui dépend du groupe.

Ensuite, nous avons proposé une méthode de résolution améliorée pour résoudre la minimisation de la charge totale de la machine, en utilisant un problème d'affectation rectangulaire.

Dans le tableau 5.5, nous reprenons l'essentiel des résultats obtenus dans ce chapitre. Notons qu'une comparaison avec les résultats antérieurs a été présentée à la fin de chaque section.

Tableau 5.5 – Tableau récapitulatif des résultats

Problèmes	**Approche globale**	
	Complexités	Références
$R\|p(l,j,r), \mathcal{MP} = k\|\gamma$	$O([max\{k,m\}]^{m-1} n^{m+k+2})$	Théorème. 5.2
$R\|p(l,j,r), \mathcal{MP} = k\|\gamma_{cost}$	$O([max\{k,m\}]^{m-1} n^{m+k+2})$	Remarque. 5.1
$R\|p(l,j,r), \mathcal{MP} \leq k_0\|\gamma, \gamma_{cost}$	$O([max\{k_0,m\}]^{m-1} n^{m+k_0+2})$	Remarque. 5.2
$R\|p^{(i)}(l,j,r), \mathcal{MP}^{(i)} = k\|\gamma$	$O([max\{k,m\}]^{m-1} n^{m+k+2})$	Théorème. 5.4
$R\|p^{(i)}(l,j,r), \mathcal{MP}^{(i)} = k\|\gamma_{cost}$	$O([max\{k,m\}]^{m-1} n^{m+k+2})$	Remarque. 5.3
$R\|p^{(i)}(l,j,r), \mathcal{MP}^{(i)} \leq k_0\|\gamma, \gamma_{cost}$	$O([max\{k_0,m\}]^{m-1} n^{m+k_0+2})$	Théorème. 5.5
$R\|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)}\|\sum_{l=1}^{m} C_{max}^{l}$	$O(n^{m+3})$	Théorème. 5.8

Jusqu'à présent, nous avons proposé une approche de résolution globale pour les mo-

dèles dont le temps opératoire des tâches varie en fonction de la position sur l'ordonnancement $p(l, j, r)$. Dans le chapitre suivant, nous allons examiner plusieurs extensions et cas particuliers.

6

Cas particuliers

Résumé : Dans ce chapitre, nous étudions des cas particuliers issus du modèle général, présenté dans le chapitre 3. De plus, nous allons étudier une politique de maintenance optionnelle. Cette étude permet de résoudre des problèmes d'ordonnancement non traités avant et aussi de généraliser et améliorer des résultats antérieurs.

Sommaire

Ces travaux ont été réalisés en collaboration avec Gerd Finke. Les résultats développés dans ce chapitre ont été présentés dans les articles suivants :

[37] A. Gara-Ali, G. Finke, et M.-L. Espinouse. *Parallel-machine scheduling with maintenance : Praising the assignment problem.* European Journal of Operational Research **252**, 90 – 97 (2015).

[34] A. Gara-Ali, M.-L. Espinouse, et G. Finke. *Unrelated parallel-machine scheduling with deterioration effects and multi-maintenance activities.* In *CIE-44 - 44th International conference on Computers & Industrial Engineering*, 1080-1089, Istanbul, Turkey (2014).

[35] A. Gara-Ali, M.-L. Espinouse, et G. Finke. *Unrelated parallel-machine scheduling with optional maintenance activity on each machine and general effects of deterioration.* In *MOSIM 2014 - 10th International Conference on Modeling, Optimization & SIMulation*, Nancy, France (2014).

[36] A. Gara-Ali, M.-L. Espinouse, et G. Finke. *Problèmes d'ordonnancement sur des machines parallèles avec des périodes de maintenance et un effet de détérioration : Minimisation de la charge totale des machines.* In *ROADEF - 16ème congrès annuel de la Société Française de Recherche Opérationnelle et Aide à la Décision*, Marseille, France (2015).

6.1 Introduction

Dans cette section, nous nous intéressons à la résolution de plusieurs cas particuliers de problèmes d'ordonnancement en présence de périodes d'indisponibilité, afin d'améliorer les complexités de résolution par rapport aux résultats antérieurs.

Nous allons tout d'abord étudier une politique de maintenance optionnelle, dans laquelle au plus une période de maintenance est autorisée sur chaque machine. Nous présentons ensuite une méthode de résolution pour les problèmes d'ordonnancement équilibrés et pour les problèmes d'ordonnancement avec des périodes de maintenances sans durée élémentaire. Enfin, nous développons un algorithme de résolution amélioré pour les problèmes avec des temps opératoires décomposables.

6.2 Ordonnancement avec une période d'indisponibilité optionnelle

6.2.1 Minimisation du critère γ

Dans la littérature, un modèle réduit a été largement étudié dans lequel une seule période de maintenance optionnelle est autorisée sur chaque machine, voir [21, 55, 56, 114, 116, 117, 132]. Nous nous concentrons, dans cette section, sur cette configuration du problème avec le critère d'optimisation $\gamma = \{\sum_{l=1}^{m} C_{max}^{l}; \sum C_j; TW; TADC; TADW\}$. Nous considérons n tâches $\{T_1, T_2, \ldots, T_n\}$ à ordonnancer sur m machines parallèles $\{M_1, M_2, \ldots, M_m\}$. La durée de la tâche T_j ordonnancée au rang r sur la machine M_l est définie par $p^{(i)}(l, j, r)$. Au plus, chaque machine peut être maintenue une seule fois, la durée de cette période de maintenance sur la machine M_l, si elle existe, suit le système de maintenance $\mathcal{MP}^{(i)}$, *i.e.* :

$$m_l = \sum_{r=1}^{u_1^l} w^{(1)}(l, [l, 1, r], r) p^{(1)}(l, [l, 1, r], r) + \beta_l \tag{6.1}$$

Nous rappelons que u_1^l est le nombre de tâches ordonnancées avant la maintenance (groupe G_1^l), $[l, 1, r]$ représente la tâche ordonnancée au rang r du premier groupe sur la machine M_l et β_l la durée élémentaire de la maintenance optionnelle sur la machine M_l.

Dans la notation de Graham, le système des maintenances pondérées avec une période d'indisponibilité optionnelle sur chaque machine est noté par $\mathcal{MP}_l^{(i)} \leq 1$. Le problème d'ordonnancement étudié dans cette section est noté $R|p^{(i)}(l, j, r), \mathcal{MP}_l^{(i)} \leq 1|\gamma$, avec $\gamma = \{\sum_{l=1}^{m} C_{max}^{l}; \sum C_j; TW; TADC; TADW\}$.

Il est clair que la méthode de résolution, présentée dans la section 5.4 (chapitre 5),

permet de résoudre ce problème particulier. En effet, dans l'ordonnancement optimal, deux cas sont possibles pour chaque machine M_l $(l = 1, \ldots, m)$:

- $k_l = 1$: la période de maintenance est ordonnancée sur la machine M_l. Nous avons donc deux groupes de tâches tels que $u_1^l + u_2^l = n_l$, où u_1^l, u_2^l et n_l sont, respectivement, le nombre de tâches dans le premier et le deuxième groupe, sur la machine M_l
- $k_l = 0$: la période de maintenance n'est pas ordonnancée sur la machine M_l. Nous avons un seul groupe sur la machine avec $u_1^l = n_l$

Alors le nombre de vecteurs $(k_1, k_2, \ldots, k_m)$, tel que $k_l = \{1, 0\}$, est égal à $\mathcal{O}(2^m)$. Le nombre de vecteurs (u_1^l, u_2^l), tel que $u_1^l + u_2^l = n_l$ est de l'ordre de $\mathcal{O}(n_l - 1)$. Ainsi, le nombre de combinaisons sur l'ensemble des machines est :

$$\prod_{l=1}^{m} \mathcal{O}(n_l - 1) \leq \prod_{l=1}^{m} \mathcal{O}(n) \leq \mathcal{O}(n^m) \tag{6.2}$$

D'autre part, nous avons $\mathcal{O}(n^{m-1})$ combinaisons pour partitionner les n tâches sur les m machines telles que $n_1 + \ldots + n_m = n$ (cf. lemme 4.1). Ainsi, $\mathcal{O}(2^m \times n^{m-1} \times n^m) = \mathcal{O}(2^m n^{2m-1})$ problèmes d'affectation doivent être exécutés afin de trouver la solution optimale. D'où le résultat suivant :

Théorème 6.1. *Les problèmes d'ordonnancement à machines parallèles* $R|p^{(i)}(l,j,r), \mathcal{MP}_l^{(i)} \leq 1|\gamma$*, où* $\gamma = \{\sum_{l=1}^{m} C_{max}^l; \sum C_j; TW; TADC; TADW\}$*, peuvent être résolus avec une complexité* $\mathcal{O}(2^m n^{2m+2})$.

Remarque 6.1.

- *Dans un atelier d'ordonnancement avec un nombre de machines m fixe (Rm) : le problème* $Rm|p^{(i)}(l,j,r), \mathcal{MP}_l^{(i)} \leq 1|\gamma$ *est polynomial avec une complexité* $\mathcal{O}(n^{2m+2})$.
- *Dans un atelier d'ordonnancement à une machine* $m = 1$ *: le problème* $1|p^{(i)}(j,r), \mathcal{MP}^{(i)} \leq 1|\gamma$ *est polynomial avec une complexité* $\mathcal{O}(n^4)$.
- *En particulier, lorsque* $p^{(i)}(l,j,r) = p(l,j,r)$ *et* $w^{(i)}(l,j,r) = w(l,j,r)$ *(ordonnancements indépendants du groupe), le théorème 6.1 reste vrai.*
- *Le théorème 6.1 est aussi vérifié pour le critère d'optimisation* γ_{cost}.

Les résultats de cette section permettent de résoudre des problèmes non traités avant et aussi de généraliser et améliorer des résultats antérieurs. Le théorème 6.1 généralise les travaux de Hsu et *al.* [56], dans lesquels les auteurs ont étudié un modèle avec $p(l,j,r) = p_{lj} r^{a_{lj}}$, $a_{lj} \geq 0$, $w(l,j,r) = \alpha$ et $\beta_{li} = \beta$. Dans cette publication, la minimisation de la somme des dates de fin sur m machines parallèles a été résolue avec une complexité $\mathcal{O}(n^{2m+2})$. Le tableau 6.1 fournit une comparaison entre les résultats obtenus dans cette section et les résultats antérieurs.

Tableau 6.1 – Comparaison entre les résultats de la section et les résultats antérieurs – m constants

Notre approche globale	Résultats antérieurs		
	Modèles	Complexités	Réf.
∘ Une machine ∘ Critère d'optimisation γ ∘ Temps opératoires : $p^{(i)}(1,j,r)$ ∘ Système de maintenance optionnelle $\mathcal{MP}_l^{(i)} \leq 1$: $m_{li} = \sum_{j \in G_i^l} w^{(i)}(l,j,[j])p^{(i)}(l,j,[j]) + \beta_{li}$ $\Rightarrow$ Complexité : $\mathcal{O}(n^4)$, voir remarque 6.1	- une machine - $\gamma = \sum C_j$ - $p^{(1)}(1,j,r) = p_j$ - $p^{(2)}(1,j,r) = \delta_j p_j$ avec $\delta_j > 0$ - $w^{(1)}(1,j,r) = 0$	$\mathcal{O}(n^4)$	[75]
	- une machine - $\gamma = \sum C_j$ - $p^{(1)}(1,j,r) = a_j$ - $p^{(2)}(1,j,r) = b_j$ avec $b_j \leq a_j$ - $w^{(1)}(1,j,r) = \alpha$	$\mathcal{O}(n^4)$	[90]
∘ m machines parallèles (Rm) ∘ Critère d'optimisation γ ∘ Temps opératoires : $p^{(i)}(l,j,r)$ ∘ Système de maintenance optionnelle $\mathcal{MP}_l^{(i)} \leq 1$: $m_{li} = \sum_{j \in G_i^l} w^{(i)}(l,j,[j])p^{(i)}(l,j,[j]) + \beta_{li}$ $\Rightarrow$ Complexité : $\mathcal{O}(n^{2m+2})$ voir théorème 6.1	- deux machines ($P2$) - $\gamma = \sum C_j$ - $p^{(i)}(1,j,r) = p_j$ - $p^{(2)}(1,j,r) = \delta_j p_j$ avec $\delta_j > 0$ - $w^{(1)}(l,j,r) = 0$, $\beta_l = \beta$	$\mathcal{O}(n^6)$	[139]
	- m machines (Pm) - $\gamma = \sum C_j$ - $p^{(1)}(l,j,r) = p_j$ - $p^{(2)}(l,j,r) = \delta_j p_j$ avec $0 < \delta_j \leq 1$ - $w^{(1)}(l,j,r) = \alpha_l$	$\mathcal{O}(n^{2m+3})$	[116]
	- m machines (Pm) - $\gamma = \{TADC; TADW\}$ - $p^{(i)}(l,j,r) = a_{lj}$ - $p^{(2)}(l,j,r) = b_{lj}$ avec $b_{lj} \leq a_{lj}$ - $w^{(1)}(l,j,r) = \alpha_l$	$\mathcal{O}(n^{2m+2})$	[116]
	- m machines (Rm) - $\gamma = \{\sum C_{max}^l; \sum C_j\}$ - $p(l,j,r) = p_{lj} + a_{lj}r$ - ou $p(l,j,r) = p_{lj}r^{a_{lj}}$ avec $a_{lj} > 0$ - $w^{(1)}(l,j,r) = \alpha_l$	$\mathcal{O}(n^{2m+2})$	[56]

Nous nous intéressons dans la sous-section suivante à la minimisation de la charge totale des machines avec une période de maintenance optionnelle sur chaque machine.

6.2.2 Minimisation de la charge totale des machines

Dans la section 5.5, nous avons étudié la minimisation de la charge totale des machines avec la prise en compte d'un effet de détérioration et en présence de plusieurs périodes de maintenance sur chaque machine. Notons que quatre conditions ont été imposées : (5.21), (5.23), (5.22) et (5.24). Nous avons montré que lorsque le nombre de maintenances k_l sur chaque machine M_l est connu d'avance, le problème d'ordonnancement à machines parallèles $R|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)}(k_l)|\sum_{l=1}^{m} C_{max}^{l}$ peut être résolu en temps polynomial $O(n^3 + n^2(k+m))$ avec $k = \sum_{l=1}^{m} k_l$, (cf. théorème 5.6).

Nous nous focalisons, dans cette sous-section, sur la même configuration, mais avec une période de maintenance optionnelle sur chaque machine, le problème d'ordonnancement est noté par $R|p^{(i)}(l,j,r) - det, \mathcal{MP}_l^{(i)} \leq 1|\sum_{l=1}^{m} C_{max}^{l}$.

Avec une configuration de maintenance optionnelle, le nombre de maintenances k_l sur la machine M_l est égal à 0 ou 1. Ainsi, pour résoudre le problème avec des périodes de maintenance optionnelle, il suffit de résoudre le problème $R|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)}(k_l)|\sum_{l=1}^{m} C_{max}^{l}$ pour tous les vecteurs $(k_1, k_2, \ldots, k_m)$ tels que $k_l = \{0,1\}$, $\forall\ l = 1, \ldots, m$, voir Algorithme 5.

Algorithme 5 : Problèmes d'ordonnancement $R|p^{(i)}(l,j,r) - det, \mathcal{MP}_l^{(i)} \leq 1|\sum_{l=1}^{m} C_{max}^{l}$

début

Soit $\mathscr{K}$ l'ensemble de vecteurs $(k_1, k_2, \ldots, k_m)$ tel que $k_l = \{0,1\}$ `/* 2^m vecteurs possibles */`

1 **pour chaque** $(k_1, k_2, \ldots, k_m) \in \mathscr{K}$ **faire**

1.1 $\sum C_{max}^{*}(k_l) \longleftarrow$ *Résoudre* $\left(R|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)}(k_l)|\sum_{l=1}^{m} C_{max}^{l}\right)$ `/* cf. section 5.5 */`

fin pour

2 **retourner** $TML^{*} \longleftarrow \min_{(k_1,k_2,\ldots,k_m)\in\mathscr{K}}\{\sum C_{max}^{*}(k_l)\}$

fin

D'après le théorème 5.6, lorsque le vecteur $(k_1, k_2, \ldots, k_m)$ est donné, la complexité de résolution du problème $R|p^{(i)}(l,j,r) - det, \mathcal{MP}^{(i)}(k_l)|\sum_{l=1}^{m} C_{max}^{l}$ est $\mathcal{O}(n^3 + n^2(\sum_{l=1}^{m} k_l + m)) \leq \mathcal{O}(n^3 + 2n^2 m)$, car $k_l \leq 1$ (au plus une période de maintenance sur la machine M_l). D'autre part, le nombre de vecteurs $(k_1, k_2, \ldots, k_m)$ possibles tel que $k_l = \{1,0\}$ est égal à 2^m. D'où le résultat suivant :

Théorème 6.2. *Les problèmes d'ordonnancement à machines parallèles :* $R|p^{(i)}(l,j,r) - det, \mathcal{MP}_l^{(i)} \leq 1| \sum_{l=1}^{m} C_{max}^l$ *peuvent être résolus avec une complexité :* $\mathcal{O}(2^m(n^3 + 2n^2m))$.

Par conséquent, lorsque le nombre de machines m est fixe, le problème d'ordonnancement $Rm|p^{(i)}(l,j,r) - det, \mathcal{MP}_l^{(i)} \leq 1| \sum_{l=1}^{m} C_{max}^l$ peut être résolu avec une complexité $\mathcal{O}(n^3)$.

En appliquant le même principe, ce résultat est vrai aussi pour $m = 1$, d'où le théorème suivant :

Théorème 6.3. *Les problèmes d'ordonnancement dans un atelier à une machine* $1|p^{(i)}(j,r) - det, \mathcal{MP}^{(i)} \leq 1|C_{max}$ *peuvent être résolus avec une complexité* $\mathcal{O}(n^3)$.

Le théorème 6.2 améliore et généralise le résultat de Hsu et *al.* [56], dans lequel, les auteurs ont étudié deux modèles réduits avec $p^{(i)}(l,j,r) = p_{lj}r^{a_{lj}}$ ou $p^{(i)}(l,j,r) = p_{lj} + a_{lj}r$, $a_{lj} \geq 0$, $w^{(i)}(l,j,r) = \alpha$ et $\beta_{li} = \beta$. L'objectif est la minimisation de la charge totale de m machines parallèles. Les auteurs ont montré que le problème peut être résolu avec une complexité $\mathcal{O}(n^{2m+2})$.

6.3 Les ordonnancements équilibrés

Nous étudions, dans cette sous-section, les problèmes d'ordonnancement dans un système de production équilibré et avec comme objectif la minimisation de la charge totale des machines. Rappelons qu'un problème d'ordonnancement est dit équilibré si le nombre de tâches dans chaque groupe diffère d'au plus une tâche.

> “ *Group balance principle : The group balance principle is to make the number of jobs to be as equal as possible in each group. That is, the number of jobs in each group is either* $h-1$ *or* h*...* ”
>
> W.-H. Kuo et D.-L. Yang, *[71]* , 2008

Pour la minimisation de la charge totale des machines $\sum_{l=1}^{m} C_{max}^l$ avec un effet de détérioration sur les machines, il existe toujours un ordonnancement optimal qui vérifie la propriété d'équilibrage en présence d'un système de maintenance $\mathcal{MP}$ fermé [1] et des coefficients $w(l,j,r)$ croissants selon r. Nous pouvons vérifier cette propriété en utilisant l'argument d'échange.

1. Nous définissons un système de maintenance fermé comme suit : en tant qu'utilisateur, nous commençons le traitement avec une machine bien entretenue et à la fin nous rendons la machine en parfait état, *i.e.* nous finissons l'ordonnancement avec une période d'entretien. Contrairement, au système ouvert qui ne comporte pas de maintenance à la fin. Dans notre notation, nous fixons $u_{k_l+1}^l = 0$.

Démonstration. Soit S un ordonnancement optimal et non équilibré, donc il existe deux groupes G_x et G_y $(x, y \leq k_l)$ de taille u_x^l et u_y^l sur la machine M_l tels que la différence entre les tailles de ces deux groupes est supérieure ou égale à deux tâches $u_x^l - u_y^l \geq 2$. Soit S' le même ordonnancement en déplaçant la dernière tâche T_j du groupe G_x vers le groupe G_y. On sait que $p(l, j, r)$ et $w(l, j, r)$ sont croissants selon r. Cet échange permet, donc, de réduire le makespan C_{max}^l sur la machine M_l, puisque :

$$\Big(1 + w(l, j, u_x^l)\Big)\, p(l, j, u_x^l) - \Big(1 + w(l, j, u_y^l + 1)\Big)\, p(l, j, u_y^l + 1) \geq 0$$

□

On définit le problème d'ordonnancement étudié par $R|p(l, j, r)-det, c-\mathcal{MP}|\sum_{l=1}^{m} C_{max}$, où $c - \mathcal{MP}$ indique que le système de maintenance est fermé.

Soit π_l la séquence des tâches de la machine M_l avec n_l et k_l, respectivement, le nombre de tâches et de maintenances sur la machine. Nous obtenons une répartition de k_l groupes sur la machine M_l, puisque on impose que $u_{k_l+1}^l = 0$. Par définition, la séquence π_l est équilibrée sur la machine M_l, si le nombre de tâches dans chaque groupe est égal à $h - 1$ ou h (diffère d'au plus une tâche), avec $h = \lceil \frac{n_l}{k_l} \rceil$. La séquence équilibrée est constituée de $k_l h - n_l$ groupes de taille $h - 1$ et $n_l - k_l(h - 1)$ groupes de taille h.

Exemple 6.1. *Soit un ordonnancement avec $n = 11$ tâches et deux machines. Les nombres de maintenances et de tâches sur la première et la deuxième machine sont, respectivement, $k_1 = 3$, $k_2 = 4$, $n_1 = 5$ et $n_2 = 6$.*

Cet ordonnancement est équilibré si nous avons la répartition suivante :

- *sur la machine M_1 : deux groupes de deux tâches et un groupe avec une tâche, car $h = \lceil \frac{5}{3} \rceil = 2$.*
- *sur la machine M_2 : deux groupes de deux tâches et deux groupes avec une tâche chacun, car $h = \lceil \frac{6}{4} \rceil = 2$.*

Notons que l'ordre des groupes sur chaque machine n'affecte pas l'optimalité de la solution, puisque les paramètres sont indépendants du groupe et après chaque période de maintenance la machine revient à son état initial.

Ainsi, pour un nombre de maintenances $(k_1, k_2, \ldots, k_m)$ et de tâches $(n_1, n_2, \ldots, n_m)$ donnés, nous pouvons déterminer le vecteur taille des groupes optimal $(u_1^l, \ldots, u_{k_l}^l, u_{k_l+1}^l = 0), \forall\, l = 1, \ldots, m$, en utilisant la propriété d'équilibrage, contrairement au problème étudié dans la section 5.3, dans lequel nous énumérons toutes les combinaisons possibles avec une complexité $\mathcal{O}(n^k)$ (cf. équation 5.12). Rappelons que nous imposons que la taille du dernier groupe sur chaque machine soit vide $u_{k_l+1}^l = 0$, car le système de maintenance

est fermé. Ensuite, la séquence optimale de tâches est obtenue en résolvant le problème $R|p(l,j,r),\mathcal{MP}(n_l,k_l,u^l)|\sum_{l=1}^{m} C_{max}^{l}$ (voir PLNE.6), avec une complexité $\mathcal{O}(n^3)$. On note que dans le système de maintenance fermé, la charge totale C_{max}^{l} d'une machine M_l est mesurée jusqu'à la fin de la dernière période de maintenance.

Pour trouver la solution optimale du problème $R|p(l,j,r)-det,c-\mathcal{MP}=k|\sum_{l=1}^{m} C_{max}^{l}$, nous aurions besoin de résoudre un grand nombre de problèmes d'affectation. Comme cela était précisé précédemment, le nombre de combinaisons possibles pour partitionner les tâches sur les m machines est $\mathcal{O}(n^{m-1})$ tel que $n_1+n_2+\ldots+n_m=n$. Le nombre de vecteurs $(k_1,k_2,\ldots,k_m)$ tel que $k_1+k_2+\ldots+k_m=k$ est $\mathcal{O}([max\{k,m\}]^{m-1})$.

Donc, il y a au total $\mathcal{O}([max\{k,m\}]^{m-1}n^{m-1})$ problèmes d'affectation qui devraient être résolus.

Théorème 6.4. *Les problèmes d'ordonnancement à machines parallèles avec un système de maintenance fermé*

$$R|p(l,j,r)-det,c-\mathcal{MP}=k|\sum_{l=1}^{m} C_{max}^{l}$$

possèdent un ordonnancement équilibré optimal, et peuvent être résolus avec une complexité $\mathcal{O}([max\{k,m\}]^{m-1}n^{m+2})$, où n, m et k sont, respectivement, le nombre de tâches, de machines et de maintenances.

Nous proposons l'algorithme 6 pour résoudre le problème ci-dessus.

Si le nombre de machines m et le nombre de maintenances k sont constants, *i.e.* le problème $Rm|p(l,j,r)-det,c-\mathcal{MP}=k|\sum_{l=1}^{m} C_{max}^{l}$ est considéré, nous obtenons la complexité polynomiale $\mathcal{O}(n^{m+2})$.

Remarque 6.2. *Pour le théorème 6.4, la condition du système fermé n'est pas une condition nécessaire. En effet, les problèmes d'ordonnancement $R|p(l,j,r)-det,\mathcal{MP}=k|\sum_{l=1}^{m} C_{max}^{l}$ possèdent aussi une solution optimale équilibrée dans un système de maintenance ouvert, si la durée des périodes de maintenances est constante, i.e. $w(l,j,r)=0\ \forall\ l,j$ et r.*

Démonstration. Nous pouvons prouver cette propriété en utilisant un argument d'échange. □

Maintenant, nous présentons une deuxième méthode pour résoudre le problème d'ordonnancement $R|p(l,j,r)-det,c-\mathcal{MP}=k|\sum_{l=1}^{m} C_{max}^{l}$. Rappelons qu'un effet de détérioration est appliqué sur les machines $(p(l,j,1)\leq p(l,j,2)\leq\ldots\leq p(l,j,n))$, que le système de maintenance est fermé, *i.e.* une maintenance est insérée à la fin de l'ordonnancement sur chaque machine M_l, on impose $u_{k_l+1}^{l}=0$ et que les coefficients $w(l,j,r)$ sont croissants

Algorithme 6 : Problèmes d'ordonnancement équilibrés

Données :
- $p(l, j, r)$ temps opératoire de la tâche T_j au rang r sur la machine M_l
- $w(l, j, r)$ coefficient de détérioration de la maintenance due à l'ordonnancement de la tâche T_j au rang r sur la machine M_l
- β_{li} durée élémentaire de la maintenance i sur la machine M_l
- $\mathscr{N}$ ensemble de vecteurs $(n_1, n_2, \ldots, n_m)$ possibles tels que $n_1 + n_2 + \ldots + n_m = n$
- $\mathscr{K}$ ensemble de vecteurs $(k_1, k_2, \ldots, k_m)$ possibles tels que $k_1 + k_2 + \ldots + k_m = k$

Résultat : $\sum_{l=1}^{m} C_{max}^{l*}$ la solution optimale

début

1 **pour chaque** $(n_1, n_2, \ldots, n_m) \in \mathscr{N}$ **faire**

1.1 **pour chaque** $(k_1, k_2, \ldots, k_m) \in \mathscr{K}$ **faire**

1.1.1 Déterminer la taille de chaque groupe $(u_1^l, u_2^l, \ldots, u_{k_l}^l, u_{k_l+1}^l = 0)$, en utilisant la propriété d'équilibrage

1.1.2 $\sum_{l=1}^{m} C_{max}^{l*}(n_l, k_l, u^l) \longleftarrow$ *Résoudre* $R|p(l, j, r), \mathcal{MP}(n_l, k_l, u^l)| \sum_{l=1}^{m} C_{max}^{l}$

```
/* Résoudre PLNE.6, rappelons que ∀l = 1,...,m, u^l_{k_l+1} = 0,
i.e. système de maintenance fermé
 */
```

fin pour

fin pour

2 **retourner** $\sum_{l=1}^{m} C_{max}^{l*} \longleftarrow \min\{\sum_{l=1}^{m} C_{max}^{l*}(n_l, k_l, u^l)\}$

fin

selon r. Nous rappelons aussi que pour ce problème, il existe une solution optimale qui vérifie la propriété d'équilibrage.

