N° d'ordre : 25/2009-M/MT

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE « HOUARI BOUMEDIÈNE »

FACULTÉ DE MATHÉMATIQUES

MÉMOIRE

Présenté pour l'obtention du diplôme de MAGISTER

EN : MATHÉMATIQUES

Spécialité : Recherche Opérationnelle : Mathématiques de Gestion

Par

BOUCHEBAH Kahina

Sujet

L'ORDONNANCEMENT MULTICRITÈRE

Soutenu publiquement, le 23/06/2009 devant le jury composé de :

Mr- M. AÏDER	Professeur,	**USTHB**	Président
Mr- M. MOULAÏ	Professeur,	**USTHB**	Directeur de thèse
Mr- M. BOUDHAR	Maître de Conférences,	**USTHB**	Examinateur
Mr- A/H. MEZGHICHE	Maître Assistant,	**USTHB**	Invité

Remerciements

"Je remercie ALLAH qui a exaucé mes prières et qui m'a donnée non seulement le courage mais aussi la force et la patience de réaliser ce travail".

C'est une habitude saine que de remercier au début d'un tel travail tous ceux qui, plus ou moins directement, ont contribué à sa réalisation. C'est avec mon enthousiasme le plus vif et le plus sincère que je voudrais rendre mérite à tous ceux qui à leur manière m'ont aidé à mener à bien ce mémoire.

Je désire exprimer ma profonde et respectueuse gratitude à Monsieur Mustapha MOULAÏ, Professeur à l'Université des Sciences et de la Technologie Houari Boumediène, qui a assumé la direction de ce travail et pour l'intérêt et l'enthousiasme manifestés. Sous sa direction, j'ai pu comprendre et apprendre le métier de recherche et une certaine éthique de la recherche.

Je remercie les membres du jury qui ont accepté de juger ce travail et d'y apporter leur caution :

- Monsieur M. AÏDER, Professeur à l'USTHB, Alger, qui me fait le grand honneur d'accepter la présidence du jury.
- Monsieur M. BOUDHAR, Maître de Conférence à l'USTHB, Alger, pour l'honneur qu'il me fait en acceptant de participer à ce jury. Sans ses aides et gentillesse, je n'ai pu terminer à mieux ce travail.
- Monsieur A/H. MEZGHICHE, Maître Assistant à l'USTHB, Alger, pour l'honneur qu'il me fait en acceptant également de participer à ce jury.

On n'a pas souvent l'occasion de remercier ses parents dans une vie, alors je suis contente que l'occasion m'en soit donnée ici. Merci *maman*, Merci *papa*. Je vous remercie tout au fond de mon cœur d'avoir cru en moi. Je vous remercie pour votre présence dans ma vie. Vous avez été et vous serez toujours la lueur de bonheur et d'espoir dans ma vie. Vous êtes toujours là pour moi. Merci pour votre confiance, gentillesse et bonté. Merci de m'avoir fourni tout ce dont j'avais besoin pour que je sois dans les meilleures conditions, allant même à vous sacrifier. Merci aussi à mon frère FATAH, tu a été toujours à mes côté lorsque j'en avais vraiment besoin. Je vous remercie mes sœurs surtout toi SOUHILA.

Particulièrement, je remercie C. BOUGHANI qui m'a aidée, soutenue et encouragée tout au long de ce travail. Également, mes reconnaissances les plus distinguées sont adressées à K. KABYL qui m'a vraiment encouragée. Malgré ses occupations, il a pu me donner un fort coup de main. Enfin, que tout ceux qui ont contribué de près ou de loin à l'aboutissement de ce travail trouvent ici l'expression de mes remerciements les plus distingués.

Dédicaces

A mes très chers parents, pour leur persévérance
et pour avoir suscité ma vocation
et permis d'achever mes études en tant que suis actuellement,

A mes sœurs, en particulier Souhila,
et mon frère unique Fatah, pour leur soutien moral
et leur intérêt envers mon travail,

A tous mes amis et collègues,
A tous ceux qui m'ont aidé,
A tous ceux qui me sont chers,
A tous ceux qui j'ai omis,

je dédie ce humble travail.

KAHINA BOUCHEBAH.

Dans les sciences, le chemin est plus important que le but. Les sciences n'ont pas de fin.

/* Erwin CHARGAFF (1905-2002) */

TABLE DES MATIÈRES

Table des Matières **i**

Table des Figures **iv**

Introduction **v**

1 L'Optimisation Multi-objectif **1**

1.1 Concepts de base d'optimisation multi-objectif 1

1.1.1 Formulation d'un problème multi-objectif 1

1.1.2 Les concepts d'optimalité 2

1.1.3 Quelques définitions de base 4

1.1.4 Front Pareto et surface de compromis 5

1.1.5 Optima supportés et non supportés de Pareto 6

1.2 Classes de méthodes de résolution 7

1.3 Détermination des optima de Pareto 9

1.3.1 Détermination par l'agrégation des critères 9

1.3.2 Détermination par l'analyse paramétrique 11

1.3.3 Détermination par l'approche ϵ-contrainte 13

1.3.4 Utilisation de la métrique de Tchebycheff 14

1.3.5 Détermination par l'approche du but à atteindre (Goal attainment) . . 16
1.3.6 Programmation par buts (Goal programming) 16
1.3.7 Autres méthodes pour la détermination des optima de Pareto 18

2 L'Ordonnancement Multicritère 20

2.1 Notions préliminaires en ordonnancement 20
2.1.1 Définition 20
2.1.2 Classification des problèmes d'ordonnancement 21
2.1.3 Diagramme de Gantt 27
2.1.4 Quelques notions fondamentales 27
2.2 Complexité 30
2.3 Méthodes classiques de résolution 32
2.3.1 Méthodes dédiées 32
2.3.2 Méthodes génériques 34
2.4 L'Ordonnancement Multicritère 42
2.4.1 Exemples d'application 42
2.4.2 Définition 44
2.4.3 Notation des problèmes d'ordonnancement multicritères 44
2.4.4 Complexité 46

3 Stratégies d'Approches de Résolution 47

3.1 Méthodes exactes 47
3.1.1 Programmation dynamique 47
3.1.2 Méthode de séparation et évaluation (Branch-and-Bound) 48
3.2 Approches de résolution approchées 48
3.2.1 Transformation du problème vers le mono-critère 48
3.2.2 Approches par traitement séparé des objectifs 48
3.3 Métaheuristiques 49
3.3.1 Algorithmes évolutionnaires multi-objectifs 49
3.3.2 Métaheuristiques alternatives multi-objectifs 52
3.3.3 Quelques observations sur les métaheuristiques multi-objectifs 54

4 Un État de l'Art sur Quelques Types de Problèmes d'Ordonnancement Multicritère 56

4.1 Problèmes d'ordonnancement à une seule machine 56

4.1.1 Minimiser la somme des dates de fin d'exécution des tâches 57

4.1.2 Minimiser la somme des dates de fin d'exécution pondérées des tâches . 62

4.1.3 Minimiser les coûts des durées de dépréciation pondérées de toutes les tâches 64

4.1.4 Minimiser les coûts de changement d'équipements 64

4.1.5 Minimiser des critères avec des dates échues imposées 66

4.1.6 Minimisation des critères Juste-à-Temps 67

4.2 Problèmes d'ordonnancement sur machines parallèles 72

4.2.1 Problèmes avec machines parallèles identiques 73

4.2.2 Problèmes avec machines parallèles uniformes 77

4.2.3 Problèmes avec machines parallèles générales 78

4.3 Problèmes d'ateliers 79

4.3.1 Problèmes Flowshop à deux machines 79

4.3.2 Problèmes Flowshop à m machines 85

4.3.3 Problèmes Jobshop et Openshop 88

4.3.4 Problèmes d'ateliers avec affectation de ressources 89

5 Une Solution Faiblement Efficace pour le Problème $P||Lex(C_{\max}, L_{\max})$ 91

5.1 Le problème $P||Lex(C_{\max}, L_{\max})$ 91

5.2 Une Solution Faiblement Efficace pour le Problème $P||Lex(C_{\max}, L_{\max})$ 92

5.3 Les différentes procédures de l'algorithme par séparation et évaluation 94

5.4 Résultats théoriques 96

5.5 Exemple illustratif 97

5.6 Conclusion 101

Conclusion 104

Bibliographie 106

TABLE DES FIGURES

1.1 Le front Pareto. 6

1.2 Optima supportés et non supportés de Pareto. 7

1.3 Interprétation géométrique du problème (P_α). 11

1.4 Interprétation géométrique du problème $(P_{(g,b)})$. 12

1.5 Interprétation géométrique du problème (P_{ϵ^k}). 14

1.6 Interprétation géométrique du problème (P_θ). 15

1.7 Interprétation géométrique du problème $(P_{(z^{ref},w)})$. 17

2.1 Une typologie des problèmes d'ordonnancement. 22

2.2 Exemple de diagramme de Gantt. 27

2.3 Inclusion des classes d'ordonnancements. 29

4.1 L'ordonnancement déterminé pour l'obtention de $C^+_{\max}$. 75

4.2 L'ensemble E dans l'espace des critères. 76

4.3 L'ordonnancement déterminé par l'heuristique de (NAGAR & al, 1995). 83

5.1 Un ordonnancement Pareto optimal pour l'exemple illustratif. 101

5.2 L'arborescence construite par l'algorithme 3 appliqué sur l'exemple illustratif (à suivre ...). 102

5.3 (... suite) L'arborescence construite par l'algorithme 3 appliqué sur l'exemple illustratif. 103