L'idée de cette deuxième méthode est la suivante : pour chaque machine M_l avec un nombre de maintenances k_l donné, nous supposons que $n_l = n$, *i.e.* n positions sont disponibles sur la machine M_l. Ensuite, nous utilisons la propriété d'équilibrage pour trouver les tailles des groupes $h - 1$ ou h, sur la base de $n_l = n$, avec $h = \lceil \frac{n}{k_l} \rceil$. Ces tailles sont des bornes supérieures, car nous avons considéré que toutes les tâches sont sur la même machine. Nous pouvons ainsi modéliser le problème sous forme d'un problème d'affectation rectangulaire, avec n lignes (les tâches) et mn colonnes (sur chaque machine M_l, n positions réparties sur des groupes de tailles $\lceil \frac{n}{k_l} \rceil$ et $\lceil \frac{n}{k_l} \rceil - 1$). Selon Dinic [25], la résolution de ce problème d'affectation rectangulaire est de complexité $O(n^3 + n \times nm) = O(n^3 + n^2 m)$, cf. section 3.4.2.

Il est clair que le nombre de positions nm est plus grand que le nombre de tâches n. Donc nous obtenons après l'affectation $nm - n = n(m - 1)$ positions vides. Les positions affectées seront placées au début de chaque groupe et les positions vides à la fin, car les coûts d'affectation sont croissants en fonction de la position r pour chaque groupe :

$$(1 + w(l, j, 1))p(l, j, 1) \leq (1 + w(l, j, 2))p(l, j, 2) \leq \ldots \leq (1 + w(l, j, n))p(l, j, n) \quad (6.3)$$

Cependant, des groupes vides peuvent être obtenus dans la solution du problème d'affectation rectangulaire. Dans ce cas, il suffit de supprimer ces groupes vides pour avoir une solution réalisable sans modifier son optimalité. Ensuite, les durées élémentaires des maintenances doivent être ajoutées à la solution obtenue.

Pour trouver la solution finale, nous devons résoudre cette affectation pour chaque vecteur $(k_1, k_2, \ldots, k_m)$ tel que $k_1 + k_2 + \ldots + k_m = k$. Le nombre des combinaisons possibles est égal à $O([max\{k, m\}]^{m-1})$. D'où le théorème suivant :

Théorème 6.5. *Les problèmes d'ordonnancement à machines parallèles avec un système de maintenance fermé*

$$R|p(l,j,r) - det, c - \mathcal{MP} = k| \sum_{l=1}^{m} C_{max}^{l}$$

possèdent un ordonnancement équilibré optimal, et peuvent être résolus avec une complexité $\mathcal{O}([max\{k, m\}]^{m-1}(n^3 + n^2 m))$*, où* n*,* m *et* k *sont, respectivement, le nombre de tâches, de machines et de maintenances.*

Si le nombre de machines m et le nombre de maintenances k sont constants, *i.e.* le problème $Rm|p(l,j,r) - det, c - \mathcal{MP} = k| \sum_{l=1}^{m} C_{max}^{l}$, on obtient la complexité polynomiale $\mathcal{O}(n^3)$.

Pour bien comprendre le principe de la résolution, nous allons résoudre l'exemple suivant :

Exemple 6.2. *Soit un ordonnancement avec* $n = 8$ *tâches et deux machines. Soit* $k = 5$ *avec* $k_1 = 2$ *et* $k_2 = 3$*. Nous présentons, dans les tableaux 6.2, 6.3, 6.4 et 6.5, les durées des tâches et les coefficients de détérioration de maintenances :*

Tableau 6.2 – Temps opératoires des tâches sur M_1

	$p(1,j,r)$							
$r =$	1	2	3	4	5	6	7	8
T_1	1	2	3	3	3	3	3	3
T_2	1	2	3	3	3	3	3	3
T_3	2	3	4	4	4	4	4	4
T_4	2	3	4	4	4	4	4	4
T_5	1	2	3	3	3	3	3	3
T_6	2	3	4	4	4	4	4	4
T_7	1	2	3	3	3	3	3	3
T_8	2	3	4	4	4	4	4	4

Tableau 6.3 – Temps opératoires des tâches sur M_2

	$p(2,j,r)$							
$r =$	1	2	3	4	5	6	7	8
T_1	2	3	4	5	5	5	5	5
T_2	1	2	3	3	3	3	3	3
T_3	1	2	3	3	3	3	3	3
T_4	2	3	4	5	5	5	5	5
T_5	2	3	4	4	4	4	4	4
T_6	1	2	3	3	3	3	3	3
T_7	2	3	4	4	4	4	4	4
T_8	2	3	4	4	4	4	4	4

Soit $\beta_{li} = 1 \; \forall \; l$ *et* i*. Afin d'assurer l'équilibrage du système, nous considérons que le système de maintenance est fermé. Les tailles des groupes sont :*

Tableau 6.4 – Coefficients de détérioration sur M_1

	$w(1,j,r)$							
$r=$	1	2	3	4	5	6	7	8
T_1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
T_2	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
T_3	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
T_4	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
T_5	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
T_6	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
T_7	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
T_8	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1

Tableau 6.5 – Coefficients de détérioration sur M_2

	$w(2,j,r)$							
$r=$	1	2	3	4	5	6	7	8
T_1	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2
T_2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2
T_3	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2
T_4	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2
T_5	0,2	0,2	0,2	0,2	0,3	0,3	0,3	0,3
T_6	0,2	0,2	0,2	0,2	0,3	0,3	0,3	0,3
T_7	0,2	0,2	0,2	0,2	0,3	0,3	0,3	0,3
T_8	0,2	0,2	0,2	0,2	0,3	0,3	0,3	0,3

- *Machine 1 :* $h = \lceil \frac{n}{k_l} \rceil = \lceil \frac{8}{2} \rceil = 4$. *Ainsi, sur la machine* M_1 *nous avons deux groupes de* 4 *positions chacun.*
- *Machine 2 :* $h = \lceil \frac{n}{k_l} \rceil = \lceil \frac{8}{3} \rceil = 3$. *Ainsi, sur la machine* M_2 *nous avons deux groupes de* 3 *positions chacun et un groupe de* 2 *positions.*

Nous présentons dans le diagramme de Gantt (figure 6.1) la répartition des positions dans chaque groupe et sur chaque machine.

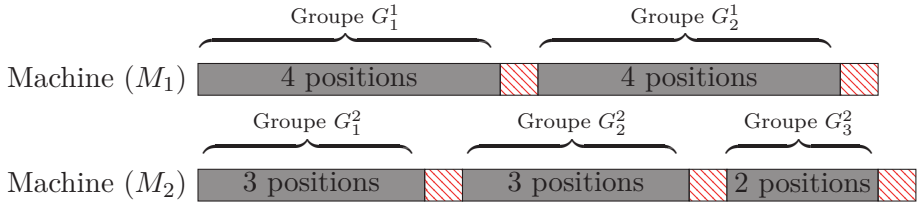

Figure 6.1 – Réparation des tâches et des groupes sur les machines

Notons que pour la minimisation de la charge totale de la machine, la position du groupe n'affecte pas l'optimalité de la solution. Maintenant, nous devons résoudre le problème d'affectation rectangulaire de taille $n \times nm = 8 \times 16$. *Comme cela a été présenté précédemment, le coût d'affectation d'une tâche* T_j *à une position* r *du groupe* G_i *sur la machine* M_l *est égal à* $V_j(l,i,r) = (1 + w(l,j,r))p(l,j,r)$. *La matrice d'affectation est présentée dans la figure 6.2.*

Les colonnes représentent les positions et les lignes indiquent les tâches. La résolution de ce problème d'affectation permet de trouver la solution optimale $V^* = 14,9$. *Le terme constant de la maintenance est* $\sum_{l=1}^{m} \sum_{i=1}^{k_l} \beta_{li} = 2 + 3 = 5$.

Par conséquent, la charge totale optimale est $\sum_{l=1}^{m} C_{max}^{l} = V^* + \sum_{l=1}^{m} \sum_{i=1}^{k_l} \beta_{li} = 14,9 + 5 = 19,9$.

$$M = \begin{array}{c} \overbrace{}^{G_1^1}\ \overbrace{}^{G_2^1}\ \overbrace{}^{G_1^2}\ \overbrace{}^{G_2^2}\ \overbrace{}^{G_3^2} \\ \left(\begin{array}{cccccccccccccc} \boxed{1,1} & 2,2 & 3,3 & 3,3 & 1,1 & 2,2 & 3,3 & 3,3 & 2,4 & 3,6 & 4,8 & 2,4 & 3,6 & 4,8 & 2,4 & 3,6 \\ 1,1 & 2,2 & 3,3 & 3,3 & 1,1 & 2,2 & 3,3 & 3,3 & 1,2 & 2,4 & 3,6 & \boxed{1,2} & 2,4 & 3,6 & 1,2 & 2,4 \\ 2,2 & 3,3 & 4,4 & 4,4 & 2,2 & 3,3 & 4,4 & 4,4 & 1,2 & 2,4 & 3,6 & 1,2 & 2,4 & 3,6 & \boxed{1,2} & 2,4 \\ 2,2 & 3,3 & 4,4 & 4,4 & 2,2 & 3,3 & 4,4 & 4,4 & 2,4 & \boxed{3,6} & 4,8 & 2,4 & 3,6 & 4,8 & 2,4 & 3,6 \\ 1,1 & 2,2 & 3,3 & 3,3 & \boxed{1,1} & 2,2 & 3,3 & 3,3 & 2,4 & 3,6 & 4,8 & 2,4 & 3,6 & 4,8 & 2,4 & 3,6 \\ 2,2 & 3,3 & 4,4 & 4,4 & 2,2 & 3,3 & 4,4 & 4,4 & \boxed{1,2} & 2,4 & 3,6 & 1,2 & 2,4 & 3,6 & 1,2 & 2,4 \\ 1,1 & \boxed{2,2} & 3,3 & 3,3 & 1,1 & 2,2 & 3,3 & 3,3 & 2,4 & 3,6 & 4,8 & 2,4 & 3,6 & 4,8 & 2,4 & 3,6 \\ 2,2 & 3,3 & 4,4 & 4,4 & 2,2 & \boxed{3,3} & 4,4 & 4,4 & 2,4 & 3,6 & 4,8 & 2,4 & 3,6 & 4,8 & 2,4 & 3,6 \end{array}\right)_{n} \\ \underbrace{}_{\text{Machine } M_1}\ \underbrace{}_{\text{Machine } M_2} \end{array}$$

Figure 6.2 – La matrice d'affectation

L'ordonnancement optimal est présenté dans la figure 6.3, ci-dessous.

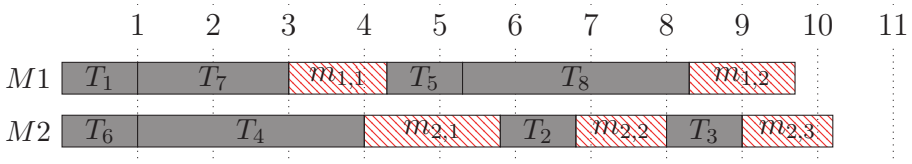

Figure 6.3 – Exemple 2 – Ordonnancement optimal

Les durées des maintenances sont :

- $m_{1,1} = w(1,1,1)p(1,1,1) + w(1,7,2)p(1,7,2) + \beta_{1,1} = 1,3$
- $m_{1,2} = w(1,5,1)p(1,5,1) + w(1,8,2)p(1,8,2) + \beta_{1,2} = 1,4$
- $m_{2,1} = w(2,6,1)p(2,6,1) + w(2,4,2)p(2,4,2) + \beta_{2,1} = 1,8$
- $m_{2,2} = w(2,2,1)p(2,2,1) + \beta_{2,2} = 1,2$
- $m_{2,3} = w(2,3,1)p(2,3,1) + \beta_{2,3} = 1,2$

6.4 Maintenance sans durée constante ($\beta_{li} = 0$)

Dans cette section, nous étudions un cas particulier du problème étudié dans la section 5.5, mais dans sa version indépendante du groupe. Nous considérons :

– le système de maintenance $\mathcal{MP}$ avec des périodes de maintenance sans durée élémentaire ($\beta_{li} = 0$)

– les coefficients de maintenance sont croissants selon r :

$$w(l,j,1) \leq w(l,j,2) \leq \ldots \leq w(l,j,n) \qquad \forall\, j = 1,\ldots,n,\ et\ l = 1,\ldots,m \tag{6.4}$$

– un effet de détérioration est appliqué sur les machines :

$$p(l,j,1) \leq p(l,j,2) \leq \ldots \leq p(l,j,n) \qquad \forall\, j = 1,\ldots,n,\ et\ l = 1,\ldots,m \tag{6.5}$$

– l'objectif est la minimisation de la charge totale des machines ($\sum_{l=1}^{m} C_{max}^{l}$)

Rappelons que, précédemment, nous avons montré que le coût d'affectation d'une tâche T_j à une position (l, i, r) est égal à $V_j(l,i,r) = \begin{cases} (1 + w(l,j,r))p(l,j,r) & \text{si } i \leq k_l \\ p(l,j,r) & \text{si } i = k_l + 1 \end{cases}$

Les temps opératoires $p(l, j, r)$ et les coefficients de maintenance $w(l, j, r)$ sont croissants selon r, voir les expressions 6.4 et 6.5, alors nous obtenons :

$$V_j(l,i,1) \leq V_j(l,i,2) \leq \ldots \leq V_j(l,i,n),\ \forall\ j,\ l \text{ et } i$$

Nous pouvons modéliser ce problème sous forme d'un problème d'affectation rectangulaire, en attribuant les k périodes de maintenance à chaque machine et en considérant que la taille de chaque groupe est égale à n. Ainsi, la taille de la matrice d'affectation est $n \times n(k+1)m$. Il est évident que la durée d'une période de maintenance est nulle, si elle suit directement une autre période de maintenance, *i.e.* le groupe de tâches qui précède cette maintenance est vide. Le nombre de périodes de maintenance est limité à $k \leq n$. Nous pouvons, donc, fixer $k = n$ et résoudre un simple problème d'affectation rectangulaire de taille $n \times n(n+1)m$ en $\mathcal{O}(n^3 + n \times n(n+1)m) = \mathcal{O}(n^3 + mn^3 + mn^2)$ (cf. section 3.4.2), soit une complexité de $\mathcal{O}(mn^3)$.

Ensuite, nous supprimons les périodes de maintenance dont les durées sont nulles pour trouver la séquence de tâches et le nombre de maintenances de la solution optimale.

Théorème 6.6. *Les problèmes d'ordonnancement à machines parallèles avec* $\beta_{li} = 0$

$$R|p(l,j,r) - det, \mathcal{MP}, \beta_{li} = 0| \sum_{l=1}^{m} C_{max}^{l},$$

dont l'objectif est de trouver la séquence de tâches qui minimise $\sum_{l=1}^{m} C_{max}^{l}$ *et le nombre de maintenances optimal, peuvent être modélisés comme des problèmes d'affectation rectangulaire et résolus en* $O(mn^3)$.

Lorsque le nombre de machines m est fixe, le problème $Rm|p(l,j,r) - det, \mathcal{MP}, \beta_{li} = 0| \sum_{l=1}^{m} C_{max}^{l}$ est polynomial avec une complexité $\mathcal{O}(n^3)$. En particulier pour $m = 1$, nous pouvons résoudre le problème $1|p(j,r) - det, \mathcal{MP}, \beta_i = 0| \sum_{l=1}^{m} C_{max}^{l}$ en $\mathcal{O}(n^3)$.

Remarque 6.3. *Nous notons que le théorème 6.6 est vrai pour tous les critères d'optimisation qui vérifient :*

$$V_j(l,i,1) \leq V_j(l,i,2) \leq \ldots \leq V_j(l,i,n) \quad \forall\ j,\ l,\ i \tag{6.6}$$

En effet, cette propriété assure que les tâches seront affectées dans des positions adjacentes à partir du rang $r = 1$ *du groupe. Notons que les critères* $\gamma = \{\sum C_j, \sum W_j, TADC, TADW\}$ *ne vérifient pas cette propriété.*

6.5 Fonction décomposable $p^{(i)}(l,j,r) = p_j g(l,i,r)$

Le temps opératoire $p^{(i)}(l,j,r)$ d'une tâche T_j est dit décomposable, s'il est de la forme $p^{(i)}(l,j,r) = p_j g(l,i,r)$, avec p_j le temps intrinsèque de la tâche T_j et $g(l,i,r)$ une fonction quelconque. Lorsque g est une fonction croissante selon r, la machine est sous un effet de détérioration. Si la fonction g est décroissante selon r, la machine connaît un effet d'apprentissage, voir Moncel et *al.*, [83]. Les temps opératoires utilisés dans la littérature sont souvent décomposables tels que : $p_j r^a$, $p_j \sigma^{r-1}$.

Lemme 6.1 (Hardy et *al.*, [50]). *Soit deux vecteurs $(x_1, x_2, \ldots, x_n)$ et $(y_1, y_2, \ldots, y_n)$ donnés. La minimisation du produit scalaire $\sum x_i y_{\pi(i)}$, avec π une permutation, est obtenue si l'un de ces vecteurs est trié dans l'ordre croissant et l'autre dans l'ordre décroissant.*

Ainsi, la minimisation de $\sum x_i y_{\pi(i)}$ est polynomiale avec une complexité $\mathcal{O}(nlog(n))$.

Nous nous intéressons, dans cette section, aux problèmes d'ordonnancement avec des fonctions décomposables. Nous considérons aussi une configuration dépendante du groupe. Le problème étudié est le suivant :

- n tâches à ordonnancer avec des temps opératoires dépendants du groupe $p^{(i)}(l,j,r)$ et qui suivent une fonction décomposable de la forme :

$$p^{(i)}(l,j,r) = p_j g(l,i,r) \tag{6.7}$$

- un système de maintenance $\mathcal{MP}^{(i)}$ avec des coefficients de maintenance $w^{(i)}(l,r)$ dépendants du groupe, indépendants des tâches et croissants selon r :

$$w^{(i)}(l,1) \leq w^{(i)}(l,2) \leq \ldots \leq w^{(i)}(l,n), \quad \forall\, l \text{ et } i \tag{6.8}$$

- k périodes de maintenance à insérer sur l'ensemble des machines
- l'objectif est la minimisation de γ, avec $\gamma = \{\sum_{l=1}^{m} C_{max}^l, \sum C_j, \sum W_j, TADC, TADW\}$

Précédemment dans la section 5.4, nous avons montré que lorsque les vecteurs $(k_1, \ldots, k_m)$, $(n_1, n_2, \ldots, n_m)$ et $(u_1^l, u_2^l, \ldots, u_{k_l+1}^l)$ sont connus d'avance, nous pouvons modéliser le problème sous forme d'un problème d'affectation linéaire (cf. Théorème 5.3). Dans le cas où les temps opératoires sont décomposables, le coût d'affectation $V_j(l,i,r)$ peut être écrit sous la forme :

$$V_j(l,i,r) = \psi(l,i,r) p_j \tag{6.9}$$

où $\psi(l,i,r)$ est un terme qui dépend seulement de la machine M_l, du rang r dans le groupe G_i^l.

Ci-dessous, nous rapportons les coûts d'affectation de différents critères d'optimisation étudiés :

La charge totale des machines $\sum_{l=1}^{m} C_{max}^{l}$
D'après l'expression (5.15), nous avons :

$$\sum_{l=1}^{m} C_{max}^{l} = \sum_{l=1}^{m} \sum_{i=1}^{k_l+1} \sum_{r=1}^{u_i^l} V_{[l,i,r]}(l,i,r) + \Phi(n,k)$$

avec $V_j(l,i,r) = \begin{cases} (1+w^{(i)}(l,r))p^{(i)}(l,j,r) & \text{si } i \leq k_l \\ p^{(i)}(l,j,r) & \text{si } i = k_l+1 \end{cases}$ et $\Phi(n,k) = \sum_{l=1}^{m} \sum_{i=1}^{k_l} \beta_{li}$

Or d'après les expressions (6.7) et (6.9), la charge totale de la machine peut être reformulée sous la forme suivante :

$$\sum_{l=1}^{m} C_{max}^{l} = \sum_{l=1}^{m} \sum_{i=1}^{k_l+1} \sum_{r=1}^{u_i^l} \psi(l,i,r)p_{[r]} + \Phi(n,k) \tag{6.10}$$

avec $\psi(l,i,r) = \begin{cases} (1+w^{(i)}(l,r))g(l,i,r) & \text{si } i \leq k_l \\ g(l,i,r) & \text{si } i = k_l+1 \end{cases}$ et $\Phi(n,k) = \sum_{l=1}^{m} \sum_{i=1}^{k_l} \beta_{li}$.

La somme des dates de fin
D'après l'expression (5.16), nous avons :

$$\sum C_j = \sum_{l=1}^{m} \sum_{i=1}^{k_l+1} \sum_{r=1}^{u_i^l} V_{[l,i,r]}(l,i,r) + \Phi(n,k)$$

où $V_j(l,i,r) = \left(n_l - s_{ir}^l + 1 + \lambda_i^l w^{(i)}(l,r)\right) p^{(i)}(l,j,r)$ et $\Phi(n,k) = \sum_{l=1}^{m} \sum_{i=1}^{k_l} \lambda_i^l \beta_{li}$
avec $\lambda_i^l = \sum_{h=i+1}^{k_l+1} u_h^l$ et $\lambda_{k_l+1}^l = 0$, $\forall\, i,l$.

Or d'après les expressions (6.7) et (6.9), la somme des dates de fin peut être reformulée sous la forme suivante :

$$\sum C_j = \sum_{l=1}^{m} \sum_{i=1}^{k_l+1} \sum_{r=1}^{u_i^l} \psi(l,i,r)p_{[r]} + \Phi(n,k) \tag{6.11}$$

avec $\psi(l,i,r) = \left(n_l - s_{ir}^l + 1 + \lambda_i^l w^{(i)}(l,r)\right) g(l,i,r)$ et $\Phi(n,k) = \sum_{l=1}^{m} \sum_{i=1}^{k_l} \lambda_i^l \beta_{li}$
et $\lambda_i^l = \sum_{h=i+1}^{k_l+1} u_h^l$ et $\lambda_{k_l+1}^l = 0$

La somme des durées d'attente

D'après l'expression (5.17), (6.7) et (6.9) la somme des durées d'attente peut être reformulée sous la forme suivante :

$$\sum W_j = \sum_{l=1}^{m} \sum_{i=1}^{k_l+1} \sum_{r=1}^{u_i^l} \psi(l,i,r) p_{[r]} + \Phi(n,k) \tag{6.12}$$

avec $\psi(l,i,r) = \Big(n_l - s_{ir}^l + \lambda_i^l w^{(i)}(l,r)\Big)\, g(l,i,r)$, $\Phi(n,k) = \sum_{l=1}^{m} \sum_{i=1}^{k_l} \lambda_i^l \beta_{li}$ et $\lambda_i^l = \sum_{h=i+1}^{k_l+1} u_h^l$ et $\lambda_{k_l+1}^l = 0$, $\forall\, i,l$.

La variation des dates de fin

D'après l'expression (5.18), (6.7) et (6.9) la somme de la variation des dates de fin peut être reformulée sous la forme suivante :

$$TADC = \sum_{l=1}^{m} \sum_{i=1}^{k_l+1} \sum_{r=1}^{u_i^l} \psi(l,i,r) p_{[r]} + \Phi(n,k) \tag{6.13}$$

où $\psi(l,i,r) = \Big[(s_{ir}^l - 1)(n_l - s_{ir}^l + 1) + \lambda_i^l w^{(i)}(l,r)\Big]\, g(l,i,r)$ et $\Phi(n,k) = \sum_{l=1}^{m} \sum_{i=1}^{k_l} \lambda_i^l \beta_{li}$ avec $\lambda_i^l = (\sum_{h=1}^{i} u_h^l)(\sum_{h=i+1}^{k_l+1} u_h^l)$, avec $\lambda_{k_l+1}^l = 0$, $\forall\, i,l$.

La variation des durées d'attente

D'après l'expression (5.19), (6.7) et (6.9) la somme de la variation des durées d'attente peut être reformulée sous la forme suivante :

$$TADW = \sum_{l=1}^{m} \sum_{i=1}^{k_l+1} \sum_{r=1}^{u_i^l} \psi(l,i,r) p_{[r]} + \Phi(n,k) \tag{6.14}$$

où $\psi(l,i,r) = \Big[s_{ir}^l (n_l - s_{ir}^l) + \lambda_i^l w^{(i)}(l,r)\Big]\, g(l,i,r)$ et $\Phi(n,k) = \sum_{l=1}^{m} \sum_{i=1}^{k_l} \lambda_i^l \beta_{li}$ avec $\lambda_i^l = (\sum_{h=1}^{i} u_h^l)(\sum_{h=i+1}^{k_l+1} u_h^l)$, avec $\lambda_{k_l+1}^l = 0$, $\forall\, i,l$.

Dans le cas général, nous pouvons écrire le critère γ comme suit :

$$\gamma = \sum_{l=1}^{m} \sum_{i=1}^{k_l+1} \sum_{r=1}^{u_i^l} \psi(l,i,r) p_{[r]} + \Phi(n,k) \tag{6.15}$$

où $[r]$ est la tâche ordonnancée au rang r. Il est clair que le terme $\Phi(n,k)$ est constant une fois le nombre de maintenances k_l et les tailles des groupes u_i^l sur chaque machine fixés. Par conséquent, minimiser l'expression (6.15) est équivalent à minimiser $\sum_{l=1}^{m} \sum_{i=1}^{k_l+1} \sum_{r=1}^{u_i^l} \psi(l,i,r) p_{[r]}$.

Pour minimiser cette dernière, nous faisons appel au lemme de Hardy (cf. lemme 6.1), les termes $\psi(l,i,r)$ et p_j sont indépendants l'un de l'autre. Alors pour minimiser le produit scalaire de ces deux vecteurs, nous proposons un algorithme de tri (Algorithme 7).

Algorithme 7 : Problèmes d'ordonnancement avec fonctions décomposables

Données :
- $p^{(i)}(l,j,r) = p_j g(l,i,r)$ temps opératoire décomposable de la tâche T_j ordonnancée dans la position (l,i,r).
- $w^{(i)}(l,r)$ coefficient de détérioration de la maintenance due à l'ordonnancement de la tâche dans la position (l,i,r)
- β_{li} durée élémentaire de la maintenance i sur la machine M_l
- $\mathscr{N}$ ensemble de vecteurs $(n_1, n_2, \ldots, n_m)$ possibles tels que $n_1 + n_2 + \ldots + n_{k+1} = n$
- $\mathscr{K}$ ensemble de vecteurs $(k_1, k_2, \ldots, k_m)$ possibles tels que $k_1 + k_2 + \ldots + k_{k+1} = k$
- $\mathscr{U}$ ensemble de vecteurs $(u_1^l + u_2^l + \ldots + u_{k_l+1}^l)$ possibles, avec $(l = 1, \ldots, m)$, tels que $u_1^l + u_2^l + \ldots + u_{k_l+1}^l = n_l$

Résultat : γ^* la solution optimale

début

1 $P \longleftarrow$ Trier les p_j dans l'ordre décroissant

2 $\Psi \longleftarrow$ Trier les $\psi(l,i,r)$ dans l'ordre croissant

3 Affecter les tâches selon la règle suivante : affecter la tâche T_j avec la plus grande valeur p_j à la position qui possède la plus petite valeur $\psi(l,i,r)$. Ensuite, la tâche avec la deuxième plus grande valeur p_j est affectée à la position qui possède la deuxième petite valeur $\psi(l,i,r)$ et ainsi de suite. Soit V^* la solution obtenue

4 **retourner** $\gamma^* \longleftarrow V^* + \Phi(n,k)$

fin

Donc le problème peut être résolu avec une complexité $\mathcal{O}(n \log n)$ lorsque la taille de chaque groupe et le nombre de maintenances par machine sont connus d'avance. Cet algorithme de tri doit être exécuté $\mathcal{O}([max\{k,m\}]^{m-1} n^{m+k-1})$ fois afin de trouver la solution optimale (cf. section 5.3, page 103).

Théorème 6.7. *Les problèmes d'ordonnancement à machines parallèles*

$$R|p^{(i)}(l,j,r) = p_j g(l,i,r), \mathcal{MP}^{(i)} = k|\gamma$$

peuvent être résolus avec une complexité $\mathcal{O}([max\{k,m\}]^{m-1} n^{m+k} \log n)$*, où* n*,* m *et* k *sont, respectivement, le nombre de tâches, de machines et de maintenances.*

Remarque 6.4. *Si le nombre de machines* m *et le nombre de maintenances* k *sont constants, i.e. le problème* $Rm|p^{(i)}(l,j,r) = p_j g(l,i,r), \mathcal{MP}^{(i)} = k|\gamma$*, on obtient la complexité polynomiale* $\mathcal{O}(n^{m+k} \log n)$*. D'autre part, si* $m = 1$*, le problème* $1|p^{(i)}(j,r) = p_j g(i,r), \mathcal{MP}^{(i)} = k|\gamma$ *est polynomial avec une complexité* $\mathcal{O}(n^{k+1} \log n)$*.*

Remarque 6.5. *Soit k_0 le nombre de maintenances maximal à insérer sur l'ensemble des machines. Pour résoudre le problème $Rm|p^{(i)}(l,j,r) = p_j g(l,i,r), \mathcal{MP}^{(i)} \leq k_0|\gamma$, il suffit d'exécuter l'algorithme 7 pour $k = 1, \ldots, k_0$ et de renvoyer la meilleure solution. Ainsi, la complexité finale de cette résolution est : $\mathcal{O}(n^{m+k_0} \log n)$.*

Remarque 6.6. *À partir de l'analyse présentée ci-dessus, nous avons montré que lorsque le temps opératoire est décomposable et les coefficients des maintenances sont indépendants de la tâche ($w^{(i)}(l,r)$), nous pouvons remplacer le programme d'affectation linéaire de complexité $\mathcal{O}(n^3)$ par un simple algorithme de tri de complexité $\mathcal{O}(n \log n)$. Cette remarque peut être étendue aux théorèmes 6.1 et 6.4 prouvés précédemment dans lesquels un programme d'affectation linéaire a été utilisé. D'où les résultats suivants :*

- *Les problèmes d'ordonnancement à machines parallèles avec une maintenance optionnelle sur chaque machine :*

$$R|p^{(i)}(l,j,r) = p_j g(l,i,r), \mathcal{MP}_l^{(i)} \leq 1|\gamma$$

  peuvent être résolus avec une complexité $\mathcal{O}(2^m n^{2m} \log n)$. Notons que lorsque le nombre de machines $m \geq 1$ est fixe, le problème est polynomial.
- *Les problèmes d'ordonnancement à machines parallèles avec un système de maintenance fermé*

$$R|p_j g(l,r) - det, c - \mathcal{MP} = k| \sum_{l=1}^{m} C_{max}^l$$

  possèdent un ordonnancement équilibré optimal, et peuvent être résolus avec une complexité $\mathcal{O}([max\{k,m\}]^{m-1} n^m \log n)$, où n, m et k sont, respectivement, le nombre de tâches, de machines et de maintenances. Notons que lorsque le nombre de machines $m \geq 1$ et de maintenances sont fixes, le problème est polynomial.

Ce résultat permet de résoudre des problèmes non traités avant et aussi de généraliser d'autres travaux antérieurs.

Le théorème 6.7 généralise, par exemple, les travaux de Rustogi et Strusevich [98]. Les auteurs ont étudié un modèle réduit sur des machines parallèles uniformes (Qm) avec : $\gamma = \sum C_j$, $p(l,j,r) = p_j g(l,i,r)$, $w(l,j,r) = \alpha_i$ et $\beta_{li} = \beta_i$. Voir aussi les travaux de Yang et *al.* [123] et Yang et Yang [134].

6.6 Conclusion

Dans ce chapitre, nous nous sommes intéressés à des extensions et des cas particuliers des problèmes d'ordonnancement, étudiés dans les chapitres 4 et 5, en présence de périodes d'indisponibilité dans les ateliers à une machine et à machines parallèles.

Plusieurs extensions et cas particuliers ont été étudiés. Une méthode de résolution a été proposée quand la politique de maintenance optionnelle est considérée, *i.e.* au plus une période de maintenance à insérer sur chaque machine. Ensuite, nous avons amélioré la résolution du problème lorsque la minimisation de la charge totale des machines est considérée. Le problème peut être résolu avec une complexité inférieure si les périodes de maintenances ne comportent pas des durées élémentaires $\beta_{li} = 0$.

De plus, nous avons montré que lorsque les temps opératoires suivent une fonction décomposable, un algorithme de tri peut être utilisé pour résoudre le problème.

Dans le tableau 6.6, nous reprenons l'essentiel des résultats obtenus dans ce chapitre. Notons que ces résultats permettent de résoudre des problèmes non traités avant, de généraliser et d'améliorer des travaux antérieurs.

Tableau 6.6 – Tableau récapitulatif des résultats

Problèmes	**Approche globale**		
	Complexités	Complexités m et k fixes	Réf.
$R\|p^{(i)}(l,j,r), \mathcal{MP}_l^{(i)} \leq 1\|\gamma$	$\mathcal{O}(2^m n^{2m+2})$	$\mathcal{O}(n^{2m+2})$	Théo. 6.1
$R\|p^{(i)}(l,j,r), \mathcal{MP}_l^{(i)} \leq 1\|\gamma_{cost}$	$\mathcal{O}(2^m n^{2m+2})$	$\mathcal{O}(n^{2m+2})$	Rem. 6.1
$1\|p^{(i)}(j,r), \mathcal{MP}^{(i)} \leq 1\|\gamma, \gamma_{cost}$	$\mathcal{O}(n^4)$	$\mathcal{O}(n^4)$	Théo. 6.1
$R\|p^{(i)}(l,j,r) - det, \mathcal{MP}_l^{(i)} \leq 1\| \sum_{l=1}^{m} C_{max}^l$	$\mathcal{O}(2^m(n^3 + 2n^2 m))$	$\mathcal{O}(n^3)$	Théo. 6.2
$1\|p^{(i)}(j,r) - det, \mathcal{MP}^{(i)} \leq 1\|C_{max}$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	Théo. 6.3
$R\|p(l,j,r) - det, c - \mathcal{MP} = k\| \sum_{l=1}^{m} C_{max}^l$	$\mathcal{O}([max\{k,m\}]^{m-1} n^{m+2})$	$\mathcal{O}(n^{m+2})$	Théo. 6.4
$R\|p(l,j,r) - det, c - \mathcal{MP} = k\| \sum_{l=1}^{m} C_{max}^l$	$\mathcal{O}([max\{k,m\}]^{m-1}(n^3 + n^2 m))$	$\mathcal{O}(n^3)$	Théo. 6.5
$R\|p(l,j,r) - det, \mathcal{MP}, \beta_{li} = 0\| \sum_{l=1}^{m} C_{max}^l$	$O(mn^3)$	$\mathcal{O}(n^3)$	Théo. 6.6
$1\|p(j,r) - det, \mathcal{MP}, \beta_i = 0\|C_{max}$	$O(n^3)$	$O(n^3)$	Théo. 6.6
$R\|p^{(i)}(l,j,r) = p_j g(l,i,r), \mathcal{MP}^{(i)} = k\|\gamma$	$\mathcal{O}([max\{k,m\}]^{m-1} n^{m+k} \log n)$	$\mathcal{O}(n^{m+k} \log n)$	Théo. 6.7
$R\|p^{(i)}(l,j,r) = p_j g(l,i,r), \mathcal{MP}^{(i)} \leq k_0\|\gamma$	$\mathcal{O}([max\{k_0,m\}]^{m-1} n^{m+k_0} \log n)$	$\mathcal{O}(n^{m+k_0} \log n)$	Rem. 6.5
$R\|p^{(i)}(l,j,r) = p_j g(l,i,r), \mathcal{MP}_l^{(i)} \leq 1\|\gamma$	$\mathcal{O}(2^m n^{2m} \log n)$	$\mathcal{O}(n^{2m} \log n)$	Rem. 6.6
$1\|p^{(i)}(j,r) = p_j g(i,r), \mathcal{MP}^{(i)} \leq 1\|\gamma$	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(n^2 \log n)$	Rem. 6.6
$R\|p_j g(l,r) - det, c - \mathcal{MP} = k\| \sum_{l=1}^{m} C_{max}^l$	$\mathcal{O}([max\{k,m\}]^{m-1} n^m \log n)$	$\mathcal{O}(n^m \log n)$	Rem. 6.6
$1\|p_j g(r) - det, c - \mathcal{MP} = k\|C_{max}$	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log n)$	Rem. 6.6
$1\|p^{(i)}(j,r) = p_j g(i,r), \mathcal{MP}^{(i)} = k\|\gamma$	$\mathcal{O}(n^{k+1} \log n)$	$\mathcal{O}(n^{k+1} \log n)$	Rem. 6.4

Dans le chapitre suivant, nous allons étudier des problèmes d'ordonnancement en présence de périodes d'indisponibilité dans un atelier flow shop à deux machines.

Troisième partie

Problème flow shop avec une période d'indisponibilité

7

Problème flow shop avec une période d'indisponibilité

Résumé : Dans ce chapitre, nous étudions le problème d'ordonnancement flow shop à deux machines en présence d'une période de maintenance de durée variable sur la deuxième machine. Notre objectif est la minimisation de la date de fin de l'ordonnancement. Nous allons montrer que lorsque la maintenance doit être placée dans l'intervalle $[0, T]$, le problème est polynomial. Cependant, le problème est NP-difficile si la période d'indisponibilité est à insérer à partir de l'instant $T > 0$. Pour résoudre ce problème, nous proposons deux heuristiques. Ensuite, nous développons trois méthodes exactes, à savoir une méthode énumérative, un programme linéaire et une approche de séparation et évaluation, conduisant à une solution optimale du problème. Ce chapitre est enfin clôturé par une étude expérimentale des différentes méthodes présentées.

Sommaire

Les résultats développés dans ce chapitre ont été présentés dans les articles suivants :

[31] A. Gara-Ali et M.-L. Espinouse. *Erratum to :"Simultaneously scheduling n jobs and the preventive maintenance on the two-machine flow shop to minimize the makespan".* International Journal of Production Economics **153**, 361 – 363 (2014).

[33] A. Gara-Ali et M.-L. Espinouse. *A two-machine flow-shop scheduling with a deteriorating maintenance activity on the second machine.* In *Industrial Engineering and Systems Management (IESM), 2015 International Conference*, 481– 488, Séville, Espagne (2015).

[32] A. Gara-Ali et M.-L. Espinouse. *Méthodes exactes pour la résolution du problème Flow-shop à deux machines avec des contraintes de disponibilité sur la deuxième machine.* In *ROADEF - 15ème congrès annuel de la Société française de recherche opérationnelle et d'aide à la décision*, Bordeaux, France (2014).

7.1 Introduction

Dans ce chapitre, nous allons étudier un problème d'ordonnancement flow shop à deux machines, en présence d'une période d'indisponibilité sur la deuxième machine. Nous montrons que la complexité du problème dépend de la position de la maintenance. En effet, le problème est polynomial si la position de l'indisponibilité est une variable de décision et doit être placée dans l'intervalle $[0, T]$.

Lorsque la maintenance doit être placée à partir de l'instant $T \geq 0$, nous montrons que ce problème est NP-difficile. Ainsi, nous proposons, en premier lieu, deux heuristiques. Puis, nous présentons un programme linéaire et une méthode énumérative pour résoudre le problème d'une manière exacte. L'utilisation de ces deux méthodes exactes est limitée à des instances de petite taille. Pour cela, une méthode basée sur l'approche de séparation et évaluation (Branch-and-Bound $B\&B$) a été conçue, ce qui nous permettra d'avoir des solutions optimales pour des problèmes de grande taille.

Enfin, nous terminons ce chapitre par une étude expérimentale des différentes méthodes présentées.

7.2 Positionnement du problème

Les problèmes d'ordonnancement de type flow shop en présence de périodes d'indisponibilité ont été étudiés depuis les années 90. Dans ce chapitre, nous nous concentrons sur l'ordonnancement d'une seule période de maintenance. Nous citons les premiers travaux de Lee en 1997 [73] qui considèrent une période d'indisponibilité fixe, *i.e.* une période d'indisponibilité qui démarre à l'instant T connu d'avance. L'auteur a étudié deux configurations ; un cas sécable : si une tâche ne peut pas se terminer avant la période d'indisponibilité alors la tâche peut se poursuivre une fois que la machine est à nouveau disponible. Dans le cas non sécable, la tâche doit redémarrer une fois la machine à nouveau disponible. Lee a démontré que lorsque la période d'indisponibilité est sur la première ou la deuxième machine, le problème est NP-difficile, voir tableau 7.1 [1] .

Ensuite, plusieurs travaux ont été menés notamment sur ce problème en ajoutant d'autres contraintes telles que :

- Des contraintes sur la position de la maintenance : en effet, la position peut être connue d'avance ou une variable de décision, dans ce cas on parle d'un ordonnancement simultané des tâches et des maintenances.
- Des contraintes sur la durée de la maintenance : la durée peut être constante, ce

1. Nous notons par $r - M_l$ une période d'indisponibilité sécable sur la machine M_l et par $nr - M_l$ une période d'indisponibilité non sécable sur la machine M_l.

Tableau 7.1 – Problèmes étudiés dans les travaux de Lee [73]

Problème	Complexité	Solution
$F2\|r-M_1\|C_{max}$	$NP-difficile$	Programmation dynamique & Heuristiques
$F2\|r-M_2\|C_{max}$	$NP-difficile$	Programmation dynamique & Heuristiques
$F2\|nr-M_1\|C_{max}$	$NP-difficile$	Programmation dynamique & Heuristiques
$F2\|nr-M_2\|C_{max}$	$NP-difficile$	Heuristiques

modèle reflète le fait que très souvent la maintenance est constituée d'un nombre fixe d'actions de maintenances de routines à effectuer. En 2005, Kubzin et Strusevich [67, 68] ont introduit un nouveau modèle de maintenance de durée variable dont la durée varie linéairement en fonction de sa date de début.

Nous reprenons dans le tableau 7.2, les différents travaux menés sur les problèmes d'ordonnancement de type flow shop à deux machines avec une période de maintenance sur la première ou la deuxième machine, ainsi que les contraintes considérées. Notons que dans ces travaux la période de maintenance n'a aucun impact sur les durées des tâches, *i.e.* les durées des tâches sont identiques avant et après la maintenance.

Tableau 7.2 – Résultats des problèmes flow shop à deux machines avec une période de maintenance

Référence	Position de la maintenance					Durée	Complexité	Solution
	sur M_1	sur M_2	entre [0,T]	à T	après T			
Lee [73]	✓			✓		Constante	NP-difficile	Prog. dynamique et Heuristique
Lee [73]		✓		✓		Constante	NP-difficile	Heuristique
Hnaien et *al.* [53]	✓			✓		Constante	NP-difficile	MIP et Branch-and-Bound
Allaoui et *al.* [3]	✓		✓			Constante	NP-difficile	Heuristique
Allaoui et *al.* [3]		✓	✓			Constante	NP-difficile	Heuristique
Kubzin et *al.* [68]	✓	✓	✓			Variable*	NP-difficile	Algorithme pseudo-polynomial

MIP : Mixed Integer Programming. * : Modèle linéaire : $\alpha t+\beta$

7.3 Présentation du problème et notations

En nous basant sur les problèmes présentés dans le tableau 7.2, pour positionner notre étude, nous définissons un problème avec une période de maintenance de durée variable sur la deuxième machine dont la position est une variable de décision (la date de début n'est pas connue d'avance). L'objectif est la minimisation de la durée totale de l'ordonnancement (Makespan).

Nous considérons un ensemble de n tâches $N = \{T_1, T_2, \ldots, T_n\}$ à ordonnancer dans un atelier flow shop à deux machines. Les tâches sont disponibles à l'instant zéro et la préemption n'est pas autorisée.

L'insertion de la période de maintenance permet de modifier la performance de la machine (RMA Rate-Modifying Activity). Soit b_{lj} la durée de la tâche T_j ordonnancée avant la période de maintenance sur la machine M_l avec $l = \{1, 2\}$ et a_{lj} sa durée si elle est insérée après la période de maintenance sur la machine M_l. Après la maintenance, les performances de la machine sont améliorées, nous supposons donc que $a_{lj} \leq b_{lj}$. On note B et A l'ensemble des tâches qui se trouvent, respectivement, avant et après la maintenance, voir figure 7.1.

Figure 7.1 – Période de maintenance sur M_2

La durée de la maintenance $m(t)$ suit une fonction croissante selon la date de début t :

$$m(t) = \alpha \times t + \beta$$

avec $\alpha \geq 0$ et $\beta > 0$, respectivement, le coefficient de détérioration et la durée élémentaire de la maintenance.

Comme précisé précédemment, dans ce chapitre, nous considérons que la position de la maintenance est une variable de décision. Nous allons étudier deux contraintes sur la position de la maintenance :

- La maintenance doit être commencée dans l'intervalle $[0, T]$.
- La machine doit être maintenue une seule fois après un instant donné $T > 0$, *i.e.* l'équipe de maintenance ne peut pas entretenir la machine entre $[0, T]$ à cause de l'indisponibilité de l'équipe maintenance ou de pièces de rechange.

Dans cette étude, nous nous intéressons à l'insertion de la période de maintenance sur la deuxième machine. Notons que le problème avec une indisponibilité variable de décision sur la première machine a été étudié dans les travaux de Kubzin et Strusevich [68]. Les auteurs considèrent que la période d'indisponibilité n'a pas d'impact sur les durées des tâches (durées des tâches constantes) et qu'aucune contrainte n'est appliquée sur son placement. Les auteurs ont montré que le problème est NP-difficile.

En utilisant la notation à trois champs de Graham [45], notre problème d'ordonnancement est noté par $F2|(b_{lj}, a_{lj}), nr-\mathcal{M}_2 = 1, m = \alpha t+\beta|C_{max}$. Notons que « $nr-\mathcal{M}_2 = 1$ »,

indique qu'une seule période de maintenance est à insérer sur la deuxième machine et que les tâches sont non sécables. La durée de la période de maintenance suit une fonction linéaire $m = \alpha t + \beta$.

Dans ce qui suit, nous considérons les notations présentées ci-dessous :

$b_{l,j}$	temps opératoire de la tâche T_j avant la maintenance sur la machine M_l
$a_{l,j}$	temps opératoire de la tâche T_j après la maintenance sur la machine M_l
α	coefficient de la maintenance
β	durée élémentaire de la maintenance
t	date d'insertion de la maintenance
$C_{l,j}$	date d'achèvement de la tâche T_j sur la machine M_l
$C_{l,m}$	date d'achèvement de la maintenance sur la machine M_l
C_{l,π_1}	date d'achèvement de la séquence π_1 sur la machine M_l

Algorithme de Johnson : Avant d'entamer l'analyse du problème flow shop, présenté ci-dessus, il est nécessaire de présenter l'algorithme de Johnson [63], qui permet de résoudre d'une façon optimale le problème flow shop à deux machines sans période d'indisponibilité, *i.e.* les machines sont toujours disponibles, avec comme objectif la minimisation du C_{max}. Johnson [63] a démontré que pour minimiser le makespan, il suffit d'ordonnancer la tâche T_i avant T_j si :

$$\min\{b_{1,i}; b_{2,j}\} < \min\{b_{1,j}; b_{2,i}\}$$

Cette propriété de base permet de développer l'algorithme suivant, voir algorithme 8.

Algorithme 8 : Algorithme de Johnson

1 Créer deux groupes de tâches :
  - le groupe G_1 contient les tâches telles que $b_{1,j} \leq b_{2,j}$;
  - le groupe G_2 contient les tâches telles que $b_{1,j} > b_{2,j}$;
2 Trier le groupe G_1 par $b_{1,j}$ croissants ;
3 Trier le groupe G_2 par $b_{2,j}$ décroissants ;
4 La séquence de Johnson est obtenue par concaténation de G_1 puis G_2;

7.4 Période d'indisponibilité entre $[0, T]$

Dans cette section, nous nous intéressons au problème $F2|(b_{lj}, a_{lj}), nr - \mathcal{M}_2 = 1, m = \alpha t + \beta|C_{max}$, dont la maintenance doit être démarrée avant l'instant T sur la deuxième

machine M_2, *i.e.* $t \leq T$.

Vu que la durée de la maintenance augmente avec sa date de début t et que les temps opératoires diminuent après l'indisponibilité $b_{lj} \geq a_{lj}$, il suffit d'insérer la maintenance à l'instant zéro (pour minimiser la durée de la maintenance, durée égale à β)

Théorème 7.1. *Le problème* $F2|(b_{lj}, a_{lj}), nr - \mathcal{M}_2 = 1, m = \alpha t + \beta|C_{max}$ *est résolu polynomialement en démarrant la maintenance à l'instant zéro et ordonnançant les tâches après, selon l'algorithme de Johnson.*

Démonstration. Pour prouver ce théorème, nous allons utiliser un argument d'échange. Soit S un ordonnancement optimal, dans lequel il existe une tâche T_j ordonnancée avant la maintenance sur la machine M_2. Une représentation de l'ordonnancement S est donnée dans la figure 7.2.

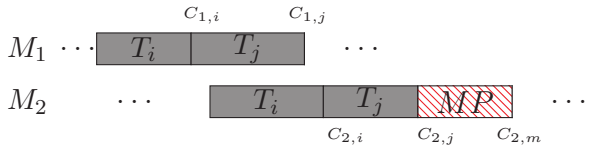

Figure 7.2 – Ordonnancement S : tâche T_j avant la maintenance

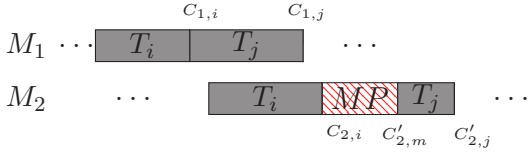

Figure 7.3 – Ordonnancement S' : tâche T_j après la maintenance

En échangeant la tâche T_j et la maintenance, on obtient l'ordonnancement S' représenté dans la figure 7.3.

Soit $C_{2,i}$ la date de fin d'exécution de la tâche T_i sur M_2, m et m' les durées de la période de maintenance, respectivement, dans les ordonnancements S et S'. Il est clair que $m' < m$, car la durée de la maintenance suit une fonction linéaire croissante selon t.

- Dans l'ordonnancement S, nous avons :
  $C_{2,j} = \max\{C_{1,j}; C_{2,i}\} + b_{2,j}$
  $C_{2,m} = C_{2,j} + m$ avec $m = \alpha C_{2,j} + \beta$
  $\Rightarrow C_{2,m} = \max\{C_{1,j}; C_{2,i}\} + b_{2,j} + m$
- Dans l'ordonnancement S', nous avons :
  $C'_{2,m} = C_{2,i} + m'$ avec $m' = \alpha C_{2,i} + \beta$
  $C'_{2,j} = \max\{C_{1,j}; C'_{2,m}\} + a_{2,j}$
  $\Rightarrow C'_{2,j} = \max\{C_{1,j}; C_{2,i} + m'\} + a_{2,j}$

Or, il est clair que : $\max\{C_{1,j}; C_{2,i}\} + m \geq \max\{C_{1,j}; C_{2,i} + m'\}$, car $m' < m$. De plus, $b_{2,j} \geq a_{2,j}$. Ainsi, on obtient $C'_{2,j} \leq C_{2,m}$. On conclut alors que dans l'ordonnancement optimal la maintenance doit être insérée avant toutes les tâches. Plus précisément, la maintenance doit être insérée à l'instant zéro, puisque sa durée est minimisée à $t = 0$. Le problème est donc équivalent à minimiser le C_{max} dans un problème flow shop à deux machines tel que la machine M_2 est indisponible entre $[0, \beta]$. Il est trivial qu'après

l'indisponibilité de la machine M_2 les tâches doivent être ordonnancées selon l'algorithme de Johnson, afin de minimiser C_{max}. □

Notons que l'insertion d'une période de maintenance à l'instant zéro peut être légitime. En effet, nous pouvons considérer que la machine a été utilisée avant le début de l'ordonnancement. Nous étudions dans le paragraphe suivant le cas où la période de maintenance n'est pas autorisée à l'instant zéro.

Maintenance non autorisée en début d'ordonnancement

Nous supposons, maintenant, que la période d'indisponibilité n'est pas autorisée à l'instant zéro. En effet, la machine est supposée dans un bon état, il est donc interdit d'insérer une maintenance tant qu'aucune tâche n'est traitée sur la machine. Nous montrons, dans ce qui suit, que le problème reste polynomial.

Proposition 7.1. *Il existe une séquence optimale telle qu'une seule tâche est ordonnancée avant la maintenance et les tâches qui suivent la période de maintenance doivent être ordonnancées selon Johnson.*

Démonstration. La propriété peut être prouvée en échangeant la dernière tâche avant la maintenance avec la période maintenance, un tel échange ne pourra jamais augmenter le makespan, voir figures 7.2 et 7.3. La preuve est semblable à la preuve précédente du théorème 7.1, par contre dans ce cas, une tâche doit être exécutée avant la maintenance puisque cette dernière n'est pas autorisée en début de l'ordonnancement. □

La question est maintenant de déterminer la tâche à ordonnancer avant la maintenance. Le reste des tâches est ordonnancé après l'indisponibilité selon Johnson.

En se basant sur l'analyse ci-dessus, nous proposons l'algorithme 9 pour résoudre le problème. L'idée de cet algorithme consiste à ordonnancer, à chaque fois, une tâche T_j $(j = 1, \ldots, n)$ avant la maintenance. Notons que cette tâche doit être achevée avant l'instant T, *i.e.* $b_{1,j}+b_{2,j} \leq T$ (étape 3.1, algorithme 9). Le reste des tâches est ordonnancé après l'indisponibilité selon la règle de Johnson.

Théorème 7.2. *Lorsque la maintenance n'est pas autorisée en début de l'ordonnancement et doit être démarrée au plus tard à l'instant T, le problème :*
$F2|(b_{lj}, a_{lj}), nr - \mathcal{M}_2 = 1, m = \alpha t + \beta|C_{max}$ est résolu dans un temps polynomial $\mathcal{O}(n^2)$ en utilisant l'algorithme 9.

Démonstration. L'étape 3.1.1 peut être résolue dans un temps $\mathcal{O}(n)$ pour chaque tâche T_j. Les étapes 1 et 2 sont exécutées une seule fois. Par conséquent, le temps d'exécution global de l'algorithme 9 est $\mathcal{O}(n^2)$. □

Algorithme 9 : Problèmes flow shop avec une période de maintenance sur M_2

Données :
- $N \in \{T_1, T_2, \ldots, T_n\}$ un ensemble des n tâches à ordonnancer sur deux machines flow shop
- $b_{l,j}$ ($a_{l,j}$) temps opératoires de la tâche T_j ordonnancée avant (après) la maintenance sur la machine M_l, ($j = 1, \ldots, n$ et $l = 1, 2$)
- T date au plus tard pour démarrer la maintenance
- α coefficient de la détérioration de la maintenance
- β durée élémentaire de la maintenance

Résultat : C^A_{max} le makespan obtenu par l'algorithme

1 $C^A_{max} \longleftarrow +\infty$ Initialiser le makespan.
2 Trier les tâches selon l'algorithme de Johnson.
3 **pour chaque** $T_j \in N$ **faire**
3.1 **si** $b_{1,j} + b_{2,j} \leq T$ **alors**
3.1.1 Construire une séquence avec la tâche T_j placée avant la maintenance et le reste des tâches après la période de maintenance, soit $C_{max}(T_j)$ le makespan obtenu par cette séquence.
3.1.2 **si** $C_{max}(T_j) \leq C^A_{max}$ **alors**
3.1.3 $C^A_{max} \longleftarrow C_{max}(T_j)$
fin si
fin si
fin pour
4 **retourner** C^A_{max}

7.4.1 Maintenance avec durée constante : Erratum

Dans [3], Allaoui et *al.* ont étudié une version réduite du problème d'ordonnancement précédent, mais avec une maintenance de durée constante ($\alpha = 0$) et sans modifier les performances de la machine ($a_{l,j} = b_{l,j}$) : $F2|nr - M_2, m = \beta|C_{max}$. Les auteurs ont considéré le problème suivant ; un ensemble de n tâches à ordonnancer dans un atelier flow shop à deux machines, en présence d'une période de maintenance de durée constante β et à placer avant l'instant T sur la deuxième machine M_2. Les auteurs considèrent que l'insertion de la maintenance n'a pas d'influence sur les performances de la machine, *i.e.* $b_{2j} = a_{2j}$, $\forall j = 1, \ldots, n$.

Contrairement au théorème 7.1, les auteurs ont prouvé par une réduction à partir du problème 2-Partition que le problème est NP-difficile au sens faible (Théorème 7 –Allaoui et *al.* p.166 [3]). Dans ce qui suit, nous allons montrer que la preuve de complexité, présentée dans [3], est fausse et que le problème est polynomial.

Dans [3], le problème de partition suivant est considéré : Soit un ensemble de n entiers $e_1, e_2, \ldots, e_n$ avec $\sum_{i=1}^{n} e_i = 2A$. Existe-t-il une partition en deux ensembles S_1 et S_2 tels que la somme des éléments de chaque ensemble est égale à A ?

Les auteurs définissent l'instance suivante : $b_{1,i} = e_i$, $b_{2,i} = e_i(A+1)$, avec $i = 1, \ldots, n$, et $b_{1,n+1} = 0$, $b_{2,n+1} = e_{max}$, avec $e_{max} = \max\{e_i/i = i \ldots, n\}$. Pour la maintenance, on a $T = A^2 + A + e_{max}$ et $\beta = A^2$. Il est demandé s'il existe un ordonnancement S tel que $C_{max}(S) = 3A^2 + 2A + e_{max}$.

Comme indiqué dans [3], si la partition a une réponse positive donc l'ordonnancement existe et il a une réponse positive. Cependant, nous pouvons remarquer que si l'ordonnancement existe (réponse positive), la partition peut avoir une réponse négative. Ainsi, la preuve de complexité proposée dans [3] est fausse.