Introduction

Depuis de nombreuses années, la théorie de l'ordonnancement est un domaine largement étudié dans la littérature et définit elle-même une partie de la Recherche Opérationnelle [1]. Elle joue un rôle privilégié s'inscrivant dans des niveaux de décision à la fois tactique et opérationnel [62]. Il s'agit de contrôler, à court ou moyen terme, l'activité d'un ensemble de ressources disponibles en quantité limitée, en gérant les conflits d'utilisation que pose la réalisation d'un ensemble d'activités sur un horizon de temps généralement imposé. En effet, un problème d'ordonnancement consiste à trouver une séquence d'un certain nombre de travaux à être exécutés sur de différentes machines afin que des contraintes technologiques soient satisfaites et un ou plusieurs critères de performances soient optimisés [188].

Plusieurs approches et modèles ont été proposés pour résoudre le problème d'ordonnancement, à savoir : La programmation mathématique à variables discrètes, les techniques de simulation, etc. Des algorithmes spécifiques ont résolu plusieurs problèmes simples ; de nombreuses heuristiques ont été aussi utilisées. Le choix d'une méthode appropriée dépend de la complexité du problème, le nombre et la configuration des machines, les systèmes de production, le système d'ordonnancement et la nature statique ou dynamique des arrivées des travaux [135].

[1] Au début des années 1900, Henry GANTT introduit l'ainsi appelé *le diagramme de Gantt*. Le premier travail sur l'ordonnancement a été publié par S. M. JOHNSON en 1954 (et écrivit en 1953) sur le problème du flowshop à deux machine avec minimisation du makespan.

Cependant, dans de nombreuses situations pratiques, le décideur doit prendre en compte plusieurs critères simultanément. Ces critères sont généralement conflictuels. Ainsi, l'optimisation d'un critère est faite aux dépends des autres. Est-il de considérer plusieurs critères à optimiser sans intérêt ? Ou facilite-il le problème ? La réponse à ces deux questions est clairement *non* pour des considérations pratiques aussi bien que théoriques. Dans des industries et des différents services des sociétés plus d'un seul critère sont souvent considérés. En outre, l'optimisation de critères conflictuels introduit une certaine incertitude, puisque quelques ordonnancements deviennent incomparables, ce qui fait recours à la détermination des ordonnancements incomparables les plus intéressants, à savoir : les optima de Pareto [2]. [55]

La difficulté de satisfaire simultanément un ensemble de critères a été l'objectif de plusieurs publications, dans ce domaine. Nons citons, à titre d'exemple, les travaux de [202], [192], [177] et [47].

Dans ce mémoire, nous nous sommes intéressés à la présentation d'un état de l'art sur quelques types de problèmes d'ordonnancement multicritère les plus importants. Après avoir évoqué le mode de traitement de ce type de problèmes - modélisation, prise en compte des critères, résolution à l'aide de procédures spécifiques - nous présentons un état de l'art sur quelques problèmes d'ordonnancement multicritère à une seule machine, sur machines parallèle et problèmes d'ateliers. Ce mémoire est clôturé par l'extension d'un algorithme optimal pour le problème $P||C_{\max}$ développé par BOUDHAR [21] et basé sur la méthode de branch-and-bound. Cette extension consiste à rajouter une procédure optimisant le critère $L_{\max}$ par la règle de tri EDD afin de déterminer un ordonnancement Pareto optimal aux problème $P||Lex(C_{\max}, L_{\max})$, où un ordre lexicographique est défini entre les critères.

Notre mémoire est organisé en cinq chapitres précédés d'une introduction et suivis par une conclusion.

[2] Vilfredo Pareto (1848-1923) était un économiste italien. Il écrivit en 1896 et 1897 une édition de trois volumes de ses publications, *Cours d'Économie Politique*, et en 1906 le *Manuel de l'Économie Politique* dans lequel il introduisit la notion d'optimalité de Pareto

Dans le premier chapitre, nous présentons les résultats de base de l'optimisation multi-objectif. Différentes approches peuvent être considérées pour tenir compte des critères, dépendant de plusieurs facteurs. Ces approches sont utilisées pour la modélisation et la notation des problèmes d'ordonnancement multicritère.

Le second chapitre est consacré à la présentation, dans un premier temps, des problèmes d'ordonnancement d'une manière générale, ensuite, une approche pour la résolution des problèmes d'ordonnancement multicritère. Enfin, une extension de la notation usuelle des problèmes d'ordonnancement ainsi que des résultats de la complexité sont introduisent à la fin de ce chapitre.

Nous présentons au troisième chapitre quelques stratégies d'approches de résolution des problèmes d'ordonnancement multicritère. Nous dressons un constat des différentes techniques utilisées dans ce cadre, telles que la programmation mathématique, les techniques de recherche arborescente et les métaheuristiques.

Le quatrième chapitre est voué à la description de quelques plus importants travaux effectués dans le domaine de l'ordonnancement multicritère à une seule machine, sur machines parallèles et problèmes d'ateliers.

Le fruit porté lors de la réalisation de ce travail est relaté dans le cinquième chapitre. nous nous sommes intéressés à la minimisation du makespan $C_{\max}$ et le décalage temporel maximal $L_{\max}$ dans le problème $P||Lex(C_{\max}, L_{\max})$. Le constat rapporte sur l'extension d'un algorithme optimal basé sur la méthode de branch-and-bound développé par BOUDHAR [21] déterminant la valeur optimale du critère $C_{\max}$ dans le problème $P||C_{\max}$. À une certaine étape de la procédure de séparation, nous rajoutons la procédure optimisant un deuxième critère $L_{\max}$ par l'application de la règle de tri EDD aux sous-problèmes $1||L^j_{\max}$, $j = 1, \ldots, m$; m étant le nombre de machines. Un exemple d'application est illustré à la fin de ce chapitre.

CHAPITRE 1

L'Optimisation Multi-objectif

La principale difficulté que l'on rencontre en optimisation mono–objectif vient du fait que modéliser un problème sous la forme d'une équation unique peut être une tâche difficile. Avoir comme but de ce ramener à une seule fonction objectif peut aussi biaiser la modélisation. Dans ce chapitre, nous présentons l'ensemble des éléments fondamentaux de l'optimisation multi–objectif : Définitions d'optima de Pareto, classes d'algorithmes, méthodes de calcul d'optima de Pareto, Goal Programming, etc.

1.1 Concepts de base d'optimisation multi-objectif

1.1.1 Formulation d'un problème multi-objectif

D'un point de vue mathématique, les problèmes d'optimisation multi-objectif sont un cas spécial des problèmes d'optimisation vectorielle définis par : [188]

$$\text{(PM)} \qquad \begin{cases} \min Z(x) \qquad \text{avec} \quad Z(x) = [Z_1(x); \ldots; Z_K(x)]^T \\ \quad x \in X; \\ \quad X = \{x/[h_1(x); \ldots; h_P(x)]^T = 0,\ [g_1(x); \ldots; g_M(x)]^T \leq 0\}. \end{cases}$$

La fonction vectorielle $Z(x) = [Z_1(x); \ldots; Z_K(x)]^T$ est appelée *critère vectoriel* (*fonction multi-objectif*), dont les composantes Z_i, $i \in I = \{1, 2, \ldots, K\}$, sont des critères d'optimisation (fonctions objectifs ou fonctions de coût). Les équations $h_j(x)$, $j = 1; \ldots; P$, et les inéquations $g_l(x)$, $l = 1; \ldots; M$, sont appelées *contraintes*.

Les problèmes d'optimisation multi-objectif sont des problèmes d'optimisation vectorielle où l'espace des solutions (appelé aussi l'espace des décisions) X et l'espace des critères $\mathcal{Z} = Z(X)$ sont des espaces vectoriels euclidiens de dimensions finies, Q et K respectivement, *i.e.* $X \subset \mathbb{R}^Q$ et $Z(X) \subset \mathbb{R}^K$ avec $1 \leq Q, K < \infty$.

Pour chaque décision $x \in X$, un et un seul vecteur lui correspond dans l'espace des critères $\mathbb{R}^K$. Selon le besoin du décideur, les fonctions objectifs peuvent être à maximiser ou à minimiser. On peut toujours passer d'un cas à maximiser à un cas à minimiser et vice versa.