Nous présentons l'exemple suivant pour montrer que la preuve est incorrecte :

Pour le problème de partition nous considérons : $n = 3$, $e_1 = 1$, $e_2 = 1$ et $e_3 = 8$ ainsi $\sum_{i=1}^{3} e_i = 10$, donc $A = 5$. L'ordonnancement correspondant à cette instance est constitué de quatre tâches avec

- Tâche 1 : $b_{1,1} = 1$, $b_{2,1} = 6$
- Tâche 2 : $b_{1,2} = 1$, $b_{2,2} = 6$
- Tâche 3 : $b_{1,3} = 8$, $b_{2,3} = 48$
- Tâche 4 : $b_{1,4} = 0$, $b_{2,4} = e_{max} = 8$

Pour la période de maintenance, $T = 38$ et $\beta = A^2 = 25$.

Nous considérons la séquence suivante $\{T_4, T_1, T_2, T_3\}$ présentée dans la figure suivante 7.4.

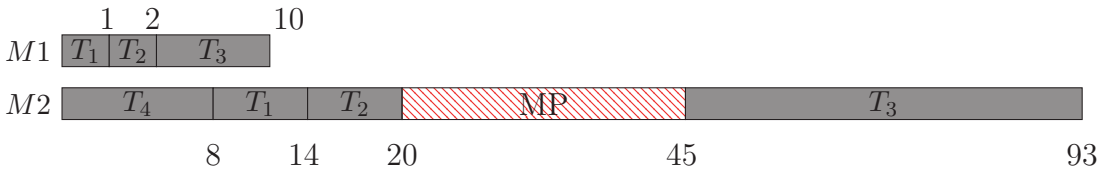

Figure 7.4 – Contre exemple

Avec cette solution présentée dans la figure 7.4, le makespan est $C_{max} = 3A^2 + 2A + e_{max} = 93$ et $S_1 = \{e_1, e_2\} \neq A$, $S_2 = \{e_3\} \neq A$. Ainsi, la preuve proposée dans [3] est fausse.

Le problème étudié dans l'article [3] est un cas particulier du problème d'ordonnancement $F2|(b_{lj}, a_{lj}), nr - \mathcal{M}_2 = 1, m = \alpha t + \beta|C_{max}$ et peut être résolu polynomialement en utilisant le théorème 7.1 si la maintenance est autorisée à l'instant zéro, et en utilisant le théorème 7.2 si la maintenance n'est pas autorisée en début de l'ordonnancement. Ces résultats ont été publiés dans la revue *« IJPE– International Journal of Prodction Economics »* [31] sous forme d'un erratum à l'article [3].

7.5 Période d'indisponibilité après l'instant T

Dans l'industrie, la maintenance peut être non autorisée sur une machine avant un instant T donné, par exemple à cause de l'indisponibilité de l'équipe maintenance ou de pièces de rechange. Dans cette section, nous considérons le problème d'ordonnancement $F2|(b_{lj}, a_{lj}), nr - \mathcal{M}_2 = 1, m = \alpha t + \beta|C_{max}$, avec une période de maintenance de durée variable ($m = \alpha t + \beta$) à ordonnancer sur M_2 après un instant $T > 0$ donné, *i.e.* $t \geq T$.

Nous commençons notre analyse du problème par une étude de complexité. Ensuite, nous allons déterminer les propriétés d'optimalité de l'ordonnancement optimal. En se basant sur ces propriétés, des méthodes de résolution approchées et exactes ont été développées pour résoudre le problème.

Soit $C^*_{max}(\emptyset)$ le makespan obtenu par l'algorithme de Johnson, lorsque la période de maintenance n'est pas insérée. Pour assurer la faisabilité du problème étudié, on considère que $C^*_{max}(\emptyset)$ est supérieure à T. En effet, si $C^*_{max}(\emptyset) \leq T$ la solution optimale est obtenue par l'ordonnancement de tâches selon l'algorithme de Johnson et aucune période de maintenance n'est insérée dans l'ordonnancement. Dans le reste de la section, nous considérons que :

$$C^*_{max}(\emptyset) \geq T \tag{7.1}$$

7.5.1 Étude de complexité

Dans cette section, nous étudions la complexité du problème d'ordonnancement étudié $F2|(b_{lj}, a_{lj}), nr - \mathcal{M}_2, m = \alpha t + \beta|C_{max}$ avec une maintenance de durée variable à insérer après l'instant T.

En utilisant la réduction à partir du problème 2-Partition, nous pouvons montrer que le problème est NP-difficile.

Théorème 7.3. *Le problème $F2|(b_{lj}, a_{lj}), nr - \mathcal{M}_2, m = \alpha t + \beta|C_{max}$, avec $t \geq T$ est NP-difficile.*

Démonstration. Le problème de partition est défini comme suit : soit un ensemble de r entiers $e_1, e_2, \ldots, e_r$, avec $\sum_{j=1}^{r} e_j = 2E$. Existe-t-il une partition qui permet de diviser cet ensemble en deux sous-ensembles disjoints R_1 et R_2, avec $\sum_{j \in R_1} e_j = \sum_{j \in R_2} e_j = E$?

Nous définissons l'instance I du problème avec :

- $n = r + 1$
- $b_{1,j} = e_j$, $b_{2,j} = 2e_j$ et $a_{2,j} = 2e_j$, avec $j = 1, \ldots, r$
- $b_{1,r+1} = 0$, $b_{2,r+1} = E$ et $a_{2,r+1} = E$
- $T = 3E$, $\alpha = 1$ et $\beta = E$

Existe-t-il un ordonnancement S tel que $C_{max}(S) = 9E$?

$\longrightarrow$ Nous supposons qu'il existe une partition R_1 et R_2, et nous construisons l'ordonnancement comme suit :
$S = \{T_{r+1}, R_1, R_2\}$. L'ordonnancement est présenté dans le diagramme de Gantt ci-dessous (figure 7.5) avec $C_{max}(S) = 9E$.

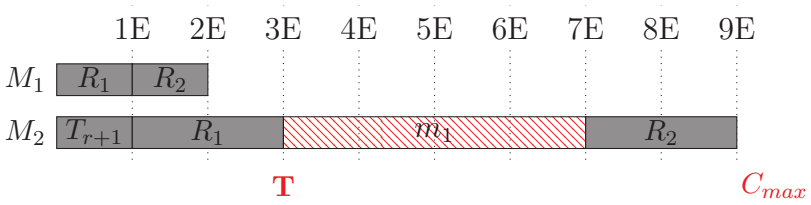

Figure 7.5 – Ordonnancement de la partition

$\longleftarrow$ Nous supposons qu'il existe un ordonnancement S avec un $C_{max}(S) = 9E$. On sait que la durée totale d'exécution des tâches sur M_2 est $\sum_{j=1}^{n} b_{2,j} = \sum_{j=1}^{n} a_{2,j} = 4E+E = 5E$. La durée minimum de la maintenance est obtenue lorsque la période de maintenance démarre à $T = 3E$, avec une durée égale à $4E$. Ainsi, aucun temps mort n'est créé entre $[0, C_{max}(S)]$ sur la deuxième machine M_2 et la maintenance doit être lancée à l'instant T. Alors, la tâche T_{r+1} doit être ordonnancée en première position. Ensuite, un ensemble des tâches A_1 doit être ordonnancé avant la période de maintenance entre les instants E et $3E$, avec $\sum_{j \in A_1} a_{2,j} = 2E$.

Le reste des tâches A_2 doit être ordonnancé après la période de maintenance avec un temps opératoire total égal à $9E - 7E = 2E$. Ainsi A_1 et A_2 forment une solution pour le problème de partition, d'où le résultat. □

7.5.2 Propriétés d'optimalité

Avant de proposer des méthodes de résolution, il est primordial de déterminer les propriétés d'optimalité qui caractérisent ce problème d'ordonnancement. Nous nous intéresserons dans cette partie aux différentes propriétés d'optimalité.

Propriété 1. *Il existe un ordonnancement optimal tel que les tâches avant (ensemble B) et après (ensemble A) la période de maintenance sont séquencées selon l'algorithme de Johnson.*

Démonstration. Cette propriété peut être prouvée facilement par un argument d'échange. Soit S un ordonnancement dans lequel il existe deux tâches consécutives T_i et T_j dans l'ensemble B qui ne vérifient pas la règle de Johnson [63], voir figure 7.6. En échangeant les tâches T_i et T_j, on obtient un ordonnancement S'. Un tel échange est toujours réalisable et ne pourra jamais augmenter le C_{max}, voir figure 7.7.

En utilisant le même principe, nous pouvons montrer que les tâches dans l'ensemble A suivent aussi l'algorithme de Johnson.

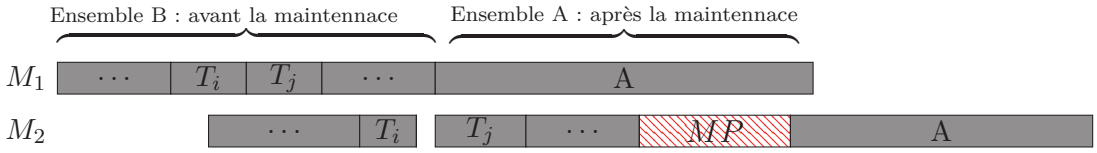

Figure 7.6 – Ordonnancement S : Tâches T_i et T_j ne vérifient pas Johnson

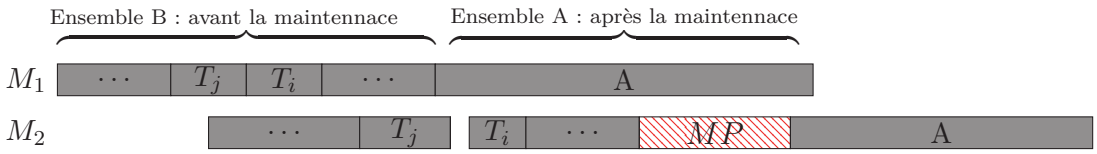

Figure 7.7 – Ordonnancement S' : Tâches T_j et T_i vérifient Johnson

□

Propriété 2. *Dans l'ordonnancement optimal, la maintenance doit être ordonnancée le plus tôt possible sur M_2 avec $t \geq T$, i.e. la dernière tâche qui précède la maintenance doit être lancée avant l'instant T.*

Nous distinguons deux cas possibles pour la position de la période de maintenance. Soit T_j la tâche qui précède la période d'indisponibilité, cette dernière peut être placée :

- à l'instant T si la machine n'est pas occupée, voir figure 7.8.
- après l'instant T, voir figure 7.9.

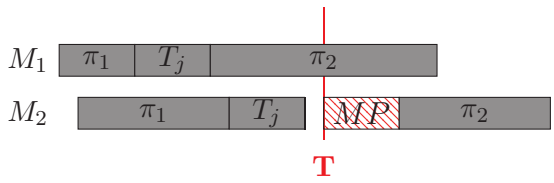

Figure 7.8 – Cas 1 : Maintenance à l'instant T

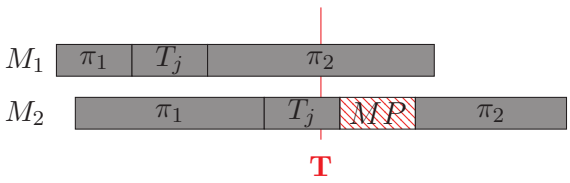

Figure 7.9 – Cas 2 : Maintenance après l'instant T

Démonstration. On considère un ordonnancement S, dont la maintenance n'est pas ordonnancée le plus tôt possible, *i.e.* il existe une tâche T_j qui précède la maintenance et démarre après l'instant T, voir figure 7.10, où π_1 et π_2 sont des ordonnancements partiels. Nous avons donc :

$$\max\{C_{1,j}; C_{2,\pi_1}\} \geq T \tag{7.2}$$

avec $C_{1,j}$ et C_{2,π_1}, respectivement, les dates de fin de la tâche T_j sur M_1 et de l'ordonnancement partiel π_1 sur M_2.

Soit S' le même ordonnancement avec le déplacement de la tâche T_j dans la position qui suit la période de maintenance, tout en conservant les tâches restantes dans leurs positions, voir figure 7.11. Notons que si un temps mort apparaît avant la maintenance, la maintenance doit être décalée à l'instant $\max\{T; C_{2,\pi_1}\}$. Soit $C_{max}(S)$ et $C_{max}(S')$, respectivement, les dates de fin pour les deux ordonnancements S et S'.

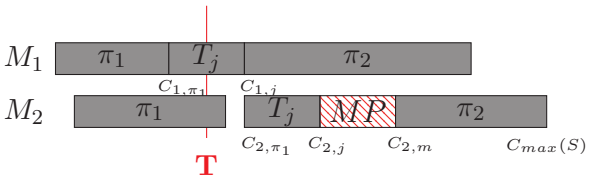

Figure 7.10 – Ordonnancement S

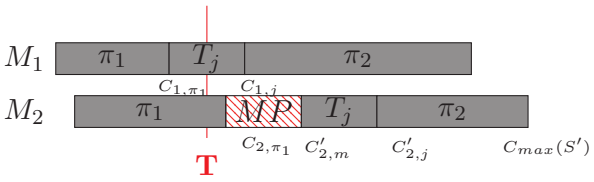

Figure 7.11 – Ordonnancement S'

Nous obtenons :

$C_{2,m} = \max\{C_{1j}; C_{2,\pi_1}\} + b_{2,j} + \alpha[\max\{C_{1,j}; C_{2,\pi_1}\} + b_{2,j}] + \beta$

$C'_{2,j} = \max\{C_{1j}; (1+\alpha)\max\{T; C_{2,\pi_1}\} + \beta\} + a_{2,j}$

Nous examinons maintenant les deux cas suivants :

- Cas 1 : Si $C_{1,j} \leq C_{2,\pi_1}$

  En utilisant l'inégalité 7.2, nous obtenons $C_{2,\pi_1} \geq T$. Ainsi, nous avons :

  $C_{2,m} = (1+\alpha)(C_{2,\pi_1} + b_{2,j}) + \beta$, et $C'_{2,j} = (1+\alpha)C_{2,\pi_1} + a_{2,j} + \beta$.

  Alors $C_{2,m} - C'_{2,j} = (1+\alpha)b_{2,j} - a_{2,j}$. Or $b_{2,j} \geq a_{2,j}$, donc $C_{2,m} \geq C'_{2,j}$.
- Cas 2 : Si $C_{1,j} \geq C_{2,\pi_1}$

  En utilisant l'inégalité 7.2, nous obtenons : $C_{1,j} \geq T$. Nous avons :

  $C_{2,m} = (1+\alpha)(C_{1,j} + b_{2,j}) + \beta$, et $C'_{2,j} = \max\{C_{1j}; (1+\alpha)\max\{T; C_{2,\pi_1}\} + \beta\} + a_{2,j}$.

  Deux cas sont possibles :
  - Cas 2.1 : $C_{1j} \leq (1+\alpha)\max\{T; C_{2,\pi_1}\} + \beta$

    Dans ce cas, nous avons

    $C'_{2,j} = (1+\alpha)\max\{T; C_{2,\pi_1}\} + a_{2,j} + \beta$.

    puisque $b_{2,j} \geq a_{2,j}$, $C_{1,j} \geq C_{2,\pi_1}$ et $C_{1,j} \geq T$, on obtient $C_{2,m} \geq C'_{2,j}$.
  - Cas 2.2 : $C_{1,j} \geq (1+\alpha)\max\{T; C_{2,\pi_1}\} + \beta$

    Nous avons $C'_{2,j} = C_{1,j} + a_{2,j}$.

    puisque $b_{2,j} \geq a_{2,j}$, on obtient $C_{2,m} \geq C'_{2,j}$.

Ainsi, le makespan de S' ($C_{max}(S')$) est inférieur ou égal au makespan de l'ordonnancement S ($C_{max}(S)$). Nous concluons donc qu'il existe un ordonnancement optimal dans lequel la dernière tâche qui précède la maintenance démarre avant l'instant T.

□

Propriété 3. *Si la dernière tâche T_j de l'ensemble B sur M_2 démarre avant l'instant T et se termine après l'instant T ($C_{2,j} > T$ et $C_{2,j} - b_{2,j} < T$), dans ce cas, le déplacement de la tâche T_j vers l'ensemble A permet d'améliorer le C_{max}, si cette inégalité est vérifiée :*

$$C_{2,j} \geq T + \frac{a_{2,j}}{1+\alpha}$$

Démonstration. Nous considérons l'ordonnancement présenté dans la figure 7.12. avec π_1 et π_2 deux ordonnancements partiels. À l'instant T, la machine M_2 est occupée par la tâche T_j et la maintenance est insérée le plus tôt possible après la fin de cette tâche. Nous utiliserons les notations présentées précédemment.

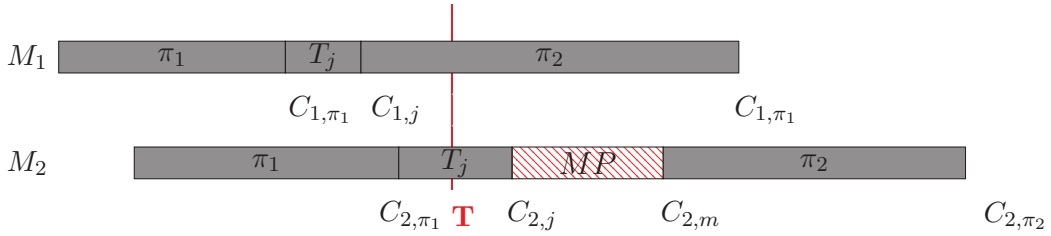

Figure 7.12 – Ordonnancement S

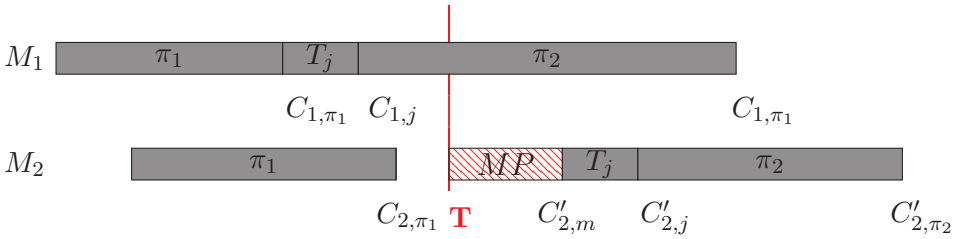

Figure 7.13 – Ordonnancement S'

Soit S' le même ordonnancement avec le déplacement de la maintenance à l'instant T et le placement de la tâche T_j après la fin de la maintenance, voir figure 7.13.

Ce déplacement permet d'améliorer le makespan, si $C_{2,m} \geq C'_{2,j}$. Or, nous avons :

$$C_{2,m} = (1+\alpha)C_{2,j} + \beta \text{ et } C'_{2,j} = (1+\alpha)T + \beta + a_{2,j}$$

Ainsi, le makespan de S' est diminué, si cette inégalité est vérifiée : $C_{2,j} \geq T + \frac{a_{2,j}}{1+\alpha}$ □

Notons qu'avec le problème $F2|(b_{lj}, a_{lj}), nr - \mathcal{M}_2 = 1, m = \alpha t + \beta | C_{max}$, avec $t \geq T$, on ne peut pas parler d'une séquence de Johnson globale pour toutes les tâches à cause des temps opératoires variables avant et après la maintenance. Donc ici, on parle plutôt d'une séquence de Johnson locale pour les tâches qui se trouvent avant ou après la période d'indisponibilité.

On note aussi que lorsque $b_{lj} = a_{lj}$, *i.e.* pas de modification de performances après la maintenance, l'algorithme de Johnson est vérifié localement (avant et après la maintenance) et est non optimal pour l'ensemble des tâches. Nous illustrons cette remarque dans l'exemple suivant (exemple 7.1).

Exemple 7.1. *Nous considérons le problème* $F2|(b_{lj}, a_{lj}), nr-\mathcal{M}_2 = 1, m = \alpha t+\beta|C_{max}$, *avec* $b_{lj} = a_{lj}$, $n = 4$ *et une période de maintenance sur la deuxième machine. Soit* $\alpha = \beta = 1$ *et* $T = 6$. *les temps opératoires sont listés dans le tableau 7.3.*

Tableau 7.3 – Exemple

Tâche j	b_{1j}	$b_{2j} = a_{2j}$
T_1	1	4
T_2	2	3
T_3	4	2
T_4	2	1

La solution optimale de l'exemple est égale à $C^*_{max} = 18$ *et est obtenue par la séquence suivante* $\{T_1 - T_4 - T_2 - T_3\}$, *voir figure 7.14.*

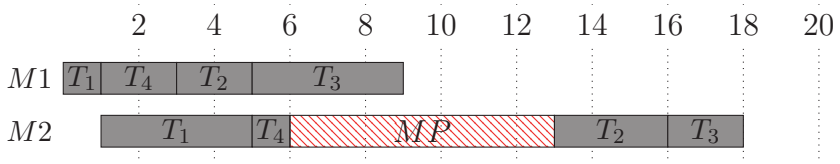

Figure 7.14 – La séquence $\{T_1 - T_4 - T_2 - T_3\}$

En appliquant l'algorithme 8, la séquence de Johnson est donnée par : $\{T_1-T_2-T_3-T_4\}$ *avec un* $C_{max} = 19$, *voir figure 7.15. Notons qu'en utilisant la propriété 3, la tâche* T_2 *est insérée après la maintenance.*

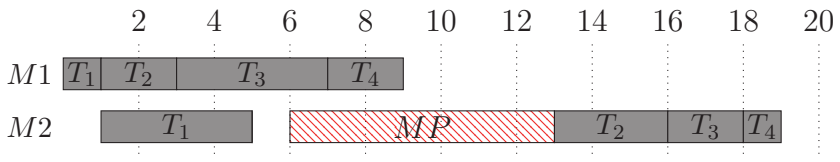

Figure 7.15 – La séquence $\{T_1 - T_2 - T_3 - T_4\}$

Nous pouvons remarquer que dans la solution optimale l'algorithme de Johnson est localement vérifié avant et après la maintenance $(T_1 - T_4)$ *et* $(T_2 - T_3)$ *et non optimal pour l'ensemble des tâches (la séquence* $\{T_1 - T_4 - T_2 - T_3\}$ *ne vérifie pas Johnson).*

7.5.3 Heuristiques

Une fois les propriétés d'optimalité déterminées, nous présentons dans cette section deux heuristiques de résolution. Ces heuristiques ont été inspirées de deux heuristiques classiques développées par Lee [73] pour résoudre des problèmes flow shop avec une période de maintenance fixe et sans modification de la performance de la machine.

En se basant sur les propriétés d'optimalité, nous présentons dans ce qui suit les deux heuristiques.

7.5.3.1 Heuristique 1 : H1

L'heuristique (H1) utilise la séquence de Johnson. Dans un premier temps, nous trions les tâches selon l'algorithme de Johnson sans tenir compte de la maintenance et de l'amélioration de performance qu'elle induit, avec $b_{1,j}$ et $b_{2,j}$ les temps opératoires sur la première et la deuxième machine. En utilisant, les propriétés 2 et 3, la maintenance doit être insérée le plus tôt possible. Ensuite, la proposition 3 est utilisée pour améliorer le makespan. Notre heuristique est présentée ci-dessous :

Algorithme 10 : Heuristique 1 : H1 — Complexité $O(n \log n)$

1 Trier les tâches selon l'algorithme de Johnson sans tenir compte de la maintenance, *i.e.* $b_{1,j}$ et $b_{2,j}$ les temps opératoires sur la première et la deuxième machine, respectivement
2 Placer dans l'ordre les tâches dans l'ensemble B tant que les dates de début des tâches sur M_2 ne dépassent pas l'instant T. Soit $C_{2,B}$ la date de fin de l'ensemble B sur la deuxième machine
3 **si** $C_{2,B} < T + \frac{a_{2,j}}{(1+\alpha)}$ **alors**
  | /* avec T_j la dernière tâche dans l'ensemble B */
4 | Placer la maintenance directement après l'ensemble B (à l'instant $C_{2,B}$) ;
sinon
5 | Retirer la dernière tâche de l'ensemble B et placer la maintenance à l'instant T ;
fin si
6 Ordonnancer le reste des tâches après la maintenance dans l'ordre de Johnson, avec $b_{1,j}$ et $a_{2,j}$ les temps opératoires, respectivement, sur la première et la deuxième machine avec $j \in \{N \backslash B\}$;

7.5.3.2 Heuristique 2 : H2

Heuristique 2 (H2) est similaire à H1, cependant les tâches sont triées dans l'ordre décroissant de $\frac{b_{1,j}}{b_{2,j}}$, voir algorithme 11.

Algorithme 11 : Heuristique 2 : H2 — Complexité $O(n \log n)$

1 Trier les tâches dans l'ordre décroissant de $\frac{b_{1,j}}{b_{2,j}}$ sans tenir compte de la maintenance *i.e.* $b_{1,j}$ et $b_{2,j}$ les temps opératoires sur la première et la deuxième machine, respectivement
2 Placer dans l'ordre les tâches dans l'ensemble B tant que les dates de début des tâches sur M_2 en appliquant l'algorithme de Johnson ne dépassent pas l'instant T. Soit $C_{2,B}$ la date de fin de l'ensemble B sur la deuxième machine
3 **si** $C_{2,B} < T + \frac{a_{2,j}}{(1+\alpha)}$ **alors**
| /* avec T_j la dernière tâche dans l'ensemble B */
4 | Placer la maintenance directement après l'ensemble B ;
sinon
5 | Retirer la dernière tâche de l'ensemble B et placer la maintenance à l'instant T ;
fin si
6 Ordonnancer le reste des tâches après la maintenance dans l'ordre de Johnson, avec $b_{1,j}$ et $a_{2,j}$ les temps opératoires sur la première et la deuxième machine avec $j \in \{N \backslash B\}$;

Notons que la complexité des heuristiques (H1) et (H2) est de $O(n \log n)$.

7.5.4 Méthodes de résolution exactes

Cette section vise à proposer des méthodes de résolution exactes qui permettent de trouver des solutions optimales pour le problème $F2|(b_{lj}, a_{lj}), nr - \mathcal{M}_2 = 1, m = \alpha t + \beta|C_{max}$, dont la période de maintenance doit être insérée à partir de l'instant T, *i.e.* $t \geq T$. Les deux premières sous-sections seront consacrées à la présentation d'une méthode énumérative et un programme linéaire mixte. Ensuite, nous présentons une méthode de résolution basée sur l'approche de séparation et évaluation (Branch & Bound). Des résultats expérimentaux, qui mettent en évidence les performances des méthodes exposées, sont présentés à la fin de la section.

7.5.4.1 Résolution du problème par une méthode énumérative

À partir des résultats présentés dans la section 7.5.2, une première méthode énumérative peut être développée, qui consiste à évaluer toutes les combinaisons possibles afin de trouver la solution optimale. Dans ce qui suit, nous présentons cette méthode de résolution, nous définissons aussi des règles utilisées afin de réduire l'espace de recherche.

Le principe de cette méthode est donc de déterminer, de manière itérative, la séquence de tâches qui permet de trouver la solution optimale. Soit n_B et n_A le nombre de tâches, respectivement, dans l'ensemble B et A, avec B (A) l'ensemble des tâches ordonnancées avant (après) la période de maintenance, voir figure 7.16.

Pour chaque n_B, nous énumérons toutes les combinaisons possibles à insérer dans l'ensemble B. Ces combinaisons doivent être de taille n_B et respectent l'ordre de Johnson. Le reste des tâches est ordonnancé dans l'ensemble A selon l'algorithme de Johnson (d'après la propriété 1). La taille n_B de l'ensemble B varie entre 0 et n, alors le nombre de combinaisons possibles est égal à $\sum_{n_B=0}^{n} \binom{n}{n_B} = 2^n$, avec $\binom{n}{i}$ le coefficient binomial qui correspond au nombre de combinaisons de i éléments parmi n.

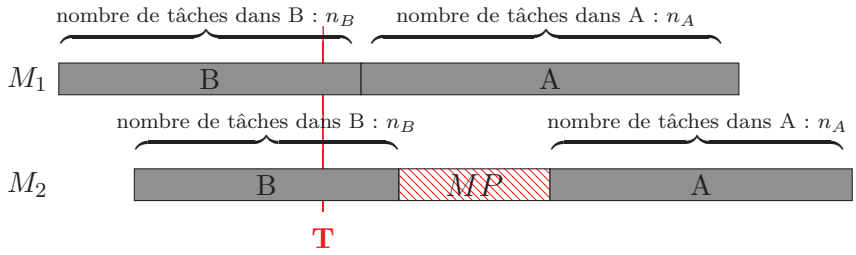

Figure 7.16 – Nombre de tâches dans l'ensemble B et A

Règles pour minimiser l'espace de recherche

À partir des propriétés d'optimalité présentées précédemment, nous définissons trois règles pour la méthode énumérative, ce qui permet de diminuer considérablement l'espace de recherche.

Règle n°1 : D'après la propriété 2, la maintenance doit être insérée le plus tôt possible. Ainsi, nous pouvons déterminer une borne supérieure pour le nombre de tâches à insérer avant la période d'indisponibilité (ensemble B). Soit $n_{B_{max}}$ le nombre maximal de tâches à insérer dans l'ensemble B, cette borne est déterminée de la façon suivante :

Algorithme 12 : Déterminer la borne supérieure de $n_{B_{max}}$

début

1 Numéroter les tâches selon l'ordre croissant $b_{2,j}$

2 **retourner** $n_{B_{max}}$ *tel que* $\min_j b_{1,j} + \sum_{j=1}^{n_{B_{max}-1}} b_{2,j} < T$ *et* $\min_j b_{1,j} + \sum_{j=1}^{n_{B_{max}}} b_{2,j} \geq T$

fin

La borne supérieure $n_{B_{max}}$ permet de limiter le nombre de combinaisons à évaluer. Nous illustrons celui-ci, dans l'exemple ci-dessous. Pour un problème avec $n = 6$ et $n_{B_{max}} = 2$. Le nombre total de combinaisons à énumérer est égal à $2^6 = 64$ combinaisons. Par contre, en utilisant la règle précédente, le nombre de combinaisons est réduit à $\binom{6}{1} + \binom{6}{2} = 21$.

Notons que $\binom{6}{1}$ est le coefficient binomial qui correspond au nombre de combinaisons possibles lorsque $n_B = 1$ et $\binom{6}{2}$ est égal au nombre de combinaisons pour $n_B = 2$. Ainsi, en utilisant cette règle, l'espace de recherche est diminué de 64 à 21 combinaisons, soit de 67 %.

Remarque 7.1. *La considération de l'ensemble B ou l'ensemble A pour générer les combinaisons n'a pas d'impact sur la performance de méthode de résolution, car d'après l'analyse combinatoire, nous avons :*

$$\binom{m}{k} = \binom{m}{m-k}$$

Or, $n_B = n - n_A$ ainsi le nombre de combinaisons, lorsqu'on considère l'ensemble B, est le même si on considère l'ensemble A :

$$\binom{n}{n_B} = \binom{n}{n-n_B} = \binom{n}{n_A}$$

Règle n°2 : La maintenance doit être ordonnancée le plus tôt possible. Ainsi, nous pouvons rejeter les séquences dont la date de début de la dernière tâche de B démarre après ou à l'instant T.

Par exemple, la séquence présentée dans la figure 7.17 est à rejeter, car la date de début de la dernière tâche de B sur M_2 (la tâche T_j) est supérieure à l'instant T.

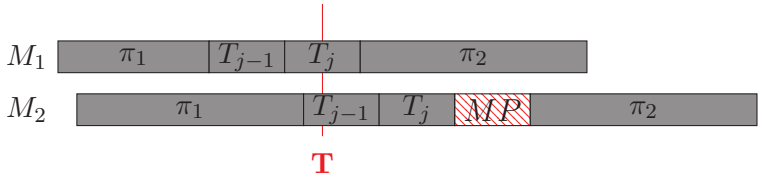

Figure 7.17 – Séquence à éliminer d'après la règle n°2

Règle n°3 : En se basant sur la propriété d'optimalité 3, nous rejetons les séquences dont la date de début de la dernière tâche T_j de l'ensemble B sur M_2 est inférieure à l'instant T et sa date de fin vérifie :

$$C_{2,B} \geq T + \frac{a_{2,j}}{(1+\alpha)}$$

Nous présentons, dans la figure 7.18, l'algorigramme de cette méthode.

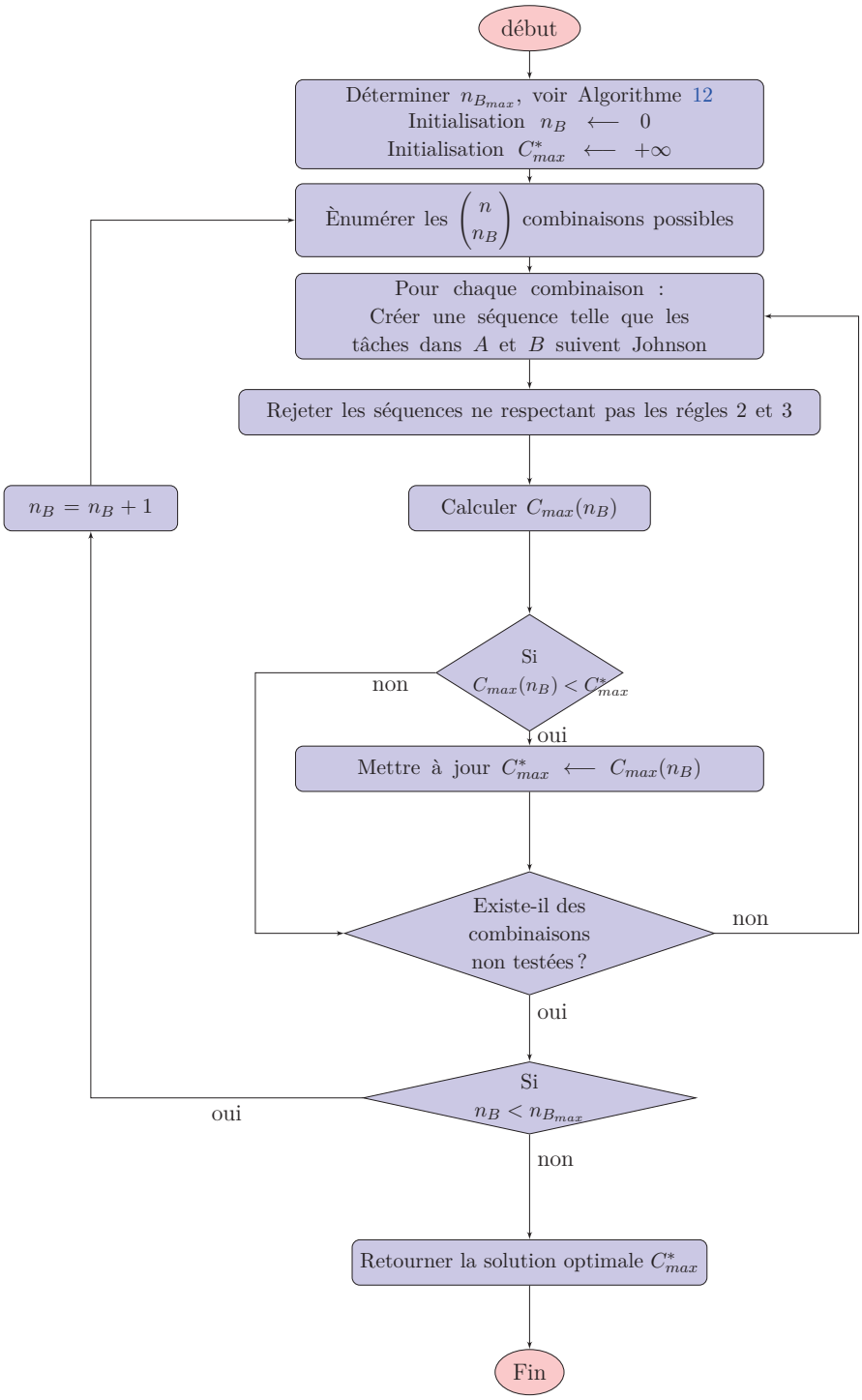

Figure 7.18 – Algorigramme de la méthode énumérative

Exemple 7.2. *Cette méthode est illustrée à l'aide de l'exemple ci-dessous. Le tableau 7.4 présente les temps opératoires des tâches à ordonnancer sur deux machines flow shop, avec* $T = 8$, $\alpha = 0,2$ *et* $\beta = 1$.

Dans un premier temps, nous commençons par déterminer la borne supérieure $n_{B_{max}}$. *Pour cela, nous trions les tâches dans l'ordre croissant de* $b_{2,j}$, *voir Algorithme 12. Ainsi,*

Tableau 7.4 – Les temps opératoires des tâches

Tâches	1	2	3	4	5
$b_{1,j}$	1	2	4	3	5
$b_{2,j}$	5	6	3	7	4
$a_{2,j}$	2	1	3	4	3

le nombre $n_{B_{max}}$ est égal à 2.

Le nombre de séquences à évaluer $\binom{5}{1} + \binom{5}{2} = 5 + 10 = 15$.

Ensuite, nous rejetons les séquences ne respectant pas les règles 2 et 3. Ces règles permettent de réduire le nombre de séquences à évaluer à 7 séquences seulement. Nous évaluons les différentes séquences dans le tableau 7.5.

Tableau 7.5 – Les séquences de la méthode énumérative

n_B	**Séquence**	C_{max}
0	$MP - T_1 - T_4 - T_5 - T_3 - T_2$	$23,6$
1	$T_1 - MP - T_4 - T_5 - T_3 - T_2$	$21,6$
	$T_2 - MP - T_1 - T_4 - T_5 - T_3$	$22,6$
	$T_3 - MP - T_1 - T_4 - T_5 - T_2$	$20,6$
	$T_4 - MP - T_1 - T_5 - T_3 - T_2$	22
	$T_5 - MP - T_1 - T_4 - T_3 - T_2$	$21,8$
2	$T_1 - T_2 - MP - T_4 - T_5 - T_3$	Rejeter : ne respecte pas la règle 3
	$T_1 - T_4 - MP - T_5 - T_3 - T_2$	Rejeter : ne respecte pas la règle 3
	$T_1 - T_5 - MP - T_4 - T_3 - T_2$	21
	$T_1 - T_3 - MP - T_4 - T_5 - T_2$	$19,8^*$
	$T_2 - T_4 - MP - T_1 - T_5 - T_3$	Rejeter : ne respecte pas la règle 2
	$T_2 - T_5 - MP - T_1 - T_4 - T_3$	Rejeter : ne respecte pas la règle 2
	$T_2 - T_3 - MP - T_1 - T_4 - T_5$	Rejeter : ne respecte pas la règle 2
	$T_4 - T_5 - MP - T_1 - T_3 - T_2$	Rejeter : ne respecte pas la règle 2
	$T_4 - T_3 - MP - T_1 - T_5 - T_2$	Rejeter : ne respecte pas la règle 2
	$T_5 - T_3 - MP - T_1 - T_4 - T_2$	Rejeter : ne respecte pas la règle 2

* : solution optimale

Ainsi, la solution optimale est égale à $C^*_{max} = 19,8$ et est obtenue par la séquence : $T_1 - T_3 - MP - T_4 - T_5 - T_2$, la maintenance est insérée après la tâche T_3 sur M_2.

Malgré l'utilisation de ces règles pour réduire l'espace de recherche à explorer, la méthode énumérative reste limitée à des problèmes de petite taille. Nous proposons, dans ce qui suit, d'autres méthodes qui assurent la résolution du problème avec des instances de grande taille.

7.5.4.2 Résolution du problème par la programmation linéaire mixte

Le problème d'ordonnancement $F2|(b_{lj}, a_{lj}), nr - \mathcal{M}_2 = 1, m = \alpha t + \beta|C_{max}$, avec une période de maintenance à ordonnancer à partir de l'instant T, *i.e.* $T \leq t$, peut être formulé sous forme d'un programme linéaire mixte. Dans cette section, nous proposons une modélisation du problème considéré utilisant la programmation linéaire mixte.

La formulation de ce programme linéaire est basée sur des variables de décision, dites variables de position, voir Pinedo [94]. L'idée de base de cette modélisation est de définir deux ensembles A et B de taille, respectivement, n et $n_{B_{max}}$ positions. Le principe consiste à affecter les n tâches à des positions, et à déterminer les dates de début et de fin pour chaque tâche associée à une position. Chaque position i dans l'ensemble s ($s = \{A, B\}$) est caractérisée par la date de fin de la tâche insérée à cette position sur la première et la deuxième machine, respectivement, $C^s_{1,i}$ et $C^s_{2,i}$.

Il est clair que le nombre de positions dans les deux ensembles A et B est supérieur au nombre total de tâches $n_{B_{max}}+n > n$. Donc, dans la solution finale du programme linéaire, nous obtenons des positions vides dans la séquence d'ordonnancement. Ainsi, après la résolution du programme linéaire, nous procédons à la suppression de ces positions vides et nous obtenons ainsi la séquence optimale. Notons que l'existence de ces positions vides ne modifie pas l'optimalité de la solution obtenue, ces positions vides sont considérées comme des tâches avec une durée nulle sur la première et la deuxième machine $b_{1,j} = b_{2,j} = a_{1,j} = a_{2,j} = 0$.

Soit $x^s_{j,i}$ une variable de décision avec :

$$x^s_{j,i} = \begin{cases} 1 & \text{si la tâche } T_j \text{ est dans la position } i \text{ de } s = \{A, B\} \\ 0 & \text{sinon} \end{cases}.$$

La date d'achèvement de la tâche en position i dans l'ensemble s sur la machine M_l est définie par $C^s_{l,i}$.

L'objectif est la minimisation de la date de fin (makespan). Nous proposons le programme linéaire suivant :

$$min \quad C^A_{2,n} \tag{7.3}$$

sous les contraintes :

$$C^s_{1,i} = C^s_{1,i-1} + \sum_{j=1}^{n} b_{1,j} x^s_{j,i}, \quad s \in \{A, B\} \; et \; i = \begin{cases} 2, \ldots, n_{B_{max}} & \text{si } s = B \\ 2, \ldots, n & \text{si } s = A \end{cases} \tag{7.4}$$

$$C^B_{2,i} \geq C^B_{1,i} + \sum_{j=1}^{n} b_{2,j} x^B_{j,i}, \quad i = 1, \ldots, n_{B_{max}} \tag{7.5}$$

$$C^B_{2,i} \geq C^B_{2,i-1} + \sum_{j=1}^{n} b_{2,j} x^B_{j,i}, \; i = 2, \ldots, n_{B_{max}} \tag{7.6}$$

$$C^A_{2,i} \geq C^A_{1,i} + \sum_{j=1}^{n} a_{2,j} x^A_{j,i}, \quad i = 1, \ldots, n \tag{7.7}$$

$$C^A_{2,i} \geq C^A_{2,i-1} + \sum_{j=1}^{n} a_{2,j} x^A_{j,i}, \quad i = 2, \ldots, n \tag{7.8}$$

$$\sum_{i=1}^{n} x^A_{j,i} + \sum_{i=1}^{n_{B_{max}}} x^B_{j,i} = 1, \quad j = 1, \ldots, n \tag{7.9}$$

$$\sum_{j=1}^{n} x^s_{j,i} \leq 1, \ s \in \{A, B\} \ et \ i = \begin{cases} 1, \ldots, n_{B_{max}} & \text{si } s = B \\ 1, \ldots, n & \text{si } s = A \end{cases} \tag{7.10}$$

$$C^A_{2,i} \geq (1+\alpha)T + \beta + \sum_{j=1}^{n} a_{2,j} x^A_{j,i}, \ i = 1, \ldots, n \tag{7.11}$$

$$C^A_{2,i} \geq (1+\alpha)C^B_{2,n} + \beta + \sum_{j=1}^{n} a_{2,j} x^A_{j,i}, \ i = 1, \ldots, n \tag{7.12}$$

$$C^B_{1,1} = \sum_{j=1}^{n} b_{1,j} x^B_{1,j} \tag{7.13}$$

$$C^B_{2,1} = C^B_{1,1} + \sum_{j=1}^{n} b_{2,j} x^B_{1,j} \tag{7.14}$$

$$C^A_{1,1} = C^B_{1,n} + \sum_{j=1}^{n} b_{1,j} x^A_{1,j} \tag{7.15}$$

avec

$$x^s_{j,i} \in \{0,1\}, \forall \ j = 1, \ldots, n, \ s \in \{A, B\} \, et \, i = \begin{cases} 1, \ldots, n_{B_{max}} & \text{si } s = B \\ 1, \ldots, n & \text{si } s = A \end{cases} \tag{7.16}$$

$$C^s_{l,i} \geq 0, \forall l \in \{1,2\}, \ s \in \{A, B\} \, et \, i = \begin{cases} 1, \ldots, n_{B_{max}} & \text{si } s = B \\ 1, \ldots, n & \text{si } s = A \end{cases} \tag{7.17}$$

Les contraintes (7.4)-(7.8) définissent les dates de fin de chaque position sur la première et la deuxième machine. L'ensemble des contraintes (7.9)-(7.10) impose une affectation faisable : les contraintes (7.9) assurent que chaque tâche doit être affectée à une seule et unique position dans l'ordonnancement et les contraintes (7.10) garantissent que chaque position est occupée par une tâche au plus. Les contraintes (7.11) et (7.12) assurent que les tâches dans l'ensemble A doivent démarrer après la période de maintenance. Les contraintes (7.13)-(7.15) initialisent les variables $C^B_{1,1}$, $C^B_{2,1}$ et $C^A_{1,1}$. Les contraintes (7.16)-(7.17) déterminent les domaines de définitions des variables.

7.5.4.3 Résolution du problème par séparation et évaluation (Branch & Bound)

Nous développons, dans cette section, une approche de résolution basée sur l'algorithme de séparation et évaluation, permettant de résoudre de manière exacte le problème d'ordonnancement considéré en présence d'une période de maintenance à insérer à partir l'instant T, *i.e.* $t \geq T$. Notons que Hnaien et *al.* [53] ont également développé une méthode de séparation et évaluation, mais pour un problème d'ordonnancement flow shop à deux machines en présence d'une période d'indisponibilité connue d'avance (machine indisponible entre $[s, t]$) sur la première machine et sans modifier les performances de la machine (les durées des tâches avant et après la maintenance sont constantes $a_{l,j} = b_{l,j}$).

Rappelons que la méthode $B\&B$ consiste à trouver une solution optimale dans un ensemble de solutions possibles. La méthode 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.

Formulation Dans l'approche par séparation et évaluation, proposée dans ce chapitre, nous utilisons la propriété 1. L'utilisation de cette propriété permet de réduire considérablement la taille de l'arbre de recherche de l'algorithme. Nous trions donc les tâches dans l'ordre de Johnson, avec $b_{1,j}$ et $b_{2,j}$ les temps opératoires, respectivement, sur la première et la deuxième machine. Notre arbre de recherche est présenté comme suit ; nous avons $n + 1$ niveaux, chaque niveau k présente l'insertion de la tâche qui se trouve dans la position k de la séquence de Johnson.

La stratégie d'exploration utilisée dans l'algorithme est une exploration en profondeur (Depth Exploration). À chaque niveau k, nous avons deux nœuds fils possibles ; nœud k^B si la tâche k est ordonnancée avant la période de maintenance, *i.e.* $k \in B$, et un nœud k^A si la tâche k est ordonnancée après la période de maintenance, *i.e.* $k \in A$, voir figure 7.19.

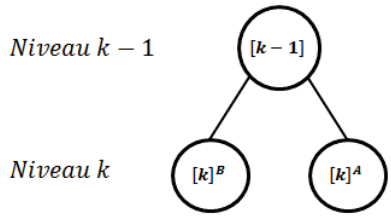

Figure 7.19 – Les nœuds de l'algorithme séparation et évaluation

Le nœud initial *root* se trouve au niveau zéro de l'arbre de recherche avec une borne supérieure initiale (BS_{init}) et une borne inférieure initiale (BI_{init}).

Dans le but de diminuer l'espace de recherche, nous présentons dans ce qui suit, les bornes inférieures et supérieures utilisées dans l'approche séparation et évaluation ($B\&B$).

Borne supérieure La borne supérieure est calculée une première fois avant la construction de l'arbre de recherche, et plus précisément au niveau du nœud '*root*'. Cette borne est calculée à partir des heuristiques présentées dans la section 7.5.3. Soit $C_{max}(H1)$ et $C_{max}(H2)$ les makespan donnés, respectivement, par les heuristiques $H1$ et $H2$. Ainsi, la borne supérieure initiale est donnée par la meilleure valeur de C_{max} trouvée par les deux heuristiques : $BS_{init} = \min\{C_{max}(H1); C_{max}(H2)\}$.

Les bornes inférieures Nous proposons, dans ce paragraphe, trois bornes inférieures (deux bornes dynamiques et une borne statique BI_2). Notons que les bornes dynamiques sont calculées à chaque exploration d'un nœud, afin d'évaluer la faisabilité du nœud exploré. Dans ce qui suit, on note A_k et B_k les ensembles des tâches ordonnancées, respectivement, avant et après la maintenance jusqu'au niveau k.

– **Borne inférieure (BI_1)** :

$$BI_1 = (1+\alpha)T + \beta + \sum_{j \in A_k} a_{2,j} \tag{7.18}$$

Démonstration. Il est clair que la durée de la maintenance est minimisée, si la période de maintenance est lancée à l'instant T, nous avons donc :

$$C_{max} \geq (1+\alpha)T + \beta + \sum_{j \in A_k} a_{2,j} + \sum_{j \in A_k} I_{[j]-1,[j]}$$

avec $I_{[j]-1,[j]}$ le temps mort entre la tâche placée en position j et la tâche $j-1$. Or $\sum_{j \in A_k} I_{[j]-1,[j]} \geq 0$, donc :

$$C_{max} \geq (1+\alpha)T + \beta + \sum_{j \in A_k} a_{2,j}$$

□

– **Borne inférieure (BI_2)** : Dans un problème d'ordonnancement flow shop à deux machines avec une période de maintenance à placer après l'instant T, le makespan (C_{max}) est égal ou supérieur à :

$$BI_2 = \max\{(1+\alpha)T + \beta; \sum_{j \in N} b_{1,j}\} + \min_{j \in N}\{a_{2,j}\} \tag{7.19}$$

Démonstration. Il est évident que le makespan (C_{max}) est supérieur ou égal à la somme des temps opératoires sur la première machine de toutes les tâches et la durée

minimale sur la deuxième machine $(\min_{j\in N}\{a_{2,j}\}) : C_{max} \geq \sum_{j\in N} b_{1,j} + \min_{j\in N}\{a_{2,j}\}$. Cependant, il existe une période de maintenance sur la deuxième machine. La durée de la maintenance est minimisée lorsqu'elle démarre à l'instant T et dont la durée est égale à $(1+\alpha)T+\beta$, voir figure 7.20. Nous obtenons ainsi l'expression suivante :

$$C_{max} \geq \max\{(1+\alpha)T+\beta; \sum_{j\in N} b_{1,j}\} + \min_{j\in N}\{a_{2,j}\} \tag{7.20}$$

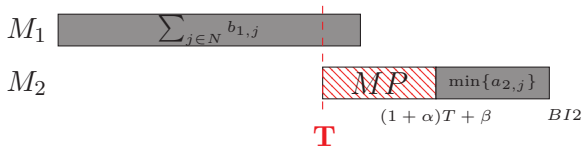

Figure 7.20 – La borne inférieure BI_2

□

– **Borne inférieure (BI_3)** : Soit $C_l(B_k)$ et $C_l(A_k)$, respectivement, les dates d'achèvement de B_k et A_k sur la machine M_l, $(l = 1, 2)$.
-Cas 1 : Si $C_2(B_k) \geq T$, la maintenance est insérée à l'instant $C_2(B_k)$. Dans ce cas, le reste des tâches doit être placé dans l'ensemble A (figure 7.21), alors :

$$BI_3 = \max\{C_1(B_k) + \min_{j\in N-\{A_k\cup B_k\}} b_{1,j}; (1+\alpha)C_2(B_k)+\beta+\sum_{j\in A_k} a_{2,j}\} + \sum_{j\in N-\{A_k\cup B_k\}} a_{2,j}$$

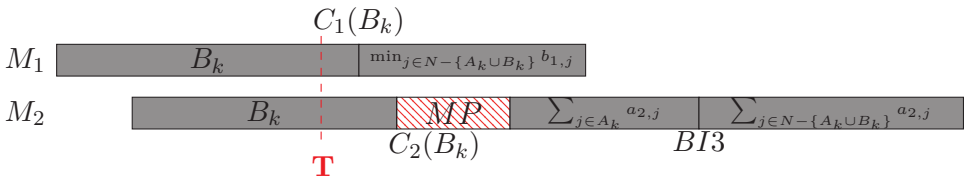

Figure 7.21 – La borne inférieure BI_3 : cas 1

-Cas 2 : Si $C_2(B_k) < T$, la maintenance est insérée à l'instant T, voir figure 7.22.

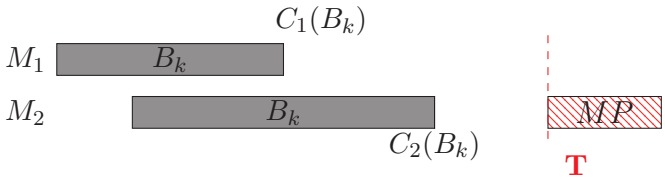

Figure 7.22 – La borne inférieure BI_3 : cas 2

Nous distinguons deux sous-cas :
- **Cas 2.1** Si $(C_2(B_k) < T)$ et $(C_1(B_k) + \min_{j\in N-\{A_k\cup B_k\}} b_{1,j} < T)$, des tâches peuvent

être insérées dans l'ensemble B (figure 7.23), alors :

$$BI_3 = C_2(B_k) + T\alpha + \beta + \sum_{j \in A_k} a_{2,j} + \sum_{j \in N - \{A_k \cup B_k\}} a_{2,j}$$

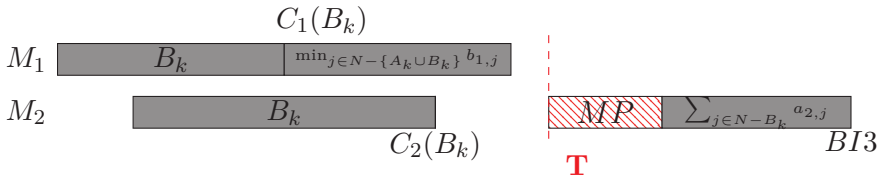

Figure 7.23 – La borne inférieure BI_3 : cas 2.1

- **Cas 2.2** Si $(C_2(B_k) < T)$ et $(C_1(B_k) + \min_{j \in N-\{A_k \cup B_k\}} b_{1,j} \geq T)$, la maintenance doit être insérée à l'instant T (figure 7.24), alors :

$$BI_3 = (1+\alpha)T + \beta + \sum_{j \in A_k} a_{2,j} + \sum_{j \in N - \{A_k \cup B_k\}} a_{2,j}$$

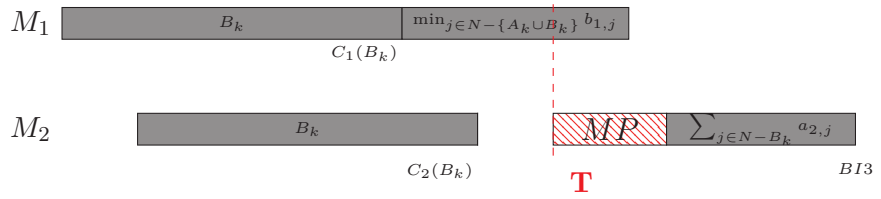

Figure 7.24 – La borne inférieure BI_3 : cas 2.2

Ainsi la borne inférieure à considérer est la suivante :

$$BI = \max\{BI_1, BI_2, BI_3\}$$

Cette borne est utilisée aussi pour calculer la borne inférieure du nœud '*root*' (niveau $k = 0$) :

$$BI_{init} = \max\{BI_1, BI_2, BI_3\}$$

Procédure de séparation et évaluation Afin d'expliquer la procédure utilisée dans l'approche de séparation et évaluation, nous présentons ci-dessous l'algorithme de la méthode (algorithme 13).

Nous présentons, dans la figure 7.25, l'algorigramme de cette méthode.

Algorithme 13 : Méthode séparation et évaluation (B & B)

1 – Trier les tâches dans l'ordre de Johnson.
2 – Créer le nœud *root* et calculer BI_{init} et BS_{init}
3 – Initialisation : Soit $BS = BS_{init}$
4 – **Séparation** : Comme mentionné précédemment, pour chaque niveau k, deux nœuds fils existent k^B si la tâche du niveau k est insérée dans l'ensemble B et k^A si la tâche du niveau k est insérée dans l'ensemble A. Nous utilisons une exploration en profondeur.
5 – **Évaluation** : Pour chaque nœud k^s, avec $s \in \{A, B\}$, nous générons la borne inférieure BI_{k^s} présentée dans le paragraphe précédent.
  5.1 si $BI_{k^s} \leq BS$, le nœud peut être un nœud dans la solution optimale, il est donc explorable.
  5.2 si $BS < BI_{k^s}$, alors ce nœud ne garantit pas une solution meilleure, le nœud est donc éliminé.
  5.3 si $C_{2,B_k} \geq T$, alors d'après la propriété 2, il est inutile d'explorer le fils $(k+1)^B$ de ce nœud.
  5.4 si k^s est une feuille, on calcule C_{max} :
    5.4.1 si $C_{max} \leq BS$: nous avons une solution faisable et meilleure que la solution actuelle, mais pas forcément optimale. La solution C_{max} obtenue est la nouvelle borne supérieure.
    5.4.2 si $C_{max} = BI_{init}$: la borne inférieure est atteinte, donc la solution obtenue est optimale.
6 – La séparation est arrêtée une fois tous les nœuds visités ou lorsque la borne inférieure BI_{init} est atteinte (étape 5.4.2).