Le décideur se trouve dans une situation, où il doit choisir une décision qui engendre la plus petite valeur possible pour chacune des fonctions objectifs. Mais en pratique, il n'existe que rarement la solution idéale $x^* \in X$.

1.1.2 Les concepts d'optimalité

Relation d'ordre et de dominance au sens de Pareto

Le choix d'une décision est basé sur les évaluations des fonctions objectifs. Dans un problème multi-objectif, l'ensemble des évaluations des fonctions objectifs est un sous ensemble de $\mathbb{R}^K$: $F = Z(x) \subset \mathbb{R}^K$. Par conséquent, le problème revient à déterminer un vecteur $z^* \in F = Z(x)$ qui entraînerait le choix de la décision correspondante $x^* \in X$ tel que $z^* = Z(x^*)$. La difficulté pour un problème multi-objectif réside dans le choix de z^* en raison de l'absence d'une relation d'ordre dans $\mathbb{R}^K$. Ainsi, dans l'optimisation multi-objectif, il s'agit beaucoup plus de définir une solution de meilleurs compromis que de rechercher une solution idéale qui n'est pas toujours possible.

Comme la solution optimale est une multitude de points de $\mathbb{R}^K$, il est vital pour identifier ces meilleurs compromis de définir une « *relation d'ordre* » entre ces éléments. Dans le cas des problèmes d'optimisation multi-objectif, ces relations d'ordre sont appelées *relations de dominance*. Plusieurs relations de dominance ont été déjà présentées, mais la plus célèbre et la plus utilisée est la *dominance de Pareto*. [11]

Dominance

Définition 1.1.1. Soient y et z deux vecteurs critères, tels que y, $z \in Z(X) \subset \mathbb{R}^K$; alors le vecteur y domine le vecteur z si et seulement si $y \leq z$ et $y \neq z$ (*i.e.* il existe au moins un $i \in I$, tel que $y_i < z_i$ et pour tout $j \neq i$, $y_j \leq z_j$).

Si y domine z aucune composantes de y n'est supérieure à la composante correspondante de z et au moins une composante de y est plus petite que la composante correspondante de z.

Dominance forte

Définition 1.1.2. Soient y et z deux vecteurs critères tels que y, $z \in Z(X) \subset \mathbb{R}^K$, alors le vecteur y domine *fortement* le vecteur z si et seulement si $y < z$ (*i.e.* $\forall i \in I$, $y_i < z_i$).

Si y domine fortement z, alors y est meilleurs que z sur tous les critères.

Efficacité

Définition 1.1.3.
Une solution $x \in X$ est une *solution efficace*, si $\nexists y \in X$ tel que $Z(y)$ domine $Z(x)$.

Un point est efficace si son image dans l'espace des critères est un vecteur critère non dominé. Cela signifie que tout gain sur un critère entraîne nécessairement une perte sur au moins un autre.

Les définitions suivantes sont extraites des deux références [188] et [11] :

Efficacité faible

Définition 1.1.4.
$x \in X$ est un *optimum de Pareto faible*, aussi appelé *solution de Pareto faible* ou *solution faiblement efficace*, si $\nexists y \in X$ tel que $\forall i \in I$, $Z_i(y) < Z_i(x)$. Nous notons FE l'ensemble des optima de Pareto faibles de l'ensemble X.

Efficacité stricte

Définition 1.1.5.
$x \in X$ est un *optimum de Pareto strict*, aussi appelé *solution de Pareto stricte* ou *solution strictement efficace*, si $\nexists y \in X$ tel que $\forall i \in I$, $Z_i(y) \leq Z_i(x)$ dont au moins une inégalité qui est stricte. Nous notons E l'ensemble des optima de Pareto stricts de l'ensemble X.

Efficacité Propre

Définition 1.1.6.
Soient $x, y \in X$, $y \neq x$ et $I_y = \{i \in [1; K]/Z_i(y) < z_i(x)\}$. $x \in X$ est un *optimum de Pareto propre*, aussi appelé *solution proprement efficace*, si x est un optimum strict de Pareto et $\exists M > 0$ tel que
$\forall y \in X$, $y \neq x$, $I_y \neq \emptyset \Rightarrow \forall i \in I_y$, $(\exists j, 1 \leq j \leq K$ avec $Z_j(x) < Z_j(y))$ tel que $\frac{Z_i(x)-Z_i(y)}{Z_j(y)-Z_j(x)} \leq M$.
Nous notons PRE l'ensemble des optima de Pareto propres de l'ensemble X.

Proposition 1.1.1. *On a les inclusions*

$$E \subseteq FE$$

$$PRE \subseteq E$$

Remarque 1.1.1. *Ces définitions sont valides si seulement si chaque critère Z_i peut atteindre sa valeur minimale.*

1.1.3 Quelques définitions de base

Le point idéal

Définition 1.1.7.
$Z^{id} = [Z_1^{id}; \ldots; Z_K^{id}]$ est le ***point idéal***, ou ***vecteur critère idéal*** si $Z_i^{id} = \min\limits_{x \in X} (Z_i(x))$, $\forall i = 1, \ldots, K$. Généralement, ce vecteur ne correspond à aucune solution réalisable.

La matrice des gains

Définition 1.1.8.
Soient K vecteurs $Z^i = [Z_1^i; \ldots; Z_K^i]$ vérifiant $Z_i^i = Z_i^{id}$, $\forall i = 1, \ldots, K$. La ***matrice des gains***, notée G est définie par $G_{j,i} = Z_j^i$, $\forall i = 1, \ldots, K$, $\forall j = 1, \ldots, K$. Cette matrice n'est pas nécessairement unique.

Le point nadir

Définition 1.1.9.
Soit G la matrice des gains. Le ***point nadir*** est le vecteur critère, noté Z^{nd}, défini par $Z_j^{nd} = \max_{i=1,\ldots,K} (G_{j,i})$, $\forall j = 1, \ldots, K$. On note que ce point dépend de la matrice des gains considérée (s'il en existe plusieurs).

Le point utopique

Définition 1.1.10.
Z^{ut} est le ***point utopique***, ou ***vecteur critère utopique*** si Z^{ut} domine Z^{id}, *i.e.* $Z^{ut} \leq Z^{id}$ avec au moins une inégalité qui est stricte. Ce point ne correspond à aucune solution réalisable.

Le point de référence

Définition 1.1.11.
Généralement, on appelle un ***point de référence***, ou ***vecteur critère de référence*** tout vecteur Z^{ref} considéré comme un objectif à atteindre. L'objectif est de trouver la solution la plus proche à ce point, dans le sens d'une fonction à optimiser. Les points Z^{id}, Z^{nd} et Z^{ut} sont des points de référence.

1.1.4 Front Pareto et surface de compromis

La solution que nous cherchons à un problème d'optimisation multi-objectif n'est pas un point unique mais un ensemble de points que nous appellons l'***ensemble des meilleurs compromis***. Ce dernier, appelé aussi *surface de compromis* ou *front Pareto*, est composé des points qui ne sont dominés par aucun autre.

Ensemble des solutions non dominées

Définition 1.1.12.
Soit $\mathcal{Z}$ l'image de l'ensemble réalisable X dans l'espace des objectifs. L'ensemble des solutions non dominées de X, est défini par l'ensemble :

$$ND(X) = \{x \in X / \ x \text{ est } \ \text{non } \ \text{dominé } \ \text{par } \ \text{rapport à } X\}$$

Front Pareto

Définition 1.1.13.
Soit $\mathcal{Z}$ l'image de l'ensemble réalisable X dans l'espace des objectifs. Le front Pareto $ND(\mathcal{Z})$ de $\mathcal{Z}$, est défini comme suit :

$$ND(\mathcal{Z}) = \{y \in \mathcal{Z} / \nexists z \in \mathcal{Z},\ z < y\}$$

Le front Pareto est aussi appelé l'***ensemble des solutions efficaces***.

Un exemple de surface de compromis (front Pareto) en dimension 2 est illustré par la FIG. 1.1. Dans cet exemple, le problème considéré est un problème de minimisation avec deux critères. Deux points particuliers apparaissent clairement : Le point idéal et le point nadir. Ces deux points sont calculés à partir du front Pareto. Le point idéal (resp. le point nadir) domine (resp. est dominé par) tous les autres points de la surface de compromis. Bien que ces points ne soient pas forcément compris dans la zone réalisable, ils servent souvent de pôle d'attraction (resp. de répulsion) lors de la résolution du problème.

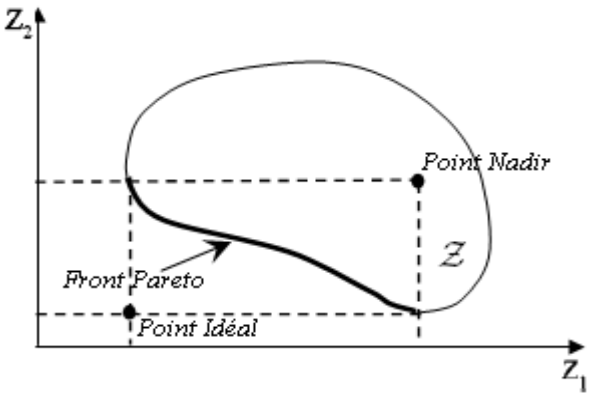


FIG. 1.1 – Le front Pareto.

1.1.5 Optima supportés et non supportés de Pareto

Dans le cas où les hypothèses de convexité sur $\mathcal{Z}$ sont absentes, deux types distincts d'optima apparaissent : Les optima supportés et les optima non supportés de Pareto. La FIG. 1.2 présente cette distinction à l'aide d'un exemple à deux dimensions. $\mathcal{Z}$ est l'ensemble des points représentés et $co(\mathcal{Z})$ est l'enveloppe convexe définie par $\mathcal{Z}$. Nous avons $co(\mathcal{Z}) = \{z \in \mathbb{R}^K\ / \forall \alpha_i \in [0;1],\ \sum_{i=1}^{K} \alpha_i = 1,\ \text{et}\ \forall z^i \in \mathcal{Z}, z = \sum_{i=1}^{K} \alpha_i z^i\}$. Le point z^0 correspond à un

ou plusieurs optima non supportés stricts de Pareto car z^0 n'appartient pas à la frontière de $co(\mathcal{Z})$. Le point z^4 représente les optima non supportés faibles de Pareto.

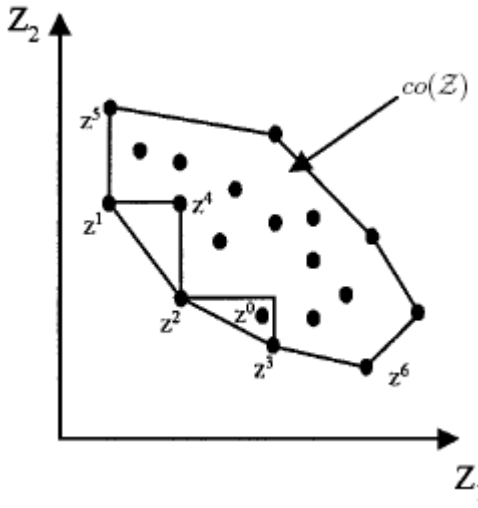


FIG. 1.2 – Optima supportés et non supportés de Pareto.

z^1, z^2, z^3, z^6 : Optima supportés de Pareto stricts.

z^0 : Optima non supportés de Pareto stricts.

z^5 : Optima supportés de Pareto faibles.

z^4 : Optima non supportés de Pareto faibles.

FE_s (resp. E_s) indique l'ensemble des optima supportés de Pareto faibles (resp. stricts). FE_{ns} (resp. E_{ns}) indique l'ensemble des optima non supportés de Pareto faibles (resp. stricts).

Alors pourquoi ne pas satisfaire des optima supportés ? Tout d'abord parce que ces solutions peuvent ne représenter qu'un petit sous-ensemble des solution efficaces. De plus, ces optima supportés ne sont pas forcément bien répartis le long du front et ne représentent pas toujours un bon compromis.

1.2 Classes de méthodes de résolution

Des optima de Pareto correspondent à des solutions de compromis entre différents critères conflictuels. Seulement, le décideur peut choisir la solution la plus satisfaisante pour son

problème, dans l'ensemble E ou FE. Traditionnellement, les problèmes d'optimisation multi-objectif constitue une partie de l'approche d'aide multicritère à la décision, qui est à son tour présentée dans une logique d'agrégation des critères permettant l'élaboration d'un modèle mathématique accordant une place importante à la phase de modélisation des préférences, privilégiant la prise en compte d'aspects qualitatifs et de critères et objectifs multiples difficilement commensurables et utilisée en suite dans une procédure d'aide à la décision. Donc, lorsqu'un décideur choisit une solution, il optimise une *fonction utilité*, *i.e.* une fonction qui agrège tous les critères appelée *fonction d'agrégation*. Elle n'est pas connue avec certitude, mais acceptant que la solution optimale est un optimum de Pareto. Parmi les solutions qui peuvent être cherchées, nous distinguons : [188]

1. Des solutions qui sont des optima de Pareto propres, stricts ou faibles.

2. Parmi les optima de Pareto, ceux qui satisfont au mieux les conditions du décideur.

L'analyste doit proposer un algorithme de résolution pour le problème d'optimisation multi-objectif, *i.e.* un algorithme qui permettra à un décideur de choisir une solution. Pour ce faire, il doit prendre en compte toutes les informations disponibles : Le décideur peut fournir les poids des critères à l'algorithme de résolution ou préciser des buts à atteindre, etc. En outre, l'analyste est conscient que la résolution du problème d'optimisation multi-objectif ne se fait pas sans l'intervention du décideur. Un algorithme de résolution qui ne fait pas intervenir le décideur, peut seulement déterminer l'ensemble de tous les optima de Pareto. [188]

EVANS [64] présente trois cas où le décideur peut intervenir : Avant, durant et après le processus de résolution. Une catégorie de méthodes peut être associée à chacun de ces cas :

1. Les méthodes permettant au décideur d'intervenir avant le processus de résolution, appelées *méthodes à priori*.

2. Les méthodes permettant au décideur d'intervenir durant le processus de résolution, appelées *méthodes interactives*.

3. Les méthodes permettant au décideur d'intervenir après le processus de résolution, appelées *méthodes à posteriori*.

Dans les *méthodes à priori*, le processus de résolution ne peut pas être déroulé sans que le décideur avait fourni un ensemble d'informations ; comme d'exemple la valeur des poids des critères pour la minimisation d'une combinaison linéaire des critères. La détermination des

valeurs de ces paramètres constitue elle-même un problème, qui demande l'utilisation d'une méthode d'aide à la décision. [199], [162] et [161].

Dans les *méthodes interactives*, le processus de résolution est itérative. Chaque itération fournie une solution au décideur, qui n'est pas nécessairement un optimum de Pareto. Alors, il dirige le processus en fournissant, directement ou indirectement, des nouvelles valeurs aux paramètres du problème. À titre d'exemple, il peut contribuer des nouveaux poids aux critères, ou l'amélioration/détérioration de certaines valeurs des critères par rapport à la solution courante. Le processus est alors capable de calculer une nouvelle solution et il peut être réitéré. Cette catégorie de méthodes a été l'objet de plusieurs études dans le domaine de l'optimisation multi-objectif et plus général dans le domaine de l'aide à la décision. [195]

Les *méthodes à posteriori* visent à fournir au décideur un ensemble exhaustif d'optima de Pareto, parmi lesquels appartient la solution la plus satisfaisante. L'ensemble des optima de Pareto proposé au décideur dépend des propriétés du problème résolu. [188]

1.3 Détermination des optima de Pareto

Un grand nombre d'approches existent pour la résolution des problèmes d'optimisation multi-objectif. Certaines utilisent des connaissances du problème pour fixer des préférences sur les critères et ainsi contourner l'aspect multicritère du problème. D'autres mettent tous les critères au même niveau d'importance, mais dans ce cas aussi il existe plusieurs manières de réaliser une telle opération. Plusieurs ouvrages et/ou articles de synthèse ont été rédigés, des états de l'art peuvent être consultés notamment dans [56] et [46].

Les résultats présentés dans cette section se rapportent sur l'agrégation des critères en un seul ou plusieurs critères plus généraux, en rajoutant des nouveaux paramètres (poids, buts, etc.) au problème. Généralement, les résultats les plus intéressants pour ces nouveaux critères sont les plus difficiles dans leurs applications (réglage des paramètres, complexité algorithmique, etc.). Le choix d'une méthode nécessite en plus un compromis entre la qualité de la solution trouvée et la facilité de son application. [188]

1.3.1 Détermination par l'agrégation des critères

En optimisation multi-objectif, combiner les objectifs est l'une des méthodes classiques pour l'évaluation d'une solution de compromis [78]. Elle consiste à transformer un problème

multi-objectif à un problème mono-objectif en agrégeant les différents critères sous forme d'une somme pondérée, pour lequel le résultat de base est représenté dans le théorème 1.3.1, aussi appelé le *théorème de Geoffrion*. C'est une condition nécessaire et suffisante pour la détermination des optima de Pareto propres.