7.5.5 Étude expérimentale

7.5.5.1 Génération d'instances

Afin de mesurer les performances des méthodes développées, nous allons tester ces méthodes sur plusieurs instances, avec différentes positions de T et plusieurs nombres de tâches n.

On considère trois groupes d'instances, chaque groupe est caractérisé par une position de T :

G_1 dans ce groupe d'instances, nous avons $T = 25\% \sum_{j=1}^{n} b_{2,j}$
G_2 dans ce groupe d'instances, nous avons $T = 50\% \sum_{j=1}^{n} b_{2,j}$
G_3 dans ce groupe d'instances, nous avons $T = 75\% \sum_{j=1}^{n} b_{2,j}$

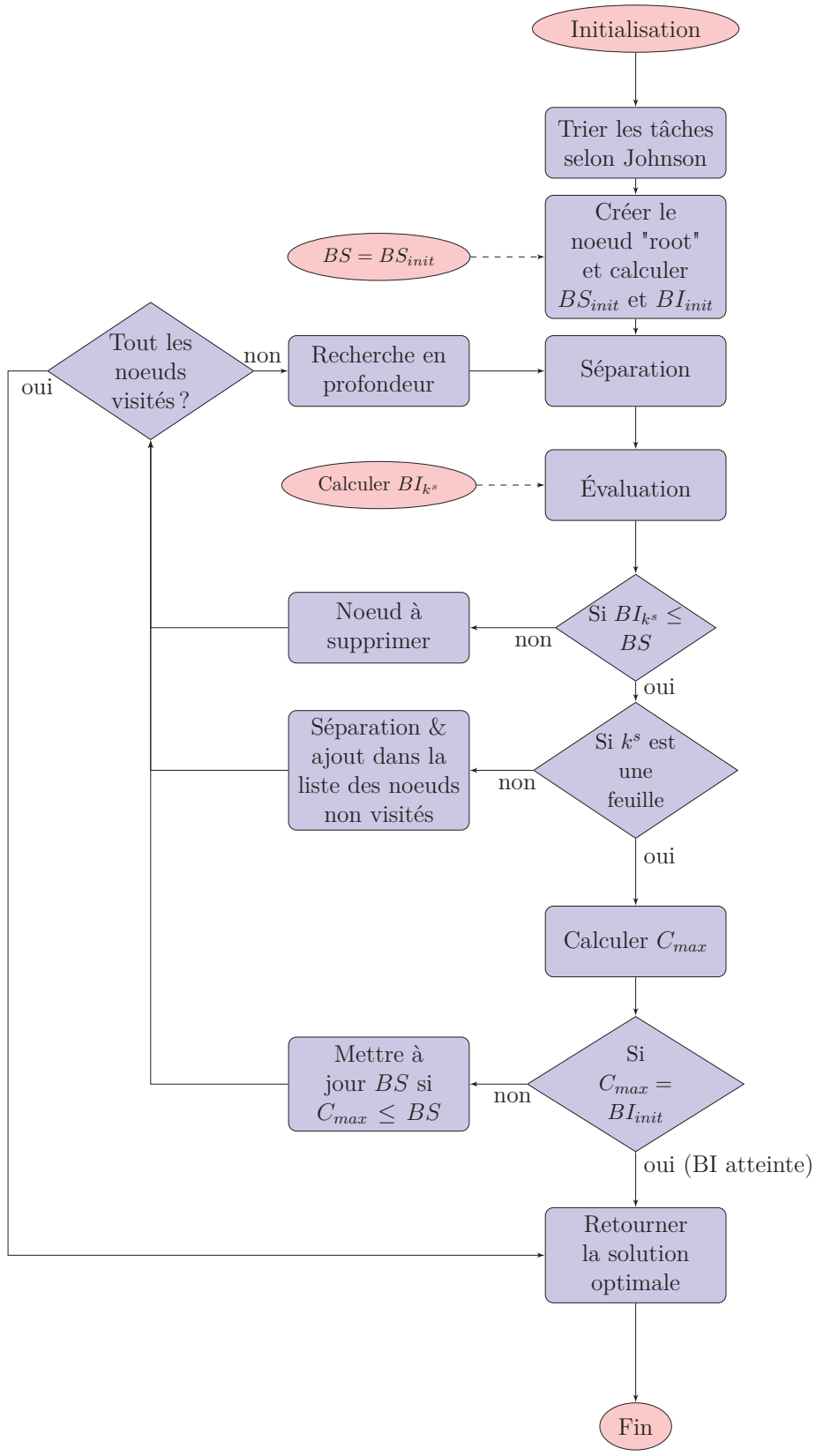

Figure 7.25 – L'algorigramme de l'approche séparation et évaluation

Vu le nombre important des instances, un générateur d'instances a été développé, qui prend comme valeurs d'entrée les paramètres suivants :

nbr_{ins} nombre d'instances pour chaque taille n.
n_{min} taille minimale de n.
n_{max} taille maximale de n.
p_{ins} différence entre deux tailles successives.
$b_{l,max}$ temps opératoire maximal sur la machine M_l, $l = \{1, 2\}$

Cet outil offre la possibilité de générer automatiquement les variables α et β. Ces valeurs sont calculées de façon à obtenir une durée de maintenance lancée à l'instant T, égale à la durée moyenne des tâches ($\frac{\sum b_{2,j}}{n}$). En outre, l'utilisateur peut aussi insérer des valeurs fixes pour les paramètres α et β.

Les durées des tâches sont générées d'une façon aléatoire et uniforme, tout en respectant les contraintes : $b_{l,j} \leq a_{l,j}$ et $b_{l,j} \leq b_{l,max}$, $\forall j = 1, \ldots, n$.

Par exemple, la configuration suivante, $n_{min} = 5$, $n_{max} = 20$, $p_{ins} = 5$ et $nbr_{ins} = 10$ permet d'obtenir les nombres d'instances présentés dans le tableau suivant :

Tableau 7.6 – Nombre d'instances pour chaque groupe et chaque nombre de tâches

	nombre de tâches n			
Position de T	5	10	15	20
G_1	10	10	10	10
G_2	10	10	10	10
G_3	10	10	10	10

Le générateur des instances est disponible sur :
http://pagesperso.g-scop.grenoble-inp.fr/~garaalia/InsGen/

7.5.5.2 Résultats numériques

Dans cette partie, nous analysons expérimentalement les différentes méthodes développées, à savoir la méthode énumérative, le programme linéaire et la méthode de séparation et évaluation. Toutes les méthodes sont développées et exécutées, sur un ordinateur personnel avec un processeur `intel® Core™ i3-4130 CPU 2,40 GHz` et 4,00 `GO` de RAM.

La méthode énumérative et l'approche par séparation et évaluation ont été implémentées en `JAVA`. Le programme linéaire a été exécuté en utilisant le logiciel `CPLEX 12.6`.

Pour ces trois méthodes , nous utilisons des instances générées par l'outil présenté dans la section précédente, avec $b_{1,j}$, $b_{2,j} \in [1, 10]$ (et $a_{2,j} \leq b_{2,j}$). Le nombre d'instances pour chaque configuration (nombre de tâches (n) et groupe) est présenté dans le tableau 7.7.

Tableau 7.7 – Nombre d'instances pour chaque groupe et chaque n

	Nombre de tâches														
Groupe	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**60**	**70**	**80**	**90**	**100**
Groupe G_1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Groupe G_2	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Groupe G_3	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

Le temps maximal de calcul est de 15 min. On arrête le calcul au bout de 15 min, si cette condition d'arrêt est atteinte, on garde la meilleure solution trouvée.

Pourcentage des instances résolues avant la condition d'arrêt

Dans un premier temps, nous présentons l'efficacité de ces méthodes en termes de pourcentage d'instances résolues à l'optimal avant la condition d'arrêt (15 minutes). Sans surprise, la méthode énumérative nous a permis de résoudre des instances de petite taille. Les pourcentages des problèmes résolus de façon optimale par la méthode énumérative sont présentés dans le tableau 7.8.

Tableau 7.8 – Pourcentages des problèmes résolus avant 15 min par la méthode énumérative (%)

	Nombre de tâches							
Groupe	**5**	**10**	**15**	**20**	**25**	**30**	35	≥ 40
Groupe G_1	100%	100%	100%	100%	100%	100%	60%	0%
Groupe G_2	100%	100%	100%	100%	100%	100%	0%	0%
Groupe G_3	100%	100%	100%	100%	100%	100%	0%	0%

Cette méthode est capable de résoudre des instances de taille allant jusqu'à $n = 30$ tâches pour toutes les positions de T (G_1, G_2 et G_3). De plus, les instances du groupe G_1 dont la taille $n = 35$ ont été résolues pour 60% des instances. Cela est expliqué par le faible nombre de combinaisons pour les instances du groupe G_1, par rapport aux autres groupes (G_2 et G_3). En effet, la taille maximale de l'ensemble B, $n_{B_{max}}$ est minimale pour les instances du groupe G_1, ce qui donne un nombre de combinaisons faible.

Les pourcentages de résolution du programme linéaire sont présentés dans le tableau 7.9. Le programme linéaire, présenté dans la section 7.5.4.2, permet de résoudre l'ensemble des instances dont la taille est inférieure ou égale à $n = 20$ avant 15 min. Pour $n = 25$, la méthode a permis de résoudre des instances des groupes G_1 et G_2 avec un

pourcentage de 70%, cependant seules 40% d'instances du groupe G_3 ont été résolues à l'optimale avant 15 *min*.

Tableau 7.9 – Pourcentages des problèmes résolus avant 15 *min* par le programme linéaire (%)

	Nombre de tâches						
Groupe	**5**	**10**	**15**	**20**	**25**	**30**	≥ 35
Groupe G_1	100%	100%	100%	100%	70%	0%	0%
Groupe G_2	100%	100%	100%	100%	70%	10%	0%
Groupe G_3	100%	100%	100%	100%	40%	10%	0%

Contrairement aux méthodes de résolution précédentes, l'approche par évaluation et séparation est capable de résoudre des instances de taille allant jusqu'à 500 tâches. Ce qui confirme l'efficacité de cette méthode et en particulier les bornes utilisées dans l'approche. Nous présentons dans le tableau 7.10, les pourcentages des problèmes résolus par cette méthode.

Tableau 7.10 – Pourcentages des problèmes résolus avant 15 *min* par l'approche $B\&B$ (%)

	Nombre de tâches									
Groupe	**5**	**10**	**15**	**20**	**25**	**30**	$35 \leq n \leq 90$	**100**	**250**	**500**
G_1	100%	100%	100%	100%	100%	100%	100%	100%	90%	70%
G_2	100%	100%	100%	100%	100%	100%	100%	100%	80%	70%
G_3	100%	100%	100%	100%	100%	100%	100%	100%	80%	60%

Temps de calcul et performances

On s'intéresse, dans ce paragraphe, à la comparaison et l'analyse de différents résultats obtenus liés au temps de calcul et les performances de ces méthodes. Le tableau 7.11 présente l'évolution du temps de calcul de la méthode énumérative pour chaque taille n et chaque groupe (G_1, G_2 et G_3).

Le programme linéaire et la méthode énumérative permettent de résoudre les instances de petite taille en un temps de calcul ne dépassant pas une minute. Le temps de calcul croît considérablement avec le nombre de tâches n.

Nous rappelons qu'on garde la meilleure solution trouvée si la condition d'arrêt est atteinte. Afin d'évaluer les solutions trouvées par rapport à la solution optimale, nous définissons le gap entre la solution trouvée par la méthode énumérative et la solution optimale obtenue par le $B\&B$, se calcule comme suit :

$$Gap_{énum} = \frac{\text{Solution méthode énumérative} - \text{Solution optimale}}{\text{Solution optimale}}$$

Les gaps maximum, minimum et moyen sont présentés dans la colonne $Gap_{énum}$ du tableau 7.11.

Tableau 7.11 – Évolution du temps de calcul (s) et des gaps (%) dans la méthode énumérative

T	n	Temps de calcul (s)			Gap_{enum} (%)		
		Min	Moy	Max	Min	Moy	Max
G_1	5	$\leq 0,001$	$\leq 0,001$	$\leq 0,001$	0	0	0
	10	$\leq 0,001$	$\leq 0,001$	$\leq 0,001$	0	0	0
	15	$\leq 0,001$	$\leq 0,001$	$\leq 0,015$	0	0	0
	20	0,016	0,033	0,048	0	0	0
	25	0,672	0,977	1,327	0	0	0
	30	13,900	35,803	49,500	0	0	0
	35	591,855,	856,876	$15\,min$	0	0	0
	40	$15\,min$: condition d'arrêt			0	0	0
	45	$15\,min$: condition d'arrêt			0	0	0
	50	$15\,min$: condition d'arrêt			0	0	0
	60	$15\,min$: condition d'arrêt			0	0	0
	70	$15\,min$: condition d'arrêt			0	0	0
	80	$15\,min$: condition d'arrêt			0	0	0
	90	$15\,min$: condition d'arrêt			0	0	0
	100	$15\,min$: condition d'arrêt			0	0	0
G_2	5	$\leq 0,001$	$\leq 0,001$	0,001	0	0	0
	10	$\leq 0,001$	$\leq 0,001$	$\leq 0,001$	0	0	0
	15	$\leq 0,001$	0,011	0,016	0	0	0
	20	0,109	0,206	0,250	0	0	0
	25	4,603	8,743	10,593	0	0	0
	30	276,200	370,320	446,614	0	0	0
	35	$15\,min$: condition d'arrêt			0	0	0
	40	$15\,min$: condition d'arrêt			0	0	0
	45	$15\,min$: condition d'arrêt			0	0	0
	50	$15\,min$: condition d'arrêt			0	0	3
	60	$15\,min$: condition d'arrêt			0	3	12
	70	$15\,min$: condition d'arrêt			0	3	15
	80	$15\,min$: condition d'arrêt			0	4	19
	90	$15\,min$: condition d'arrêt			0	2	9
	100	$15\,min$: condition d'arrêt			0	3	11
G_3	5	$\leq 0,001$	$\leq 0,001$	0,001	0	0	0
	10	$\leq 0,001$	$\leq 0,001$	$\leq 0,001$	0	0	0
	15	$\leq 0,001$	0,005	0,017	0	0	0
	20	0,297	0,407	0,438	0	0	0
	25	14,868	18,454	19,782	0	0	0
	30	723,359	807,808	826,334	0	0	0
	35	$15\,min$: condition d'arrêt			0	4	19
	40	$15\,min$: condition d'arrêt			0	8	24
	45	$15\,min$: condition d'arrêt			0	9	25
	50	$15\,min$: condition d'arrêt			0	16	35
	60	$15\,min$: condition d'arrêt			5	23	40
	70	$15\,min$: condition d'arrêt			0	20	34
	80	$15\,min$: condition d'arrêt			11	23	42
	90	$15\,min$: condition d'arrêt			15	29	44
	100	$15\,min$: condition d'arrêt			8	22	43

Nous remarquons, d'après le tableau 7.11, que pour le groupe G_1 et bien que la condition d'arrêt est atteinte ($15min$), la méthode renvoie des solutions optimales. Cette remarque est vraie aussi pour les instances de petites tailles dans les groupes G_2 et G_3. Cependant, le gap entre la solution de la méthode énumérative et la solution optimale augmente avec le nombre de tâches à ordonnancer pour les instances du groupe G_3.

Malgré que le programme linéaire ne permet pas de résoudre la totalité des instances au bout de 15 *min*, voir tableau 7.9, nous remarquons que toutes les solutions obtenues

par le PL après la condition d'arrêt (15 *min*) sont optimales.

Le tableau 7.12, présente les performances de l'approche par séparation et évaluation. Les temps de calcul moyens ainsi que le nombre moyen de nœuds explorés y sont également donnés. Nous présentons aussi dans le même tableau la qualité de la borne supérieure initiale BS_{init} et la borne inférieure initiale BI_{init} par rapport à la solution optimale, en présentant le pire gap, le meilleur gap et le gap moyen entre les bornes BS_{init}, BI_{init} et la solution optimale. Nous rappelons que la borne supérieure initiale BS_{init} est issue des heuristiques développées dans la section 7.5.3. Les pourcentages moyens de l'utilisation des trois bornes inférieures sont indiqués dans la colonne « utilisation moyenne de BI ».

Tableau 7.12 – Les résultats de la méthode par séparation et évaluation

G	n	CPU	$\frac{BS_{init}-C^*_{max}}{C^*_{max}}$ (in %)			$\frac{C^*_{max}-BI_{init}}{C^*_{max}}$ (in %)			Utilisation moyenne de BI (%)			Nœuds
			Min	*Max*	*Moy*	*Min*	*Max*	*Moy*	$BI1$	$BI2$	$BI3$	
G_1	05	<0,001	0	0,48	0,18	0	0,14	0,03	5,38	25,38	**69,23**	12
	10	<0,001	0	0,54	0,23	0	0,06	0,01	4,80	33,05	**62,15**	29
	15	<0,001	0	0,52	0,13	0	0,01	0,00	0,00	**50,00**	**50,00**	37
	20	<0,001	0	0,40	0,14	0	0	0	0,00	**83,08**	16,92	31
	25	<0,001	0	0,28	0,06	0	0	0	0,00	**100,00**	0,00	33
	30	<0,001	0	0,28	0,07	0	0	0	0,00	**100,00**	0,00	39
	35	<0,001	0	0,3	0,05	0	0	0	0,00	**100,00**	0,00	45
	40	<0,001	0	0,29	0,09	0	0	0	0,00	**99,88**	0,13	52
	45	<0,001	0	0,25	0,04	0	0	0	0,00	**100,00**	0,00	57
	50	<0,001	0	0,23	0,06	0	0	0	0,00	**99,50**	0,50	63
	60	<0,001	0	0,22	0,04	0	0	0	0,00	**100,00**	0,00	76
	70	<0,001	0	0,19	0,03	0	0	0	0,00	**100,00**	0,00	87
	80	<0,001	0	0,22	0,04	0	0	0	0,00	**100,00**	0,00	98
	90	<0,001	0	0,18	0,04	0	0	0	0,00	**100,00**	0,00	112
	100	<0,005	0	0,12	0,01	0	0	0	0,00	**100,00**	0,00	124
G_2	05	<0,001	0	0,38	0,17	0	0,18	0,03	6,29	22,38	**71,33**	14
	10	<0,001	0	0,49	0,17	0	0,19	0,03	21,33	23,43	**55,26**	49
	15	<0,001	0	0,33	0,08	0	0,01	0	**76,97**	11,12	11,91	33
	20	<0,002	0	0,46	0,09	0	0	0	25,95	30,26	**43,79**	122
	25	<0,001	0	0,37	0,05	0	0	0	5,44	44,79	**49,77**	100
	30	<0,006	0	0,47	0,09	0	0	0	**92,68**	3,99	3,32	1304
	35	<0,001	0	0,43	0,07	0	0	0	**90,35**	7,49	2,15	870
	40	<0,001	0	0,28	0,12	0	0	0	2,59	**84,70**	12,71	67
	45	<0,001	0	0,19	0,04	0	0	0	0,00	**99,89**	0,11	65
	50	<0,001	0	0,24	0,05	0	0	0	2,78	**73,66**	23,55	94
	60	<0,001	0	0,18	0,06	0	0	0	0,00	**100,00**	0,00	86
	70	<0,002	0	0,18	0,05	0	0	0	0,00	**100,00**	0,00	100
	80	<0,001	0	0,17	0,03	0	0	0	0,00	**100,00**	0,00	114
	90	<0,001	0	0,08	0,02	0	0	0	0,00	**100,00**	0,00	127
	100	<0,002	0	0,20	0,02	0	0	0	0,00	**83,96**	16,04	161
G_3	05	<0,001	0	0,32	0,13	0	0,13	0,05	32,19	18,49	**49,32**	15,45
	10	<0,001	0	0,25	0,11	0	0,08	0,02	34,71	27,59	**37,70**	46
	15	<0,001	0	0,23	0,09	0	0,11	0,03	38,22	10,79	**50,98**	160
	20	<0,001	0	0,25	0,09	0	0,06	0,01	29,71	12,91	**57,38**	221
	25	<0,001	0	0,24	0,07	0	0	0	**87,50**	0,00	12,50	380
	30	<0,003	0	0,23	0,10	0	0,09	0,01	**90,99**	1,02	7,99	3864
	35	<0,009	0	0,22	0,05	0	0	0	11,56	14,34	**74,10**	497,1
	40	<0,001	0	0,20	0,07	0	0,20	0,07	29,39	21,80	**48,81**	313
	45	<0,001	0	0,19	0,04	0	0	0	0,00	**53,20**	46,80	137
	50	<0,004	0	0,24	0,07	0	0	0	44,08	5,80	**50,12**	1833
	60	<0,001	0	0,23	0,07	0	0	0	**93,09**	0,06	6,85	197524
	70	<0,001	0	0,14	0,04	0	0	0	0,08	**53,95**	45,97	185
	80	<0,002	0	0,25	0,03	0	0,03	0	0,00	**73,49**	26,51	73307940
	90	<0,838	0	0,17	0,05	0	0	0	**52,55**	9,42	38,03	1785
	100	<2,216	0	0,11	0,03	0	0	0	0,00	**85,09**	14,91	179

D'après le tableau 7.12, on peut conclure que l'approche séparation et évaluation est plus performante en termes de temps de calcul. La majorité des instances ont été calculées dans un délai ne dépassant pas 0.009 sec.

Les résultats figurant dans le tableau 7.12 permettent également de montrer l'efficacité des heuristiques présentées dans la section 7.5.3. En effet, les écarts moyens entre les bornes supérieures et les solutions optimales ne dépassent pas $0,54\%$. Notons aussi que dans plusieurs instances ces heuristiques ont permis de trouver la solution optimale.

Les résultats des pourcentages d'utilisation des bornes inférieures, utilisées dans les coupes de l'approche séparation et évaluation, montrent que la borne BI_2 est pratiquement la borne la plus efficiente pour les instances du groupe G_1 et légèrement moins pour G_2. Cependant, la borne inférieure BI_3 est la plus utilisée dans la résolution des instances du groupe G_3.

Nous remarquons également que les écarts relatifs entre les bornes inférieures initiales BI_{init} et les solutions optimales sont nuls dans la majorité des instances notamment pour les groupes G_1 et G_2 sauf pour les instances de petite taille. Ce qui confirme la qualité et la fiabilité des bornes inférieures présentées dans la méthode de séparation et évaluation.

D'autres paramètres sont susceptibles d'être des facteurs de complexité tels que les temps opératoires des tâches et la durée de la maintenance. Pour tester plus précisément l'effet de chaque paramètre, nous testons notre problème avec des temps opératoires plus élevés. Ensuite, nous étudions le problème avec une période de maintenance plus longue.

Nous avons testé, donc, des tâches avec des temps opératoires plus grands et qui varient entre $[1, 50]$. La méthode de résolution par séparation et évaluation, dans ce cas, est toujours efficace et la totalité des instances a été résolue d'une façon optimale.

Maintenant, nous allons considérer une période de maintenance avec une durée plus importante par rapport à celle étudiée dans les tests précédents. En effet, les paramètres α et β de la maintenance sont calculés de telle façon que la durée de cette dernière à l'instant T est égale au temps opératoire le plus élevé sur la deuxième machine ($\max_{j=1,..n} b_{2,j}$). On note cette configuration de la durée de maintenance par MP_{max} et MP_{moy} pour la configuration initiale. Des expérimentations ont été réalisées sur les instances précédentes, mais avec la configuration MP_{max}. Les tests ont montré que toutes les instances ont été résolues d'une façon optimale dans un temps de calcul maximal ne dépassant pas 5 secondes (sauf deux instances avec des temps de calcul, respectivement, 17 sec et 30 sec).

Nous présentons, dans les figures 7.26, 7.27 et 7.28, les courbes d'évolution du temps de calcul moyen pour les deux configurations MP_{max} et MP_{moy} et pour chaque groupe (G_1, G_2 et G_3).

D'après les résultats comparatifs représentés par ces figures, on peut conclure que la durée de période de maintenance n'a pas d'impact sur les deux premiers groupes (G_1 et

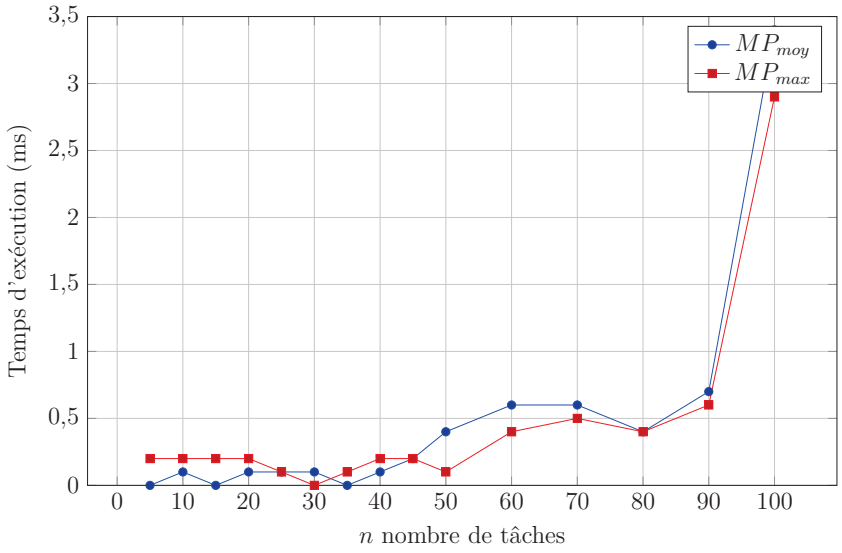

Figure 7.26 – Comparaison entre les temps d'exécution moyens (ms) des deux configurations de maintenance MP_{max} et MP_{moy} pour les instances de G_1

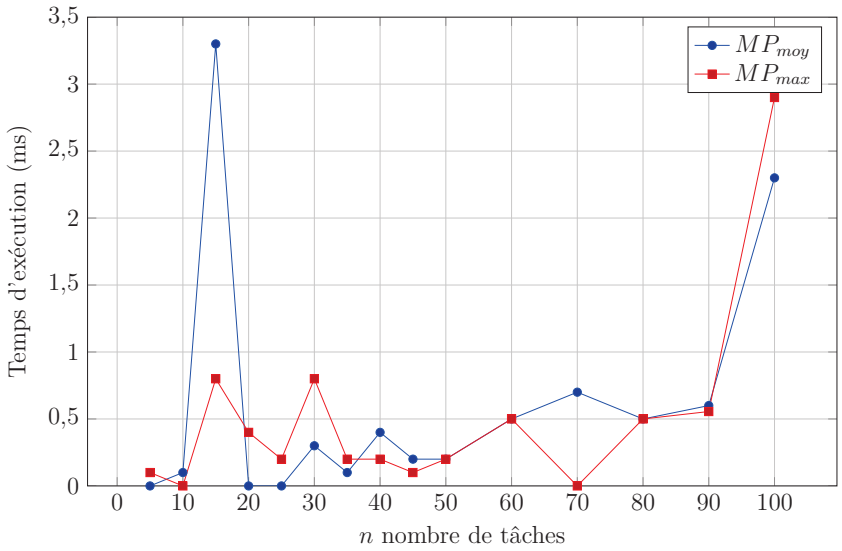

Figure 7.27 – Comparaison entre les temps d'exécution moyens (ms) des deux configurations de maintenance MP_{max} et MP_{moy} pour les instances de G_2

G_2). Cependant, le temps de calcul a augmenté dans quelques instances de grandes tailles (n supérieure à 50) du dernier groupe (G_3), mais les temps de calcul sont toujours faibles par rapport aux autres méthodes proposées et ne dépassent pas les 30 secondes dans le pire cas.

7.6 Conclusion

Dans ce chapitre, nous nous sommes intéressés au problème de flow shop avec une période de maintenance de durée variable sur la deuxième machine. Deux modèles ont

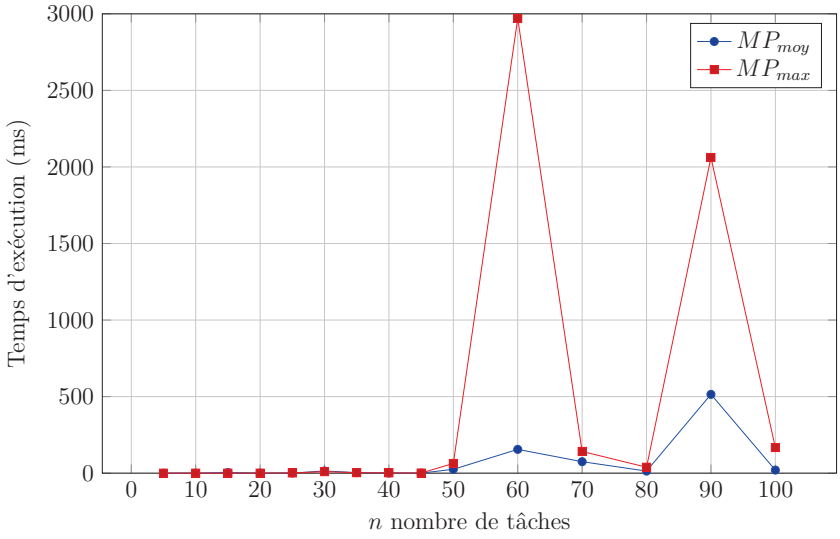

Figure 7.28 – Comparaison entre les temps d'exécution moyens (ms) des deux configurations de maintenance MP_{max} et MP_{moy} pour les instances de G_3

été étudiés, en premier lieu, nous avons étudié une période de maintenance à insérer dans l'intervalle $[0, T]$. Nous avons montré que ce problème est polynomial, des propriétés d'optimalité et un algorithme de résolution ont été proposés.