Théorème 1.3.1. *[78]*
Soit X l'ensemble convexe des solutions et K critères Z_i convexes sur X. x^0 est un optimum de Pareto propre si et seulement si $\exists \alpha \in \mathbb{R}^K$ avec $\alpha_i \in]0;1[$ et $\sum_{i=1}^{K} \alpha_i = 1$, tel que x^0 est une solution optimale du problème (P_α) suivant :

$$(\mathrm{P}_\alpha) \qquad \begin{cases} \min\ g(Z(x)) \quad avec \quad g(Z(x)) = \sum_{i=1}^{K} \alpha_i Z_i(x) \\ \\ \qquad x \in X. \end{cases}$$

Les *courbes de niveau* sont un moyen pratique et adéquat pour interpréter géométriquement des différents problèmes d'optimisation. Concernant la minimisation d'une combinaison convexe des objectifs, le problème (P_α) peut être interprété de la manière suivante : Soit $S_=(a) = \{x \in X / \sum_{i=1}^{K} \alpha_i Z_i(x) = a$ avec $\alpha_i \in]0;1[$ et $\sum_{i=1}^{K} \alpha_i = 1\}$ l'ensemble des courbes de niveau dans l'espace des décisions. On note $L_=(a) = Z(S_=(a))$. Résoudre (P_α) est équivalent à déterminer la courbe de niveau d'une valeur minimale g^* tel que $L_=(g^*)$ soit tangente à $\mathcal{Z}$ dans l'espace des critères (Fig. 1.3). $L_=(g^*) \cap \mathcal{Z}$ définit dans l'espace des décision un ensemble des optima de Pareto pour le problème multi-objectif.

Dans le cas des méthodes à priori, on doit définir une manière à avoir les poids des critères à utiliser dans la fonction objectif. Dans le cas des méthodes à posteriori, on doit se référer à l'analyse paramétrique. On considère $\wedge = \{\alpha = (\alpha_1; \ldots; \alpha_K) / \forall i, \alpha_i \in]0;1[$ et $\sum_{i=1}^{K} \alpha_i = 1\}$. L'idée de base est de considérer la répartition de $\wedge$ en v parties $\wedge_i$ tel que $\wedge = \bigcup_{i=1}^{v} \wedge_i$. Chaque partie $\wedge_i$ est allouée à une division OPT_i de l'ensemble des optima de Pareto tel que $\forall \alpha^1, \alpha^2 \in \wedge_i$, si on note $OPT_i(\alpha^1)$ l'ensemble des solutions optimales de (P_α^1) et $OPT_i(\alpha^2)$ l'ensemble des solutions optimales de (P_α^2) alors $OPT_i(\alpha^1) = OPT_i(\alpha^2) = OPT_i$. Dans le cas des méthodes interactives, on peut itérativement varier, selon les instructions du décideur, les valeurs des poids α_i. Ces instructions peuvent être, selon l'algorithme à considérer, des nouveaux poids, des améliorations désirées, etc.

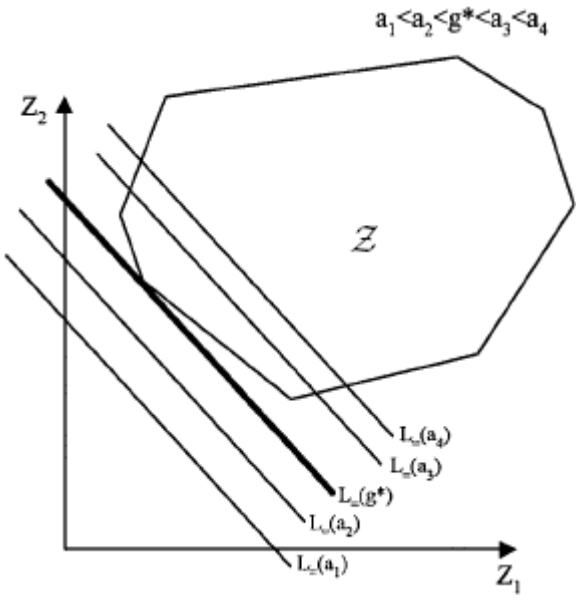


FIG. 1.3 – Interprétation géométrique du problème (P_α).

1.3.2 Détermination par l'analyse paramétrique

Un résultat intéressant et facile à utiliser, est proposé par SOLAND [178], depuis, il permet de calculer tous les optima de Pareto. Avant de présenter ce résultat, on rappelle la définition d'une fonction strictement croissante :

Définition 1.3.1. [169]
Une fonction $f : \mathbb{R}^K \longrightarrow \mathbb{R}$ est strictement croissante si $\forall x, y \in \mathbb{R}^K$, $x \neq y$, $x \leq y \Rightarrow f(x) < f(y)$.

Théorème 1.3.2. *[178]*
Soit G_Y l'ensemble des fonctions strictement croissantes de $\mathbb{R}^K$ à $\mathbb{R}$, et bornées inférieurement dans $\mathcal{Z}$, et $g \in G_Y$. $x^0 \in X$ est un optimum strict de Pareto si et seulement si $\exists b \in \mathbb{R}^K$ tel que x^0 est une solution optimale du problème $(P_{(g,b)})$ suivant :

$$(\mathrm{P}_{(\mathrm{g,b})}) \qquad \begin{cases} \min\ g(Z(x)) \\ \quad x \in X; \\ \quad Z(x) \leq b. \end{cases}$$

Le problème $(P_{(g,b)})$ est générique et simple à utiliser tant que la fonction g est choisie par l'analyste.

Géométriquement, Le problème $(P_{(g,b)})$ peut être interprété en utilisant les courbes de niveau. Soient $X' = \{x \in X / Z(x) \leq b\}$, $S_=(a) = \{x \in X' / g(Z(x)) = a\}$ et $L_=(a) = Z(S_=(a))$. La résolution de $(P_{(g,b)})$ correspond à la détermination d'une courbe de niveau d'une valeur minimale g^* tel que $L_=(g^*)$ soit tangente à $\mathcal{Z}'$ dans l'espace des critères (FIG. 1.4). $L_=(g^*) \cap \mathcal{Z}'$ définit dans l'espace des décisions un ensemble des optima de Pareto stricts du problème multi-objectif.

L'utilisation de ces résultats dans la résolution des problèmes multi-objectifs dépend es-

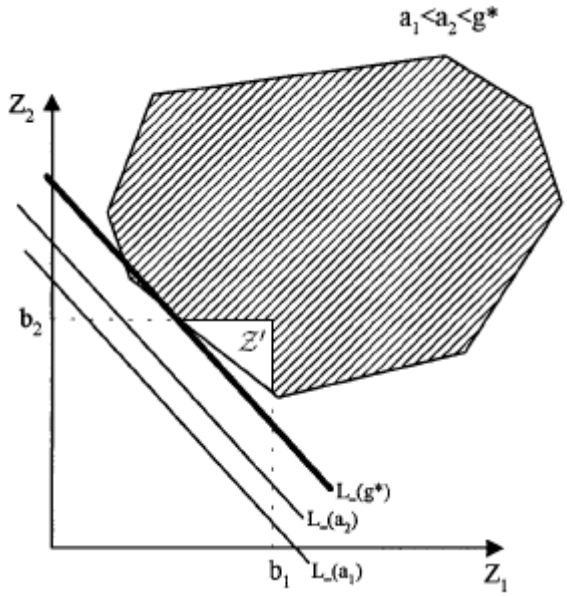


FIG. 1.4 – Interprétation géométrique du problème $(P_{(g,b)})$.

sentiellement sur la détermination du vecteur b au moment où g soit fixée. Dans une procédure itérative où dans une procédure à posteriori, on doit varier le vecteur b itérativement afin d'obtenir plusieurs optima de Pareto. À titre d'exemple, le vecteur b peut être initialisé avec des valeurs élevées et ensuite réduire ces valeurs soit par rapport aux instructions de l'analyste (algorithme interactif) soit par rapport à une procédure de réduction qui permet d'énumérer l'ensemble E. Dans une méthode à priori, il est suffisant d'interroger le décideur sur les valeurs limites b_i.

1.3.3 Détermination par l'approche ϵ-contrainte

L'approche ϵ-contrainte est souvent utilisée dans la littérature. Elle minimise un critère en connaissant que les autres $(K-1)$ critères soient bornés supérieurement. Le théorème 1.3.3 suivant permet de calculer des optima de Pareto faibles.

Théorème 1.3.3. *[141]*
Soit $x^0 \in X$. Si $\exists k \in [1;K]$, et $\exists \epsilon^k = (\epsilon_1^k; \ldots; \epsilon_{k-1}^k; \epsilon_{k+1}^k; \ldots; \epsilon_K^k) \in \mathbb{R}^{K-1}$, tel que x^0 soit une solution optimale du problème (P_{ϵ^k}) suivant :

$$(\mathrm{P}_{\epsilon^k}) \qquad \begin{cases} \min\ Z_k(x) \\ \quad x \in X; \\ \quad Z_i(x) \leq \epsilon_i^k,\ \forall i \in [1;K],\ i \neq k. \end{cases}$$

alors x^0 est un optimum de Pareto faible.

Géométriquement, le problème (P_{ϵ^k}) peut être interprété en utilisant les courbes de niveau. Soient $k \in [1;K]$ et $\epsilon^k = (\epsilon_1^k; \ldots; \epsilon_{k-1}^k; \epsilon_{k+1}^k; \ldots; \epsilon_K^k) \in \mathbb{R}^{K-1}$. Définissons $X^k = \{x \in X / Z_i(x) \leq \epsilon_i^k,\ \forall i \in [1;K],\ i \neq k\}$, $S_=(a)^k = \{x \in X^k / Z_k(x) = a\}$ et $L_=(a)^k = Z(S_=(a)^k)$. Résoudre (P_{ϵ^k}) est équivalent à déterminer la courbe de niveau avec la valeur minimale g^* tel que $L_=(g^*)^k$ soit tangente à $Z(X^k)$ dans l'espace des critères (Fig. 1.5). Si $\forall k, \exists X^k$ tel que $L_=(g^*)^k \cap Z(X^k) = \{Z(x^*)\}$ alors x^* est un optimum de Pareto strict du problème multi-objectif.

L'approche ϵ-contrainte a été utilisée largement dans la littérature (voir à titre d'exemple [181]) car elle est facile à utiliser dans un algorithme interactif : Le décideur peut interactivement spécifier et modifier les bornes et analyser l'influence de ces modifications sur la solution finale. Dans le context d'un algorithme déterminant l'ensemble de tous les optima de Pareto stricts, l'un des résultats présentés dans [188] peut être utilisé pour varier les bornes supérieures. Pour chaque borne fixée, un optimum de Pareto faible est calculé en résolvant le problème (P_{ϵ^k}). Un autre avantage de cette approche réside dans le fait qu'à chaque itération, on retrouve un problème mono-objectif pour lequel on connaît un algorithme de résolution efficace .

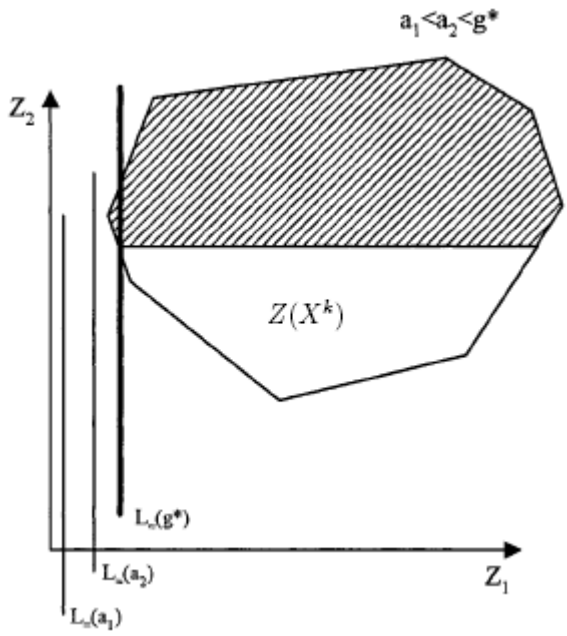


FIG. 1.5 – Interprétation géométrique du problème (P_{ϵ^k}).

1.3.4 Utilisation de la métrique de Tchebycheff

Pour déterminer les optima de Pareto, il est possible d'utiliser une métrique et chercher la solution la « *plus proche possible* » à un vecteur critère de référence. Dans cette section, une métrique particulière est présentée : « *La métrique de Tchebycheff* » ([23]).

Définition 1.3.2. [23]
Soient z^1 et $z^2 \in \mathbb{R}^K$. La *métrique de Tchebycheff* est une mesure de la distance entre z^1 et z^2 dans $\mathbb{R}^K$, définie par :

$$\| z^1 - z^2 \|_T = \max_{i=1,\ldots,K} (| z_i^1 - z_i^2 |).$$

Afin de déterminer les optima de Pareto en utilisant cette métrique, BOWMAN [23] a utilisé un point de référence particulier appelé *point de Tchebycheff*. Soit $z^* \in \mathbb{R}^K$ tel que z^* soit une solution optimale du problème de minimisation de K critères par rapport à l'ordre lexicographique $Z_1 \rightarrow Z_2 \rightarrow \cdots \rightarrow Z_K$. C'est-à-dire que le vecteur z^* tel que $z_i^* = \min_{x \in X^{i-1}} (Z_i(x))$ avec $X^i = \{x \in X^{i-1} / Z_i(x) = \min_{x' \in X^{i-1}} (Z_i(x'))\}$ et $X^0 = X$. $\forall \theta = (0, \theta_2, \ldots, \theta_K) \in \mathbb{R}^K$, $(z^* - \theta)$ est appelé *un point de Tchebycheff*.

Théorème 1.3.4. *[23]*
Si $x^0 \in X$ est un optimum de Pareto strict alors $\exists \theta^* = (0, \theta_2^*, \ldots, \theta_K^*) \in \mathbb{R}^K$ tel que x^0 soit une solution optimale du problème (P_θ) suivant :

$$(\mathrm{P}_\theta) \qquad \begin{cases} \min \ \| Z(x) - (z^* - \theta^*) \|_T \\ \quad x \in X. \end{cases}$$

Géométriquement, le problème (P_θ) peut être interprété en utilisant les courbes de niveau (FIG. 1.6). Soient z^* et θ fixés, $X_=(a) = \{x \in X / \ \| \ Z(x) - (z^* - \theta^*) \ \|_T = a\}$ et $L_=(a) = Z(X = (a))$. Résoudre (P_θ) est équivalent à déterminer la courbe de niveau de la valeur minimale g^* tel que $L_=(g^*) \neq \emptyset$. Les solutions de $X_=(g^*)$ sont alors celles de (P_θ).

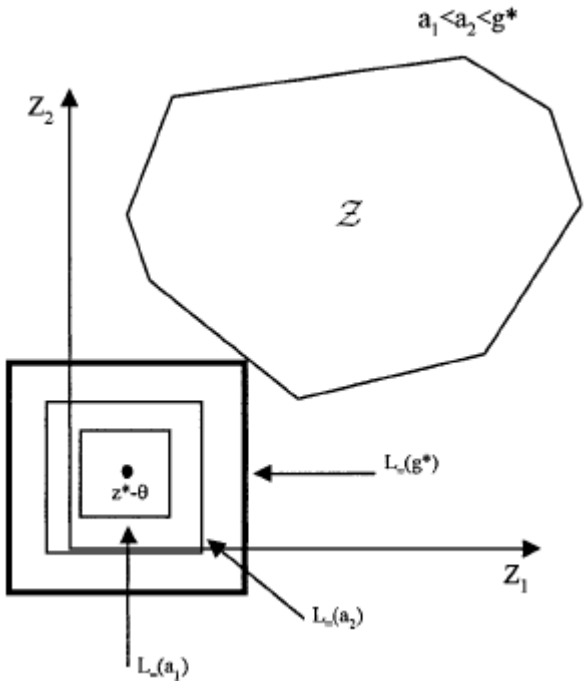


FIG. 1.6 – Interprétation géométrique du problème (P_θ).

L'utilisation de cette métrique dans un algorithme de résolution est reliée à la position du point de Tchebycheff $(z^* - \theta)$. Dans un algorithme interactif, une première solution peut être proposée au décideur et qui correspond à un point de Tchebycheff. Par les instructions données par celui-ci, ce point est varié en répétant le processus, jusqu'à l'obtention d'une solution satisfaisante. Dans un algorithme à posteriori, on doit y avoir une procédure permettant de varier $(z^* - \theta)$ et éliminer toutes les solutions dominées de l'ensembles des solutions obtenues.

Remarque 1.3.1. *[188]*
Si des poids λ_i sont contribués aux critères, il est possible d'utiliser une généralisation de la métrique de Tchebycheff appelée « métrique pondérée de Tchebycheff » ([23]). Il est encore possible d'utiliser une forme plus générale appelée « métrique pondérée augmentée de Tchebycheff » dont le résultat de base est intéressant mais difficile à appliquer

1.3.5 Détermination par l'approche du but à atteindre (Goal attainment)

L'approche du but à atteindre est similaire à celles utilisant la métrique de Tchebycheff ([203] et [77]). Elle nécessite la définition d'un but, pour les critères, et on cherche la solution qui lui approche au mieux. La différence entre cette approche et celles utilisant la métrique de Tchebycheff est la manière dont la solution est recherchée.