Ensuite, nous avons présenté un deuxième modèle dans lequel la maintenance doit être insérée après l'instant T. En premier lieu, la complexité du problème a été étudiée et nous avons montré que le problème est NP-difficile. L'analyse du problème a permis aussi de déterminer des propriétés d'optimalité. Nous avons proposé des méthodes de résolution approchées basées sur deux heuristiques $(H1)$ et $(H2)$.

Des méthodes de résolution exacte ont été proposées pour résoudre le problème. Nous avons développé une méthode énumérative et un programme linéaire mixte. Ces deux méthodes permettent de résoudre de façon optimale des instantes de petite taille jusqu'à 30 tâches avant la condition d'arrêt (15 min).

Une troisième méthode, plus efficace, basée sur l'approche par séparation et évaluation $(B\&B)$ a été présentée. Nous avons utilisé les propriétés d'optimalité pour déterminer trois bornes inférieures de la méthode, les deux heuristiques développées précédemment ont permis de déterminer la borne supérieure.

Une étude expérimentale des méthodes de résolution a été présentée. Cette étude nous a permis d'évaluer la performance de ces méthodes. Pour les instances de petite taille, les deux premières méthodes ont réussi à résoudre des instances de petite taille. Cependant, l'approche par séparation et évaluation est plus efficace et elle peut résoudre des instances de taille allant jusqu'à $n = 500$ tâches.

Comme perspective à ce travail, nous proposons une étude du problème lorsque plusieurs périodes de maintenance peuvent être insérées sur la machine. Cette proposition

semble plus réaliste pour des cas industriels. À plus court terme, nous pouvons améliorer la modélisation du programme linéaire en intégrant les règles de la méthode énumératives. Enfin, nous pouvons envisager une approche globale similaire à l'approche présentée dans la deuxième partie de cette thèse avec la prise en compte des variations positionnelles pour les tâches.

Conclusion générale et perspectives

Les problèmes d'ordonnancement en présence de périodes d'indisponibilité ont été introduits en début des années 90 avec des périodes connues d'avance (la date de début et la date de fin sont fixées). À partir de 2000, un modèle avec une indisponibilité, considérée comme variable de décision, a été introduit par Lee et Leon [75]. Dans ce cas, on parle d'un problème d'ordonnancement simultané de tâches et de périodes d'indisponibilité.

Outre cela, l'état des machines peut affecter les performances de production. En effet, les machines peuvent connaître des effets de variation, tels qu'un effet de détérioration ou d'apprentissage. Ainsi, l'insertion des périodes d'indisponibilité, souvent considérées comme des périodes de maintenance, permet de contrer l'effet de détérioration et de rétablir les conditions initiales de production.

Après avoir présenté les notions de base d'un problème d'ordonnancement, les différents effets appliqués sur les machines et une analyse sur l'insertion des indisponibilités dans le chapitre 1, une revue de littérature sur le sujet a été présentée dans le chapitre 2. Nous nous sommes particulièrement focalisés sur les problèmes d'ordonnancement simultanés des tâches et des périodes d'indisponibilité.

Dans la deuxième partie de la thèse (partie II), nous avons proposé une approche globale de résolution des problèmes d'ordonnancement simultané de tâches et de maintenance dans un atelier à une machine (chapitre 4) et à machines parallèles (chapitre 5). Notons que dans cette partie nous nous sommes intéressés au modèle dépendant de la position r de la tâche dans la séquence *(position-dependent model)*.

Avant d'entamer la présentation de cette approche, un modèle général a été défini dans le chapitre 4. Ce modèle est capable d'englober les modèles antérieurs de la littérature. Il comporte un nouveau système des maintenances pondérées que nous notons $\mathcal{MP}$. De plus, aucune fonction spécifique n'est imposée pour la variation des temps opératoires. Une version dépendante du groupe a aussi été présentée dans ce chapitre.

Les travaux présentés dans les chapitres (4 et 5) ont principalement consisté à résoudre le modèle global dans un atelier à une machine et à machines parallèles, sous plusieurs critères d'optimisation, tels que la minimisation de la durée totale de l'ordonnancement, la somme des dates de fin, la somme des durées d'attente, la somme des variations des dates

de fin et la somme des variations des durées d'attente. Nous avons montré en particulier que la minimisation de la durée totale de l'ordonnancement C_{max} peut être résolue avec une complexité améliorée par rapport aux autres critères. Nous avons prouvé également que cette approche reste valide pour les ordonnancements dépendants du groupe. Dans le chapitre 6, nous avons étudié des cas particuliers issus du modèle global.

L'approche de résolution présentée dans ce mémoire est basée sur les problèmes d'affectation linéaire. Nous soulignons la simplicité d'obtenir la matrice M d'affectation. Une approche unifiée pour construire la matrice M a été présentée pour résoudre le problème comme un simple problème d'affectation. Un grand nombre de différents critères d'optimisation et de modèles de maintenance peuvent être traités de cette façon, fournissant ainsi l'accès à tous ces modèles qui ont souvent été étudiés séparément dans la littérature. Les travaux élaborés dans cette partie ont permis de résoudre des problèmes d'ordonnancement non traités avant et aussi de généraliser et améliorer des résultats antérieurs.

Nous nous sommes concentrés dans le chapitre 7 sur un atelier flow shop à deux machines en présence d'une période de maintenance variable sur la deuxième machine. L'objectif est la minimisation de la durée totale de l'ordonnancement C_{max}. Notre étude a montré que le problème est polynomial si l'indisponibilité doit être insérée avant un instant T donné. Cependant, le problème est *NP-difficile* si la période de maintenance doit être insérée après l'instant T. Trois méthodes exactes (une méthode énumérative, un programme linéaire et une méthode de séparation et évaluation) ont été développées pour résoudre ce problème. L'étude expérimentale a montré que la troisième méthode est plus performante que les deux premières méthodes. La figure 7.29 représente le plan général de la thèse.

Ce travail a donné lieu à des publications dans des revues internationales ([31], [37] et [30]). Nous avons également publié nos résultats lors de conférences internationales ([34], [35] et [33]) et nationales ([32] et [36]).

Perspectives pour l'approche globale de résolution

Une hypothèse forte est faite sur les temps opératoires des tâches sont dépendants de la position r, avec $p(j,r)$ la durée de la tâche T_j dans la position r. Nous avons vu dans la littérature qu'un deuxième modèle intéressant existe, dont la durée de la tâche est en fonction de sa date de début t. Il serait donc intéressant d'étendre notre approche globale en intégrant le modèle temporel $p(j,t)$. En se basant sur le fait que la date de début t d'une tâche peut être exprimée en fonction des temps opératoires des tâches exécutées avant la tâche, *i.e.* $t = \sum_{s=1}^{j-1} p_{[s]}$, nous pouvons généraliser notre approche au modèle temporel.

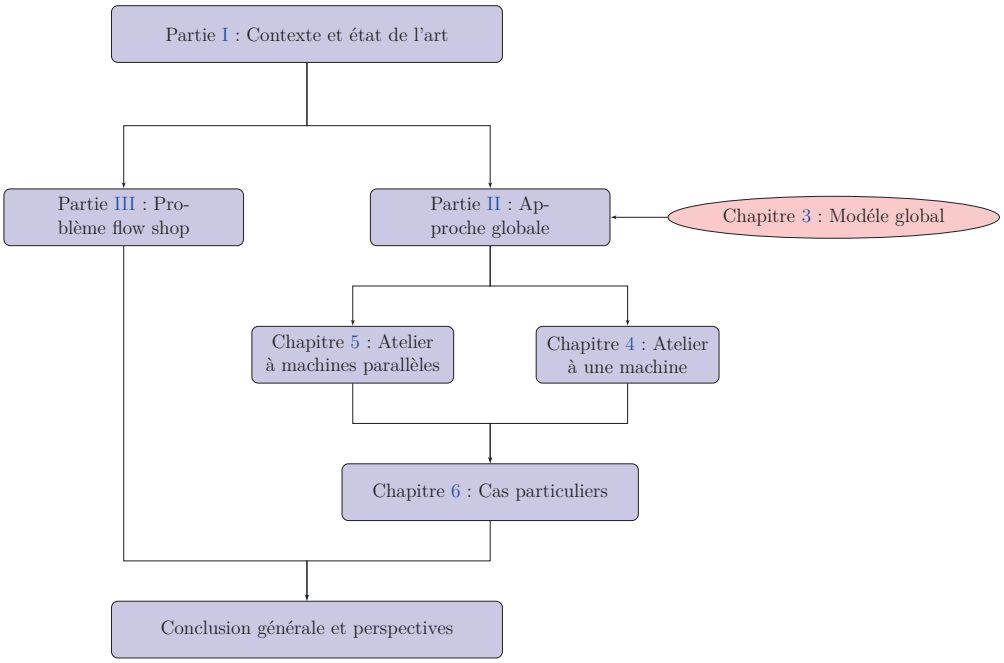

Figure 7.29 – Plan général de la thèse

Pour aller plus loin, nous proposons aussi d'étendre notre approche sur des temps opératoires plus généraux qui varient à la fois en fonction de la position r et du temps t, avec $p(j, r, t)$ le temps opératoire de la tâche T_j ordonnancée dans la position r à l'instant t. Il nous semble donc intéressant d'étudier ce modèle.

Nous avons supposé que les ressources des maintenances sont illimitées. En pratique, cette hypothèse n'est pas toujours vraie. Il serait donc intéressant d'étudier le problème en tenant en compte la limitation des ressources. Par exemple, dans le cas d'un atelier de machines parallèles, nous pouvons étudier notre problème avec une interdiction de chevauchement des maintenances, car une seule équipe de maintenance est disponible.

Perspectives pour le problème flow shop

Dans la troisième partie, nous nous sommes limités à la résolution du problème flow shop à deux machines avec une seule période de maintenance sur la deuxième machine. Néanmoins, les problèmes avec plusieurs périodes de maintenance sont plus proches de la réalité industrielle. Ainsi, la prise en compte de plusieurs périodes de maintenance constitue une perspective intéressante à envisager comme suite à ce travail.

Dans cette partie, aucun effet de variation n'est appliqué sur les machines. Évidemment, en pratique, les machines connaissent des effets de variation (effet d'apprentissage / effet de détérioration). Dans ce cas, nous proposons d'étudier le problème en présence d'un effet de détérioration ou d'apprentissage.

Nous proposons aussi de réétudier les méthodes présentées dans cette partie (la méthode énumérative, le programme linéaire et l'approche de séparation et évaluation) avec une période de maintenance sur la première machine ou sur les deux machines.

Comme perspectives générales à ce travail, nous en citons ici trois qui nous semblent importantes :

- une étude qui prend en compte une fenêtre de réalisation pour la maintenance. Notons que les problèmes avec des fenêtres de réalisation ont été étudiés dans le cadre des maintenances flexibles, mais avec un modèle très restreint, cf. Aggoune et *al.* [1]. On pourrait étudier le modèle présenté dans cette thèse dans un environnement de réalisation flexible.
- le modèle général proposé dans cette thèse pourrait être étendu, en considérant d'autre système de maintenance et/ou de variations des tâches.
- une étude du modèle global, présenté dans cette thèse, avec d'autres critères d'optimisation tels que la minimisation des retards.

Références bibliographiques

[1] R. Aggoune. *Ordonnancement d'ateliers sous contraintes de disponibilité des machines*. Thèse de doctorat, Université de Metz, 2002. 180

[2] B. Alidaee et N.K. Womer. Scheduling with time dependent processing times : Review and extensions. *Journal of the Operational Research Society*, 50(7):711–720, 1999. 16

[3] H. Allaoui, S. Lamouri, A. Artiba et E. Aghezzaf. Simultaneously scheduling n jobs and the preventive maintenance on the two-machine flow shop to minimize the makespan. *International Journal of Production Economics*, 112(1):161 – 167, 2008. 14, 30, 39, 140, 145, 146

[4] F. Angel-Bello, A. Alvarez, J. Pacheco et I. Martinez. A heuristic approach for a scheduling problem with periodic maintenance and sequence-dependent setup times. *Computers & Mathematics with Applications*, 61(4):797 – 808, 2011. 31

[5] A. Bachman et A. Janiak. Minimization of the maximum completion time for the deteriorating jobs with ready times. *Scientific Bulletins of Silesian Polytechnics, S. Automatics*, 123:33–44, 1998. 17

[6] A. Bachman et A. Janiak. Minimizing maximum lateness under linear deterioration. *European Journal of Operational Research*, 126(3):557 – 566, 2000. 17

[7] A. Bachman et A. Janiak. Scheduling jobs with position-dependent processing times. *Journal of the Operational Research Society*, 55:257 – 264, 2004. 16, 19, 58, 67

[8] A. Bachman, A. Janiak et M. Y. Kovalyov. Minimizing the total weighted completion time of deteriorating jobs. *Information Processing Letters*, 81(2):81 – 84, 2002. 17

[9] U. Bagchi. Simultaneous minimization of mean and variation of flow time and waiting time in single machine systems. *Operations Research*, 37(1):118–125, 1989. 13, 199

[10] K. R. Baker et G. D. Scudder. Sequencing with earliness and tardiness penalties : A review. *Operations Research*, 38(1):22–36, 1990. 13

[11] D. Biskup. Single-machine scheduling with learning considerations. *European Journal of Operational Research*, 115(1):173 – 178, 1999. 18, 19, 58, 67

[12] D. Biskup. A state-of-the-art review on scheduling with learning effects. *European Journal of Operational Research*, 188(2):315 – 329, 2008. 18

[13] A. Bosio et G. Righini. A dynamic programming algorithm for the single-machine scheduling problem with deteriorating processing times. *Electronic Notes in Discrete Mathematics*, 25:139 – 142, 2006. CTW2006 - Cologne-Twente Workshop on Graphs and Combinatorial Optimization. 19

[14] S. Browne et U. Yechiali. Scheduling deteriorating jobs on a single processor. *Operations Research*, 38(3):495–498, 1990. 16, 17

[15] R. Burkard, M. Dell'Amico et S. Martello. *Assignment Problems*. Society for Industrial and Applied Mathematics, 2009. 66

[16] J. Carlier et P. Chrétienne. *Problémes d'ordonnancement*. Masson, Paris, 1988. 8

[17] J.-S. Chen. Scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan. *European Journal of Operational Research*, 190(1):90 – 102, 2008. 31

[18] M.-B. Cheng et S.-J. Sun. Two scheduling problems in group technology with deteriorating jobs. *Applied Mathematics-a Journal of Chinese Universities*, 20(2): 102–111, 2005. 19

[19] T.C.E. Cheng et Q. Ding. Scheduling start time dependent tasks with deadlines and identical initial processing times on a single machine. *Computers & Operations Research*, 30(1):51 – 62, 2003. 19

[20] T.C.E. Cheng, Q Ding et B.M.T. Lin. A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research*, 152(1):1 – 13, 2004. 16

[21] T.C.E Cheng, C.-J. Hsu et D.-L. Yang. Unrelated parallel-machine scheduling with deteriorating maintenance activities. *Computers & Industrial Engineering*, 60(4):602 – 605, 2011. 49, 50, 52, 117

[22] T.C.E. Cheng, W.-H. Kuo et D.-L. Yang. Scheduling with a position-weighted learning effect based on sum-of-logarithm-processing-times and job position. *Information Sciences*, 221(0):490 – 500, 2013. 19

[23] T.C.E. Cheng, C.-C. Wu et W.-C. Lee. Some scheduling problems with sum-of-processing-times-based and job-position-based learning effects. *Information Sciences*, 178(11):2476 – 2487, 2008. 19

[24] T.C.E Cheng, S.-J. Yang et D.-L. Yang. Common due-window assignment and scheduling of linear time-dependent deteriorating jobs and a deteriorating maintenance activity. *International Journal of Production Economics*, 135(1):154 – 161, 2012. 46, 48

[25] E. Dinic. On solution of two assignment problems. *Studies in Discrete Optimization, A.A. Fridman ed., 'Nauka', Moscow*, pages 33–348, 1976. , In Russian. 66, 111, 124

[26] E. Dinic et M.A. Kronrod. An algorithm for the solution of the assignment problem. *Soviet Math. Dokl*, 10(6):1324–1326, 1969. 65, 91

[27] D. Dolev et M.K. Warmuth. Profile scheduling of opposing forests and level orders. *SIAM Journal on Algebraic Discrete Methods*, 6(4):665–687, 1985. 28

[28] D. Dolev et M.K. Warmuth. Scheduling flat graphs. *SIAM Journal on Computing*, 14(3):638–657, 1985. 28

[29] G. Finke, A. Gara-Ali, M.-L. Espinouse, V. Jost et J. Moncel. Single-machine production and maintenance : a unified approach. *Cahiers Leibniz*, (205), 2013. 70, 202

[30] G. Finke, A. Gara-Ali, M.-L. Espinouse, V. Jost et J. Moncel. Unified matrix approach to solve production-maintenance problems on a single machine. *Omega*, 2016. 70, 178, 201

[31] A. Gara-Ali et M.-L. Espinouse. Erratum to :"simultaneously scheduling n jobs and the preventive maintenance on the two-machine flow shop to minimize the makespan"[int. j. prod. econ. 112 (2008) 161-167]. *International Journal of Production Economics*, 153:361–363, 2014. 39, 138, 146, 178, 201

[32] A. Gara-Ali et M.-L. Espinouse. Méthodes exactes pour la résolution du problème Flow-shop à deux machines avec des contraintes de disponibilité sur la deuxième machine. *In 15ème congrès annuel de la Société française de recherche opérationnelle et d'aide à la décision ROADEF*, Bordeaux, France, 2014. 138, 178, 202

[33] A. Gara-Ali et M. L. Espinouse. A two-machine flow-shop scheduling with a deteriorating maintenance activity on the second machine. *In International Conference on Industrial Engineering and Systems Management IESM*, pages 481–488, 2015. 138, 178, 201

[34] A. Gara-Ali, M.-L. Espinouse et G. Finke. Unrelated parallel-machine scheduling with deterioration effects and multi-maintenance activities. *In 44th International Conference on Computers and Industrial Engineering CIE-44 and IMSS*, pages 1080–1089, Istanbul, Turkey, 2014. 96, 116, 178, 202

[35] A. Gara-Ali, M.-L. Espinouse et G. Finke. Unrelated parallel-machine scheduling with optional maintenance activity on each machine and general effects of deterioration. *In 10th International Conference on Modeling, Optimization & SIMulation MOSIM*, Nancy, France, 2014. 116, 178, 202

[36] A. Gara-Ali, M.-L. Espinouse et G. Finke. Problèmes d'ordonnancement sur des machines parallèles avec des périodes de maintenance et un effet de détérioration : Minimisation de la charge totale des machines. *In 15ème congrès annuel de*

la Société française de recherche opérationnelle et d'aide à la décision ROADEF, Marseille, France, 2015. 96, 116, 178, 202

[37] A. Gara-Ali, G. Finke et M.-L. Espinouse. Parallel-machine scheduling with maintenance : Praising the assignment problem. *European Journal of Operational Research*, 252:90 – 97, 2016. 96, 116, 178, 201

[38] S. Gawiejnowicz. Scheduling deteriorating jobs subject to job or machine availability constraints. *European Journal of Operational Research*, 180(1):472 – 478, 2007. 19

[39] S. Gawiejnowicz. *Time-Dependent Scheduling*. Springer Publishing Company, Incorporated, 1 édition, 2008. 16

[40] S. Gawiejnowicz et L. Pankowska. Scheduling jobs with varying processing times. *Information Processing Letters*, 54(3):175 – 178, 1995. 19

[41] V. Gordon, C.N. Potts, V.A. Strusevich et J.D. Whitehead. Single machine scheduling models with deterioration and learning : handling precedence constraints via priorty generation. *Journal of Scheduling*, 11(5):357–370, 2008. 19, 58, 67

[42] V. Gordon, J.-M. Proth et C. Chu. A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, 139(1):1 – 25, 2002. 13

[43] V. Gordon, J.-M Proth et C. Chu. A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, 139(1):1 – 25, 2002. 46

[44] R.L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966. 9

[45] R.L. Graham, E.L. Lawler, J.K. Lenstra et A.H.G.R. Kan. Optimization and approximation in deterministic sequencing and scheduling : a survey. volume 5 de *Annals of Discrete Mathematics*, pages 287 – 326. Elsevier, 1979. 11, 141

[46] A. Grigoriev, J.V.D. Klundert et F.C.R. Spieksma. Modeling and solving the periodic maintenance problem. *European Journal of Operational Research*, 172(3): 783 – 797, 2006. 30, 31

[47] A.-X. Guo et J.-B. Wang. Single machine scheduling with deteriorating jobs under the group technology assumption. *International Journal of Pure and Applied Mathematics*, 18(2):225–31, 2005. 19

[48] J.N.D. Gupta et S.K. Gupta. Single facility scheduling with nonlinear processing times. *Computers & Industrial Engineering*, 14(4):387 – 393, 1988. 16, 19

[49] S. K. Gupta, A. S Kunnathur et K. Dandapani. Optimal repayment policies for multiple loans. *Omega*, 15(4):323 – 330, 1987. 17

[50] G.H. Hardy, J.E. Littlewood et G. Polya. *Inequalities*. London, Cambridge Mathematical Library, 1 édition, 1934. 129

[51] Y. He, M. Ji et T. C. E. Cheng. Single machine scheduling with a restricted rate-modifying activity. *Naval Research Logistics (NRL)*, 52(4):361–369, 2005. 30, 32, 35, 36

[52] Y. He et L. Sun. One-machine scheduling problems with deteriorating jobs and position-dependent learning effects under group technology considerations. *International Journal of Systems Science*, 46(7):1319–1326, 2015. 63

[53] F. Hnaien, F. Yalaoui et A. Mhadhbi. Makespan minimization on a two-machine flowshop with an availability constraint on the first machine. *International Journal of Production Economics*, 164:95 – 104, 2015. 140, 161

[54] K. I-J. Ho, J. Y-T. Leung et W-D. Wei. Complexity of scheduling tasks with time-dependent execution times. *Information Processing Letters*, 48(6):315 – 320, 1993. 18

[55] C.-J. Hsu, T.C.E. Cheng et D.-L. Yang. Unrelated parallel-machine scheduling with rate-modifying activities to minimize the total completion time. *Information Sciences*, 181(20):4799 – 4803, 2011. 30, 37, 40, 117

[56] C.-J. Hsu, M. Ji, J.-Y. Guo et D.-L. Yang. Unrelated parallel-machine scheduling problems with aging effects and deteriorating maintenance activities. *Information Sciences*, 253(0):163 – 169, 2013. 19, 49, 50, 52, 58, 67, 117, 118, 119, 121

[57] X. Huang, M.-Z. Wang et P. Ji. Parallel machines scheduling with deteriorating and learning effects. *Optimization Letters*, 8(2):493–500, 2014. 20

[58] X. Huang, M.-Z. Wang et J.-B. Wang. Single-machine group scheduling with both learning effects and deteriorating jobs. *Computers & Industrial Engineering*, 60(4):750 – 754, 2011. 63

[59] M. Ji et T.C.E Cheng. Scheduling with job-dependent learning effects and multiple rate-modifying activities. *Information Processing Letters*, 110(11):460 – 463, 2010. 38, 40, 67, 107, 108

[60] M. Ji, Y. He et T.C.E. Cheng. Scheduling linear deteriorating jobs with an availability constraint on a single machine. *Theoretical Computer Science*, 362(1 – 3):115 – 126, 2006. 19

[61] M. Ji, X. Tang, X. Zhang et T.C.E. Cheng. Machine scheduling with deteriorating jobs and dejong's learning effect. *Computers & Industrial Engineering*, 91:42 – 47, 2016. 19, 20

[62] M. Ji, D. Yao, Q. Yang et T.C.E Cheng. Machine scheduling with dejong's learning effect. *Computers & Industrial Engineering*, 80(0):195 – 200, 2015. 18

[63] S. M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1(1):61–68, 1954. 142, 148

[64] A.H.G. Rinnooy Kan. *Machine scheduling problems : classification, complexity and computations*. Springer Science & Business Media, 2012. 16

[65] J. J. Kanet. Minimizing variation of flow time in single machine systems. *Management Science*, 27(12):1453–1459, 1981. 13, 197

[66] A. Kononov. Single machine scheduling problems with processing times proportional to an arbitrary function. *Discrete Analysis and Operations Research*, 5(3):17–37, 1998. in Russian. 19

[67] M.A. Kubzin et V.A. Strusevich. Two-machine flow shop no-wait scheduling with machine maintenance. *4OR*, 3(4):303–313, 2005. 24, 29, 30, 41, 53, 64, 140

[68] M.A. Kubzin et V.A. Strusevich. Planning machine maintenance in two-machine shop scheduling. *Operations Research*, 54(4):789–800, 2006. 24, 30, 41, 53, 64, 140, 141

[69] W.-H. Kuo et D.-L. Yang. Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect. *European Journal of Operational Research*, 174(2):1184 – 1190, 2006. 18

[70] W.-H. Kuo et D.-L. Yang. Single-machine scheduling problems with start-time dependent processing time. *Computers & Mathematics with Applications*, 53(11): 1658 – 1664, 2007. 19

[71] W.-H. Kuo et D.-L. Yang. Minimizing the makespan in a single-machine scheduling problem with the cyclic process of an aging effect. *The Journal of the Operational Research Society*, 59(3):416–420, 2008. 30, 33, 36, 67, 121

[72] C.-Y. Lee. Machine scheduling with an availability constraint. *Journal of Global Optimization*, 9(3-4):395–416, 1996. 20, 29, 30

[73] C.-Y. Lee. Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. *Operations Research Letters*, 20(3):129 – 139, 1997. xiv, 20, 29, 30, 139, 140, 153

[74] C.-Y. Lee et Z.-L. Chen. Scheduling jobs and maintenance activities on parallel machines. *Naval Research Logistics (NRL)*, 47(2):145–165, 2000. 36, 37, 40

[75] C.-Y. Lee et V.J. Leon. Machine scheduling with a rate-modifying activity. *European Journal of Operational Research*, 128(1):119 – 128, 2001. 20, 29, 30, 31, 32, 35, 36, 37, 119, 177

[76] W.-C. Lee et Z.-S. Lu. Group scheduling with deteriorating jobs to minimize the total weighted number of late jobs. *Applied Mathematics and Computation*, 218(17):8750 – 8757, 2012. 63

[77] Z. Liu et E. Sanlaville. Preemptive scheduling with variable profile, precedence constraints and due dates. *Discrete Applied Mathematics*, 58(3):253 – 280, 1995. 29

[78] E.J. Lodree et C. D. Geiger. A note on the optimal sequence position for a rate-modifying activity under simple linear deterioration. *European Journal of Operational Research*, 201(2):644 – 648, 2010. 34, 36

[79] C. Low et W.-Y. Lin. Minimizing the total completion time in a single-machine scheduling problem with a learning effect. *Applied Mathematical Modelling*, 35(4): 1946 – 1951, 2011. 19

[80] Y.-Y Lu, C.-M. Wei et J.-B. Wang. Several single-machine scheduling problems with general learning effects. *Applied Mathematical Modelling*, 36(11):5650 – 5656, 2012. 19

[81] Y. Ma, C. Chu et C. Zuo. A survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering*, 58(2):199 – 211, 2010. 1, 14, 29, 31

[82] N. Milo, S. Zakov, E. Katzenelson, E. Bachmat, Y. Dinitz et M. Ziv-Ukelson. Unrooted unordered homeomorphic subtree alignment of rna trees. *Algorithms for Molecular Biology*, 8(1):13, 2013. 66, 87, 111

[83] J. Moncel, G. Finke et V. Jost. A note on single-machine scheduling problems with position-dependent processing times. *In 13th International Conference on Project Management and Scheduling PMS*, pages 231–234, Leuven, Belgium, 2012. 129

[84] B. Mor et G. Mosheiov. Scheduling a maintenance activity and due-window assignment based on common flow allowance. *International Journal of Production Economics*, 135(1):222 – 230, 2012. xiii, 47, 48

[85] G. Mosheiov. Scheduling jobs under simple linear deterioration. *Computers & Operations Research*, 21(6):653 – 659, 1994. 19

[86] G. Mosheiov. Parallel machine scheduling with a learning effect. *Journal of the Operational Research Society*, 52(10):1165–1169, 2001. 81