Théorème 1.3.5. *[203], [77]*
$x^0 \in X$ est un optimum de Pareto faible si et seulement si $\exists z^{ref} \in \mathbb{R}^K$ un point de référence et $w \in \mathbb{R}^K_{+*}$ un vecteur des poids tel que x^0 soit une solution optimale du problème $(P_{(z^{ref},w)})$ suivant :

$$(\mathrm{P}_{(\mathrm{z^{ref}},\mathrm{w})}) \qquad \begin{cases} \max\ g(Z(x)) \quad avec \quad g(Z(x)) = \min\limits_{i=1,\dots,K}(\frac{1}{w_i}(z_i^{ref} - Z_i(x))) \\ \qquad x \in X. \end{cases}$$

Une interprétation géométrique du problème $(P_{(z^{ref},w)})$ est illustrée par la Fig. 1.7. Deux cas peuvent être représentés dépendant de la position du point z^{ref}. Dans le premier cas, z^{ref} ne correspond à aucune solution réalisable. La solution de $(P_{(z^{ref},w)})$ est équivalente à la projection du point z^{ref} sur la frontière de compromis dans une direction spécifiée par les valeurs des poids w_i. Dans le deuxième cas, le résultat est identique sauf que z^{ref} correspond à une ou plusieurs solutions réalisables.

1.3.6 Programmation par buts (Goal programming)

Les origines de la méthode remontent aux travaux de [32] et [31]. Elle se rapproche beaucoup de la méthode du but à atteindre. La principale différence est qu'après avoir transformé la forme du problème d'optimisation, on se retrouve avec des contraintes d'égalité et non plus des contraintes d'inégalité. La démarche est la suivante :

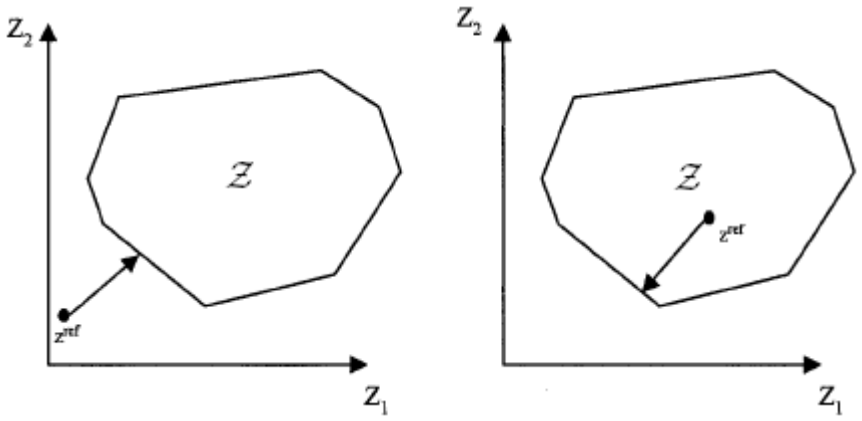


FIG. 1.7 – Interprétation géométrique du problème $(P_{(z^{ref},w)})$.

- On choisit un vecteur de fonctions objectifs initial Z ;
- À chaque objectif, on associe deux nouvelles variables que l'on appelle les « déviations » par rapport au vecteur de fonctions objectifs initial fixé ;
- On minimise en suite une des deux variables. Le choix de la variable se fait en fonction du type de dépassement que l'on veut (au dessus ou au dessous de l'objectif que l'on s'est fixé).

On part du problème (PM). On choisit un vecteur de fonctions objectifs initial $Z \in \mathbb{R}^K$. En suite, on associe un ensemble de variables d_i^+ et d_i^- à chaque fonction objectif . On obtient alors le problème suivant :

$$\begin{cases} \min\ (d_1^+ \text{ ou } d_1^-, \ldots, d_K^+ \text{ ou } d_K^-) \\ \quad z_1(x) = Z_1 + d_1^+ - d_1^-; \\ \quad \vdots \\ \quad z_K(x) = Z_K + d_K^+ - d_K^-; \\ \quad x \in X. \end{cases}$$

Les variables de déviation que l'on cherche à minimiser doivent respecter certaines contraintes :

$$d_i^+ \text{ et } d_i^- \geq 0,$$

$$d_i^+ . d_i^- = 0 \text{ avec } i \in \{1, \ldots, K\}.$$

En fonction de la façon dont on désire atteindre le but Z, on cherchera à minimiser différentes combinaisons des coefficients d_i^+ et d_i^-. Ces combinaisons sont réunies dans le Tab. 1.1 suivant :

Type	Valeur de la déviation	Variable
On désire atteindre par valeurs supérieures l'objectif i	Positive	d_i^+
On désire atteindre par valeurs inférieures l'objectif i	Négative	d_i^-
On désire atteindre l'objectif sans dépassement	Nulle	$d_i^+ + d_i^-$

Tab. 1.1 – Différentes combinaisons de déviations.

1.3.7 Autres méthodes pour la détermination des optima de Pareto

Autres méthodes pour la détermination des optima de Pareto existent dans la littérature. À titre d'exemple, il est possible d'utiliser une métrique différente à celles présentées dans les sections précédentes. Principalement, cette section est vouée à la présentation des approches sans aucune agrégation permise où un ordre lexicographique entre les critères est défini. D'autres méthodes nous permettent de déterminer les optima de Pareto existants.

1.3.7.1 L'utilisation d'un ordre lexicographique

Une technique fréquemment utilisée pour minimiser plusieurs critères consiste en la définition d'un ordre d'optimisation. Ce type de problème apparaît lorsqu'aucun compromis entre les critères n'est permis. Elle concerne un problème d'optimisation par rapport à un ordre lexicographique, défini sans aucune perte de généralité par les indices des critères, $Z_1 \rightarrow Z_2 \cdots \rightarrow Z_K$, et noté $\min_{Lex}(Z)$. Déterminer une solution optimale x^0 de $\min_{Lex}(Z)$ est équivalent à définir une solution $x^0 \in X^K$ avec :

$$X^1 = \{x^0 \in X / Z_1(x^0) = \min_{x \in X}(Z_1(x))\},$$

$$X^2 = \{x^0 \in X^1 / Z_2(x^0) = \min_{x \in X^1}(Z_2(x))\}, \ldots,$$

$$X^K = \{x^0 \in X^{K-1} / Z_K(x^0) = \min_{x \in X^{K-1}}(Z_K(x))\}.$$

Chaque critère Z_i est borné inférieurement sur chaque ensemble X^{i-1} et $X \neq \emptyset$, est une condition nécessaire est suffisante de l'existence d'une solution pour le problème $\min_{Lex}(Z)$. [181]

Propriété 1.3.7.1. *[188]*

1. $\forall x^0 \in X^k$, $1 \leq k \leq K$, x^0 *est un optimum de Pareto faible.*
2. $\forall x^0 \in X^K$, x^0 *est un optimum de Pareto strict.*

1.3.7.2 L'utilisation d'un ordre lexicographique avec des buts

Une approche issue du problème $\min_{Lex}$ consiste à considérer que les critères sont préservés par rapport à l'ordre $Z_1 \rightarrow Z_2 \cdots \rightarrow Z_K$ et que pour chaque critère est associé un but à atteindre. Ainsi, on ne cherche pas la valeur minimale pour chaque critère dans l'ensemble X^i, mais une solution plus proche au but désiré à atteindre. Ce problème noté $\min_{Lexobj}$ est formulé de la manière suivante : Soit z^{ref} un vecteur de référence. La détermination d'une solution optimale x^0 du problème $\min_{Lexobj}(Z)$ est équivalente à trouver une solution $x^0 \in X^K$ avec :

$$X^1 = \{x^0 \in X / |Z_1(x^0) - z_1^{ref}| = \min_{x \in X}(|Z_1(x) - z_1^{ref}|)\},$$

$$X^2 = \{x^0 \in X^1 / |Z_2(x^0) - z_2^{ref}| = \min_{x \in X^1}(|Z_2(x) - z_2^{ref}|)\}, \ldots,$$

$$X^K = \{x^0 \in X^{K-1} / |Z_K(x^0) - z_K^{ref}| = \min_{x \in X^{K-1}}(|Z_K(x) - z_K^{ref}|)\}.$$

Dans la définition du problème $\min_{Lexobj}(Z)$, la distance entre le point de référence z^{ref} et l'ensemble de vecteurs de $\mathcal{Z}$ est mesurée par la métrique L_∞ [1] dans $\mathbb{R}$.

[1] L_∞ : Norme pondérée de Tchebycheff

CHAPITRE 2

L'Ordonnancement Multicritère

Dans ce chapitre, nous présentons les problèmes d'ordonnancement d'une manière générale. En suite une approche pour la résolution des problèmes d'ordonnancement multicritère. Cette approche est représentée par trois phases : Modélisation, prise en compte des critères et résolution à l'aide de procédures spécifiques. Nous clôturons ce chapitre par une extension de la notation usuelle des problèmes d'ordonnancement ainsi que des résultats de la complexité.