[87] G. Mosheiov. Scheduling problems with a learning effect. *European Journal of Operational Research*, 132(3):687–693, 2001. 19, 58, 67

[88] G. Mosheiov. A note on scheduling deteriorating jobs. *Mathematical and Computer Modelling*, 41(8-9):883–886, 2005. 19, 58, 67

[89] G. Mosheiov et J. B. Sidney. Scheduling with general job-dependent learning curves. *European Journal of Operational Research*, 147(3):665 – 670, 2003. 19, 58, 67

[90] G. Mosheiov et J. B. Sidney. Scheduling a deteriorating maintenance activity on a single machine. *J Oper Res Soc*, 61(5):882–887, May 2010. 20, 41, 43, 45, 46, 47, 48, 119

[91] J.-L. Mott, A. Kandel et T.-P. Baker. *Discrete Mathematics for Computer Scientists & Mathematicians*. Prentice-Hall, Inc., 1986. 82, 103

[92] C.-T. Ng, T.-C.-E. Cheng, A. Bachman et A. Janiak. Three scheduling problems with deteriorating jobs to minimize the total completion time. *Information Processing Letters*, 81(6):327 – 333, 2002. 17

[93] D. Okolowski et S. Gawiejnowicz. Exact and heuristic algorithms for parallel-machine scheduling with dejong's learning effect. *Computers & Industrial Engineering*, 59(2):272 – 279, 2010. 19

[94] M.L. Pinedo. *Scheduling : theory, algorithms, and systems*. Springer Science & Business Media, 2012. 159

[95] X. Qi, T. Chen et F. Tu. Scheduling the maintenance on a single machine. *The Journal of the Operational Research Society*, 50(10):1071–1078, 1999. 29, 31, 34, 36

[96] N.-P. Rachaniotis et C.-P. Pappis. Scheduling fire-fighting tasks using the concept of "deteriorating jobs". *Canadian Journal of Forest Research*, 36(3):652–658, 2006. 17

[97] A. Rudek et R. Rudek. Makespan minimization flowshop with position dependent job processing times-computational complexity and solution algorithms. *Computers & Operations Research*, 40(8):2071–2082, 2013. 16

[98] K. Rustogi et V.-A. Strusevich. Simple matching vs linear assignment in scheduling models with positional effects : A critical review. *European Journal of Operational Research*, 222(3):393 – 407, 2012. 19, 64, 67, 93, 107, 108, 133

[99] K. Rustogi et V.-A. Strusevich. Single machine scheduling with general positional deterioration and rate-modifying maintenance. *Omega*, 40(6):791 – 804, 2012. xiii, xiv, 19, 34, 36, 42, 43, 64, 67, 92, 93

[100] E. Sanlaville et G. Schmidt. Machine scheduling with availability constraints. *Acta Informatica*, 35(9):795–811, 1998. 31

[101] M. Sbihi et C. Varnier. Single-machine scheduling with periodic and flexible periodic maintenance to minimize maximum tardiness. *Computers & Industrial Engineering*, 55(4):830 – 840, 2008. 31

[102] G. Schmidt. Scheduling with limited machine availability. *European Journal of Operational Research*, 121(1):1 – 15, 2000. 1, 14, 29, 31

[103] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998. 65

[104] R. Singh et S. Arvinderjit. Maintenance planning and control of hand tools unit in india : a case study. *International Journal of Indian Culture and Business Management*, 7(1):109–132, 2013. 1

[105] V. Tanaev, W. Gordon et Y.-M. Shafransky. *Scheduling theory. Single-stage systems*, volume 284. Springer Science & Business Media, 2012. 19

[106] M.-D. Toksari et E. Guner. The common due-date early/tardy scheduling problem on a parallelmachine under the effects of time-dependent learning and linear and nonlinear deterioration. *Expert Systems with Applications*, 37(1):92 – 112, 2010. 20

[107] D.-L. Yang W.-H. Kuo. A note on due-date assignment and single-machine scheduling with deteriorating jobs and learning effects. *The Journal of the Operational Research Society*, 62(1):206–210, 2011. 20

[108] J.-B. Wang. Single machine scheduling with a time-dependent learning effect and deteriorating jobs. *Journal of the Operational Research Society*, 60(4):583–586, 2009. 20

[109] J.-B. Wang. Single-machine scheduling with learning effect and deteriorating jobs. *Computers & Industrial Engineering*, 57(4):1452 – 1456, 2009. 20

[110] J.-B. Wang, A.-X. Guo, F. Shan, B. Jiang et L.-Y. Wang. Single machine group scheduling under decreasing linear deterioration. *Journal of Applied Mathematics and Computing*, 24(1-2):283–293, 2007. 19

[111] J.-B. Wang, X. Huang, X.-Y. Wang, N. Yin et L.-Y. Wang. Learning effect and deteriorating jobs in the single machine scheduling problems. *Applied Mathematical Modelling*, 33(10):3848 – 3853, 2009. 20

[112] J.-B. Wang, L. Lin et F. Shan. Single-machine group scheduling problems with deteriorating jobs. *The International Journal of Advanced Manufacturing Technology*, 39(7-8):808–812, 2008. 19

[113] J.-B. Wang et J.-J. Wang. Scheduling jobs with a general learning effect model. *Applied Mathematical Modelling*, 37(4):2364 – 2373, 2013. 19

[114] J.-B. Wang et C.-M. Wei. Parallel machine scheduling with a deteriorating maintenance activity and total absolute differences penalties. *Applied Mathematics and Computation*, 217(20):8093 – 8099, 2011. 51, 117

[115] J.-B. Wang et Z.-Q. Xia. Scheduling jobs under decreasing linear deterioration. *Information Processing Letters*, 94(2):63 – 69, 2005. 19

[116] J.-J. Wang, J-B. Wang et F. Liu. Parallel machines scheduling with a deteriorating maintenance activity. *Journal of the Operational Research Society*, 62(10):1898 – 1902, 2011. 49, 52, 117, 119

[117] L.-Y. Wang, X. Huang, P. Ji et E.-M. Feng. Unrelated parallel-machine scheduling with deteriorating maintenance activities to minimize the total completion time. *Optimization Letters*, 8(1):129–134, 2014. 50, 52, 117

[118] X.-Y. Wang et J.-J. Wang. Scheduling deteriorating jobs with a learning effect on unrelated parallel machines. *Applied Mathematical Modelling*, 38(21-22):5231 – 5238, 2014. 20

[119] G. Wei, Y. Qiu et M. Ji. Scheduling with position-based deteriorating jobs and multiple deteriorating rate-modifying activities. *Asia-Pacific Journal of Operational Research*, 31(01):1450009, 2014. 83, 84

[120] C.-C. Wu, Y.-R. Shiau et W.-C. Lee. Single-machine group scheduling problems with deterioration consideration. *Computers & Operations Research*, 35(5):1652 – 1659, 2008. Part Special Issue : Algorithms and Computational Methods in Feasibility and Infeasibility. 19

[121] Y.-B. Wu, M.-Z. Wang et J.-B. Wang. Some single-machine scheduling with both learning and deterioration effects. *Applied Mathematical Modelling*, 35(8):3731 – 3736, 2011. 20

[122] P. Xue, Y. Zhang et X. Yu. Single-machine scheduling with piece-rate maintenance and interval constrained position-dependent processing times. *Applied Mathematics and Computation*, 226(0):415 – 422, 2014. 47, 48

[123] D.-L. Yang, T.-C.-E. Cheng, S.-J. Yang et C.-J. Hsu. Unrelated parallel-machine scheduling with aging effects and multi-maintenance activities. *Computers & Operations Research*, 39(7):1458 – 1464, 2012. 17, 19, 39, 40, 67, 112, 113, 133

[124] D.-L. Yang et W.-H. Kuo. Some scheduling problems with deteriorating jobs and learning effects. *Computers & Industrial Engineering*, 58(1):25 – 28, 2010. 20

[125] D.-L. Yang et S.-J. Yang. Unrelated parallel-machine scheduling problems with multiple rate-modifying activities. *Information Sciences*, 235(0):280 – 286, 2013. 30, 38, 40

[126] S.-J. Yang. Single-machine scheduling problems with both start-time dependent learning and position dependent aging effects under deteriorating maintenance consideration. *Applied Mathematics and Computation*, 217(7):3321 – 3329, 2010. 14, 20, 30, 42, 43, 44, 45, 48

[127] S.-J. Yang. Group scheduling problems with simultaneous considerations of learning and deterioration effects on a single-machine. *Applied Mathematical Modelling*, 35(8):4008 – 4016, 2011. 63

[128] S.-J. Yang. Single-machine scheduling problems simultaneously with deterioration and learning effects under deteriorating multi-maintenance activities consideration. *Computers & Industrial Engineering*, 62(1):271 – 275, 2012. 42, 43, 45

[129] S.-J. Yang. Unrelated parallel-machine scheduling with deterioration effects and deteriorating multi-maintenance activities for minimizing the total completion time. *Applied Mathematical Modelling*, 37(5):2995 – 3005, 2013. 50, 52, 67, 104, 105

[130] S.-J. Yang, C.-J. Hsu et D.-L. Yang. Note on "unrelated parallel-machine scheduling with deteriorating maintenance activities". *Computers & Industrial Engineering*, 62(4):1141 – 1143, 2012. 50, 52

[131] S.-J. Yang, C.-J. Hsu et D.-L. Yang. Single-machine scheduling and slack due-date assignment with aging effect and deteriorating maintenance. *Optimization Letters*, 6(8):1855–1873, 2012. 46, 47, 48

[132] S.-J. Yang, C.-J. Hsu et D.-L. Yang. Note on "unrelated parallel-machine scheduling with rate-modifying activities to minimize the total completion time". *Information Sciences*, 260(0):215 – 217, 2014. 37, 38, 40, 67, 117

[133] S.-J. Yang, C.-J. Hsu, D.-L. Yang et T.-C.-E. Cheng. Single-machine due-window assignment and scheduling with job-dependent aging effects and deteriorating maintenance. *Computers & Operations Research*, 37(8):1510 – 1514, 2010. 47, 48

[134] S.-J. Yang et D.-L. Yang. Minimizing the makespan on single-machine scheduling with aging effect and variable maintenance activities. *Omega*, 38(6):528 – 533, 2010. 14, 19, 23, 30, 41, 43, 58, 67, 89, 133

[135] S.-J. Yang et D.-L. Yang. Minimizing the total completion time in single-machine scheduling with aging/deteriorating effects and deteriorating maintenance activities. *Computers & Mathematics with Applications*, 60(7):2161 – 2169, 2010. xiii, 19, 44, 45, 58, 67, 83, 84

[136] S.-J. Yang et D.-L. Yang. Note on a unique integer mathematical model for scheduling deteriorating jobs with rate-modifying activities on a single machine. *The International Journal of Advanced Manufacturing Technology*, 64(9-12):1759–1764, 2013. 19, 58, 67

[137] Y. Yin, D. Xu et J. Wang. Single-machine scheduling with a general sum-of-actual-processing-times-based and job-position-based learning effect. *Applied Mathematical Modelling*, 34(11):3623 – 3630, 2010. 19

[138] C.-L. Zhao et H.-Y. Tang. Single machine scheduling with general job-dependent aging effect and maintenance activities to minimize makespan. *Applied Mathematical Modelling*, 34(3):837 – 841, 2010. 19, 30, 33, 36, 58, 67, 89

[139] C.-L. Zhao, H.-Y. Tang et C.-D. Cheng. Two-parallel machines scheduling with rate-modifying activities to minimize total completion time. *European Journal of Operational Research*, 198(1):354 – 357, 2009. 30, 37, 40, 119

Annexe A

A Glossaire des notations utilisées

C_j	date d'achèvement de la tâche T_j
$C_{[r]}$	date d'achèvement de la tâche ordonnancée en position r
G_i^l	groupe des tâches entre l'indisponibilité $i-1$ et i sur la machine M_l
k_l	nombre d'indisponibilités sur la machine M_l
$k = \sum_{l=1}^{m} k_l$	nombre d'indisponibilités sur l'ensemble des machines
k_0	borne supérieure du nombre d'indisponibilités
m	nombre de machines
m_{li}	durée de la période d'indisponibilité i sur la machine M_l
n	nombre de tâches
$p(l,j,r)$	temps opératoire de la tâche T_j à la position r sur M_l
p_j	durée intrinsèque de la tâche T_j
r	position de la tâche dans la séquence de l'ordonnancement
t_{li}	durée d'exécution séparant deux indisponibilités $i-1$ et i sur M_l
$TADC$	variation des dates de fin
$TADW$	variation des durées d'attente
TC	somme des dates de fin
TML	charge totale des machines ($\sum_{l=1}^{m} C_{max}^l$)
TW	somme des durées d'attente
$V_j(l,i,r)$	coût d'affectation de la tâche j à la position (machine M_l, groupe i, rang r)
W_j	durée d'attente de la tâche T_j
$W_{[r]}$	durée d'attente de la tâche ordonnancée en position r
$\alpha > 0$	facteur de détérioration de la période d'indisponibilité
$\alpha_{li} > 0$	facteur de détérioration de la $i^{\text{ème}}$ indisponibilité sur M_l
$\beta \geq 0$	durée standard de la période d'indisponibilité
$\beta_{li} \geq 0$	durée standard de la $i^{\text{ème}}$ indisponibilité sur M_l

Annexe B

B Les critères d'optimisation

B.1 Minimisation de la somme des dates de fin

La minimisation de la somme des dates de fin est donnée par la formule :

$$\sum C_j = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} C_{[i,r]} = \sum_{i=1}^{k+1} \left(C_{[i,1]} + C_{[i,2]} + \ldots + C_{[i,n_i]} \right)$$

Ce critère est défini en fonction des dates de fin de chaque tâche. Tout d'abord, nous considérons le problème avec des périodes de maintenance de durées nulles ($m_i = 0$). Dans ce cas, il est clair que la somme des dates de fin est égale à :

$$\sum C_j = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} \left(n - \sum_{h=0}^{i-1} n_h - r + 1 \right) p([i,r],r) = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} (n - s_{ir} + 1)\, p([i,r],r) \quad (7.1)$$

où $s_{ir} = r + \sum_{h=0}^{i-1} n_h$, avec $n_0 = 0$, s_{ir} représente le rang global de la tâche sur la machine.

Pour notre problème, où les maintenances suivent le modèle $\mathcal{MP}$, les dates de fin des tâches ordonnancées dans un groupe G_i sont augmentées par les durées des périodes de maintenance qui précèdent G_i (les périodes de maintenance $m_1, m_2, \ldots, m_{i-1}$), voir figure B.1. Par conséquent, la portion de la maintenance, associée à la tâche T_j ordonnancée au rang r du groupe G_i, contribue aux dates de fin de toutes les tâches qui se trouvent dans les groupes suivants par une valeur $w(j,r)p(j,r)$. Le nombre de tâches dans les groupes qui suivent G_i est : $\sum_{h=i+1}^{k+1} n_h$. Ainsi, la contribution totale de la portion de la maintenance, liée à la tâche T_j insérée dans le groupe G_i, est égale à $(\sum_{h=i+1}^{k+1} n_h) w(j,r)p(j,r)$. De même, la durée élémentaire β_i de la $i^{\text{ème}}$ période de maintenance contribue à la somme des dates de fin ($\sum C_j$) par une valeur égale à : $\sum_{h=i+1}^{k+1} n_h \beta_i$. En ajoutant ces valeurs à la

somme des dates de fin (7.1), nous obtenons :

$$\sum C_j = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} \left(n - \sum_{h=0}^{i-1} n_h - r + 1 + \sum_{h=i+1}^{k+1} n_h w([i,r],r) \right) p([i,r],r) + \sum_{i=1}^{k} \sum_{h=i+1}^{k+1} n_h \beta_i \tag{7.2}$$

Figure B.1 – Répartition des tâches et des maintenances sur une machine

Pour bien illustrer cette expression, nous présentons l'exemple suivant.

Exemple :

On considère la séquence suivante : $\{T_1, T_2, m_1, T_3, T_4, m_2, T_5\}$, voir la figure B.2.

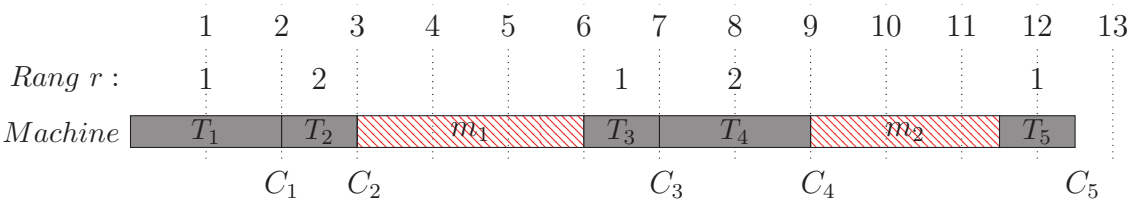

Figure B.2 – Exemple avec une machine et deux périodes de maintenance

La portion de la maintenance m_1, due à l'ordonnancement de la tâche T_2, contribue aux dates de fin des tâches des groupes suivants (trois tâches : T_3, T_4 et T_5). Ainsi, la contribution totale de cette portion à l'objectif est : $3 \times w(2,2)p(2,2)$. De même, la durée élémentaire de la première maintenance β_1 contribue à la $\sum C_j$ par une valeur $3 \times \beta_1$.

B.2 Minimisation de la variation des dates de fin

La minimisation de la variation des dates de fin est donnée par la formule :

$$TADC = \sum_{i=1}^{n} \sum_{j=i}^{n} |C_j - C_i|$$

Ce critère est défini en fonction des dates de fin de chaque tâche. Tout d'abord, en considérant le problème avec des périodes de maintenance de durées nulles ($m_i = 0$). Dans ce cas, la variation des dates de fin peut être exprimée comme suit (voir Kanet [65]) :

$$TADC = \sum_{i<j} |C_i - C_j| = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} \left(\sum_{h=0}^{i-1} n_h + r - 1 \right) \left(n - \sum_{h=0}^{i-1} n_h - r + 1 \right) p([i,r],r) \quad (7.3)$$

$$= \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} (s_{ir} - 1)\,(n - s_{ir} + 1)\,p([i,r],r) \quad (7.4)$$

où $s_{ir} = r + \sum_{h=0}^{i-1} n_h$, avec $n_0 = 0$, s_{ir} représente le rang global de la tâche sur la machine.

Lorsque les périodes de maintenance suivent le modèle $\mathcal{MP}$ (durées non nulles), les dates de fin des tâches ordonnancées dans un groupe G_i sont augmentées par les durées des périodes qui le précèdent (les périodes de maintenance $m_1, m_2, \ldots, m_{i-1}$). La somme de la variation des dates de fin augmente par la durée de la première maintenance si une tâche du groupe G_1 est combinée avec une tâche de groupes G_2, G_3,.... D'une façon générale, une période de maintenance i augmente la valeur de $TADC$, lorsqu'une tâche appartient à un groupe G_x ($x \leq i$) et combinée avec une tâche de groupe G_y ($y > i$).

La portion de maintenance, associée à une tâche T_j ordonnancée au rang r du groupe G_i, participe à la valeur finale du $TADC$. La durée de cette portion $w(j,r)p(j,r)$ intervient dans chaque différence $|C_x - C_y|$, avec $x \leq i$ et $y > i$. Ainsi, la contribution totale de la portion de maintenance $w(j,r)p(j,r)$, liée à la tâche T_j insérée dans le groupe G_i, est : $(\sum_{h=1}^{i} n_h)(\sum_{h=i+1}^{k+1} n_h)w(j,r)p(j,r)$. De même, la durée élémentaire de la $i^{\text{ème}}$ période de maintenance contribue $(\sum_{h=1}^{i} n_h)(\sum_{h=i+1}^{k+1} n_h)$ fois dans la valeur du $TADC$.

Alors, la durée du $TADC$, en présence de k périodes de maintenance dont les durées suivent le modèle $\mathcal{MP}$, peut être exprimée par :

$$TADC = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} \left((s_{ir} - 1)(n - s_{ir} + 1) + (\sum_{h=1}^{i} n_h)(\sum_{h=i+1}^{k+1} n_h)w([i,r],r) \right) p([i,r],r)$$
$$+ (\sum_{h=1}^{i} n_h)(\sum_{h=i+1}^{k+1} n_h)w([i,r],r)\beta_i$$

Pour bien illustrer cette expression, nous considérons l'exemple suivant :

Exemple :

On considère la séquence suivante : $\{T_1, T_2, m_1, T_3, T_4, m_2, T_5\}$, voir la figure B.3.

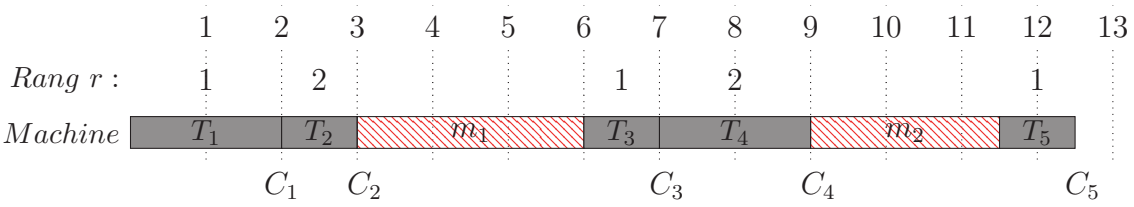

Figure B.3 – Exemple avec une machine et deux périodes de maintenance

La portion de la maintenance m_1 due à l'ordonnancement de la tâche T_2 contribue par une valeur $w(2,2)p(2,2)$ aux différences entre les dates de fin des tâches du groupe G_1 et celles des groupes G_2 et G_3, *i.e.* six différences : $|C_3 - C_1|$, $|C_3 - C_2|$, $|C_4 - C_1|$, $|C_4 - C_2|$, $|C_5 - C_1|$ et $|C_5 - C_2|$.

Ainsi, la contribution totale de cette portion est : $6 \times w(2,2)p(2,2)$. De même, la durée élémentaire de la première maintenance β_1 contribue au $TADC$ par une valeur $6 \times \beta_1$.

B.3 Minimisation de la variation des durées d'attente

La minimisation de la variation des durées d'attente est donnée par la formule :

$$TADW = \sum_{i=1}^{n} \sum_{j=i}^{n} |W_j - W_i|$$

Ce critère est défini en fonction des durées d'attente de chaque tâche. Tout d'abord, nous considérons le problème avec des périodes de maintenance de durées nulles ($m_i = 0$). Dans ce cas, la variation des durées d'attente peut être exprimée comme suit (voir Bagachi [9]) :

$$\begin{aligned} TADW = \sum_{i=1}^{n} \sum_{j=i}^{n} |W_j - W_i| &= \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} \left(\sum_{h=0}^{i-1} n_h + r \right) \left(n - \sum_{h=0}^{i-1} n_h - r \right) p([i,r],r) \\ &= \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} s_{ir} \left(n - s_{ir} \right) p([i,r],r) \end{aligned} \tag{7.5}$$

où $s_{ir} = r + \sum_{h=0}^{i-1} n_h$, avec $n_0 = 0$, s_{ir} représente le rang global de la tâche sur la machine.

En utilisant la même analyse que précédemment, nous pouvons montrer que la contribution de la maintenance à la valeur finale du $TADW$ est identique à celle du $TADC$. Ainsi, la valeur finale de la variation des durées d'attentes peut être exprimée comme suit :

$$\begin{aligned} TADW = \sum_{i=1}^{k+1} \sum_{r=1}^{n_i} \Bigg(& s_{ir}(n - s_{ir}) + (\sum_{h=1}^{i} n_h)(\sum_{h=i+1}^{k+1} n_h) w([i,r],r) \Bigg) p([i,r],r) \\ & + (\sum_{h=1}^{i} n_h)(\sum_{h=i+1}^{k+1} n_h) w([i,r],r) \beta_i \end{aligned}$$

Annexe C

C Liste des nos publications

C.1 Articles de revue

[37] A. Gara-Ali, G. Finke, et M.-L. Espinouse. *Parallel-machine scheduling with maintenance : Praising the assignment problem.* European Journal of Operational Research **252**, 90 – 97 (2016).

[31] A. Gara-Ali et M.-L. Espinouse. *Erratum to :"Simultaneously scheduling n jobs and the preventive maintenance on the two-machine flow shop to minimize the makespan".* International Journal of Production Economics **153**, 361 – 363 (2014).

[30] G. Finke, A. Gara-Ali, M.-L. Espinouse, V. Jost, et J. Moncel. *Unified matrix approach to solve production-maintenance problems on a single machine.* Omega (2016).

C.2 Conférences internationales

[33] A. Gara-Ali et M.-L. Espinouse. *A two-machine flow-shop scheduling with a deteriorating maintenance activity on the second machine.* In *Industrial Engineering and Systems Management (IESM), 2015 International Conference*, 481– 488, Séville, Espagne (2015).

[34] A. Gara-Ali, M.-L. Espinouse, et G. Finke. *Unrelated parallel-machine scheduling with deterioration effects and multi-maintenance activities.* In *CIE-44 - 44th International conference on Computers & Industrial Engineering*, 1080-1089, Istanbul, Turkey (2014).

[35] A. Gara-Ali, M.-L. Espinouse, et G. Finke. *Unrelated parallel-machine scheduling with optional maintenance activity on each machine and general effects of deterioration.* In *MOSIM 2014 - 10th International Conference on Modeling, Optimization & SIMulation*, Nancy, France (2014).

C.3 Conférences nationales

[36] A. Gara-Ali, M.-L. Espinouse, et G. Finke. *Problèmes d'ordonnancement sur des machines parallèles avec des périodes de maintenance et un effet de détérioration : Minimisation de la charge totale des machines.* In *ROADEF - 16ème congrès annuel de la Société Française de Recherche Opérationnelle et Aide à la Décision*, Marseille, France (2015).

[32] A. Gara-Ali et M.-L. Espinouse. *Méthodes exactes pour la résolution du problème Flow-shop à deux machines avec des contraintes de disponibilité sur la deuxième machine.* In *ROADEF - 15ème congrès annuel de la Société française de recherche opérationnelle et d'aide à la décision*, Bordeaux, France (2014).

C.4 Working papers

[29] G. Finke, A. Gara-Ali, M.-L. Espinouse, V. Jost, et J. Moncel. *Single-machine production and maintenance : a unified approach.* Cahiers Leibniz (205) (2013).

Ahmed Gara-Ali : *Ordonnancement de tâches et de périodes d'indisponibilité de durée variable.* Thèse de doctorat, 19 juillet 2016

Résumé

Dans cette thèse, nous nous intéressons aux problèmes d'ordonnancement simultané de tâches et de périodes d'indisponibilité. Dans un premier temps, nous réalisons une revue de littérature sur la prise en compte des indisponibilités dans les problèmes d'ordonnancement.

Ensuite, nous définissons un modèle général qui englobe des modèles existants de la littérature pour des ateliers à une machine et à machines parallèles. Une approche globale de résolution basée sur les problèmes d'affectation linéaire a été développée. Cette approche permet de résoudre le modèle général comme un simple problème d'affectation. Un grand nombre de critères d'optimisation et de modèles de maintenance peuvent être traités en utilisant cette approche, fournissant ainsi l'accès à tous les modèles qui ont souvent été étudiés séparément dans la littérature. Les résultats élaborés avec cette approche ont permis de résoudre des problèmes d'ordonnancement non traités avant et aussi de généraliser et améliorer des résultats antérieurs.

Nous proposons, en dernier lieu, une étude d'un problème flow shop à deux machines en présence d'une période d'indisponibilité sur la deuxième machine. Une étude de complexité est menée sur le problème. Ensuite, nous définissons des propriétés d'optimalité. En se basant sur ces propriétés, trois méthodes de résolution exacte sont proposées ; une méthode énumérative, un programme linéaire et une méthode basée sur l'approche de séparation et évaluation $B\&B$. Une analyse expérimentale est présentée afin d'évaluer les performances de ces méthodes.

Mots clés : période d'indisponibilité, maintenance, ordonnancement simultané, durée variable

Abstract

In classical scheduling problems, machines are assumed to be continuously available. However, in a real manufacturing system, machine may become unavailable during the scheduling period due to preventive maintenance. In this dissertation, we are interested in the problems of jointly scheduling jobs and unavailability periods.

We start our study by introducing a general framework for scheduling problems and we present a review of the scheduling problems with unavailability periods.

Then, we consider a general model for scheduling jobs on single-machine and unrelated parallel-machines with maintenance interventions. A unified approach is presented to solve this model as an assignment problem. A large number of performance criteria and maintenance models can be treated in this way, thus providing access to models that have often been studied separately in the published literature.

Finally, we focus on the problem of a two-machine flow-shop makespan scheduling with the deteriorating maintenance period on the second machine. Then, we establish some conditions of the optimal schedule. In order to solve the problem, we proposed different exact methods : enumerative method, mixed-integer programming (MIP) model and a branch & bound algorithm. Numerical experiments are reported for all the proposed methods.

Keywords : unavailability constraints, maintenance activities, scheduling, variable duration