2.1 Notions préliminaires en ordonnancement

2.1.1 Définition

Dans un problème d'ordonnancement, quatre notions fondamentales interviennent. Ce sont les travaux (jobs ou tâches), les ressources, les contraintes et les objectifs. Un travail est défini par un ensemble d'opérations qui doivent être exécutées. Une ressource est un moyen matériel ou humain à disposition pour la réalisation d'un travail. Les contraintes représentent les limites imposées par l'environnement. Tandis que l'objectif est le critère d'optimisation. [87]

Plusieurs définitions d'un problème d'ordonnancement sont données dans la littérature, nous indiquons ici celle proposée dans [62] :

> *« Le problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches compte tenu de contraintes temporelles (délais, contraintes d'enchaînement, etc.) et de contraintes portant sur l'utilisation et la disponibilité de ressources requises ».*

Dans les problèmes d'ordonnancement classiques, deux hypothèses importantes sont communément considérées :

a) À chaque instant, une machine ne peut exécuter qu'une seule tâche.

b) À chaque instant, une tâche peut être exécutée par une machine au plus.

Notons que ces hypothèses peuvent être levées pour des problèmes d'ordonnancement plus spécifiques.

2.1.2 Classification des problèmes d'ordonnancement

Plusieurs classifications des problèmes d'ordonnancement sont proposées dans la littérature. À titre d'exemple, celle introduite dans [137] et basée sur la notion structurante de « *gamme opératoire* ». En effet, une gamme opératoire, souvent utilisée comme critère de classification, impose un ordre de passage des produits sur les machines, et donc un ordre des opérations associées à chaque travail. Cette typologie, proposée dans [17], peut être synthétisée et généralisée ainsi que l'illustre la Fig. 2.1.

Généralement, les problèmes d'ordonnancement se divisent en deux grandes catégories selon le nombre d'opérations nécessaires à la réalisation de chaque travail. La première catégorie regroupe les problèmes pour lesquels chaque travail nécessite une seule opération, la deuxième regroupe ceux pour lesquels chaque travail requiert plusieurs opérations.

La première catégorie se subdivise à son tour en plusieurs types de problèmes, en fonction de la configuration de machines considérées :

- *Machine unique* : Tous les travaux sont appelés à être exécutés sur une même machine [129], [123].
- *Machines spécialisées* : Plusieurs machines, chacune étant spécialisée pour l'exécution de certains travaux.

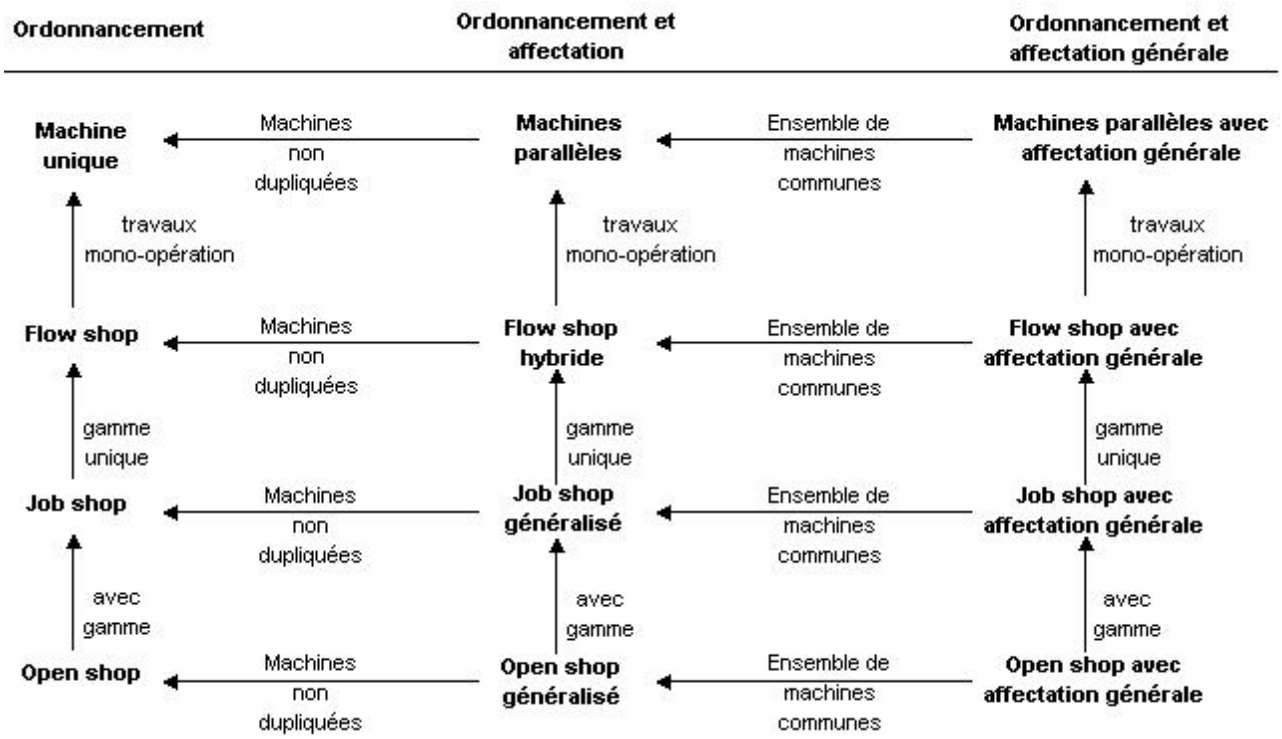


FIG. 2.1 – Une typologie des problèmes d'ordonnancement.

- *Machines parallèles* : Plusieurs machines, qui remplissent toutes les mêmes fonctions. Dans ce cas, on distingue trois modèles différents de machines en fonction des vitesses de celles-ci :
  - *Machines identiques* : Toutes les machines présentent la même vitesse d'exécution quelle que soit le travail [118].
  - *Machines uniformes* : La vitesse d'une machine diffère d'une autre par un coefficient de proportionnalité [88].
  - *Machines générales* : Chaque machine présente une vitesse particulière pour chaque travail ; la durée opératoire d'un travail dépend donc du travail et de la machine qui l'exécute [25].

Les problèmes de la deuxième catégorie sont dits *problèmes d'atelier* du fait de nécessité de passage de chaque travail sur deux ou plusieurs machines spécialisées. Ils sont généralement spécifiés par la donnée de m machines et de n travaux composés chacun de m opérations ; chaque opération devant être exécutée par une machine différente. Trois sous-classes de problèmes sont alors différenciées selon le mode de passage des opérations sur les différentes

machines, à savoir :

- *Flow-shop* : L'ordre de passage des opérations sur les machines est le même pour tous les travaux [85].
- *Job-shop* : Chaque travail a un ordre propre de passage des opérations sur les machines [155], [85].
- *Open-shop* : L'ordre de passage des opérations sur les machines est libre [84].

À partir de ces modèles de base, d'autres modèles d'ordonnanement peuvent être définis afin de répondre à des problèmes industriels spécifiques :

- *Flow shop de permutation* : S'il existe une contrainte selon laquelle toutes les machines doivent exécuter les n travaux dans le même ordre [114].
- *Job shop à machines dupliquées* : Où chaque travail a un ordre propre de passage des opérations sur les machines et où chaque opération est réalisée par une machine à sélectionner dans un ensemble.
- *Flow shop hybride* : Une même opération peut être exécutée par une machine disponible en plusieurs exemplaires [20].
- etc.

S'inspirant de cette typologie, nous utilisons couramment, un formalisme issu des travaux de [115] et [40], permettant de distinguer les problèmes d'ordonnancement entre eux et les classer. Ce formalisme proposé dans [88], puis repris par [19], comporte trois champs $\alpha/\beta/\gamma$.

Le champ α : Organisation des ressources

Il est spécifié par la concaténation de deux éléments : $\alpha = \alpha_1\alpha_2$

- Le paramètre α_1 représente la configuration de machines utilisées : $\alpha_1 \in \{\emptyset, P, Q, R, F, O, J\}$.

  $\alpha_1 = \emptyset$: Une seule machine est utilisée.

  $\alpha_1 = P$: Plusieurs machines identiques sont disponibles. (*i.e.* les ressources sont composées de machines travaillant suivant la même cadence, disposées en parallèle et pouvant exécuter tous les travaux).

$\alpha_1 = Q$: Plusieurs machines parallèles uniformes sont disponibles. (*i.e.* les cadences des machines sont différentes (selon un facteur de proportionnalité), mais restent indépendantes des travaux).

$\alpha_1 = R$: Plusieurs machines générales sont disponibles. (*i.e.* les cadences des machines sont différentes et dépendent des travaux exécutés).

$\alpha_1 = F$: Plusieurs machines spécialisées fonctionnant en flow-shop. (*i.e.* les travaux sont décomposés en plusieurs opérations qui doivent être exécutées sur l'ensemble des machines, celles-ci étant disposées en série pour un même routage).

$\alpha_1 = O$: Plusieurs machines spécialisées fonctionnant en open-shop. (*i.e.* les travaux sont décomposés en plusieurs opérations qui doivent être exécutées sur l'ensemble des machines sans restriction sur les routage des travaux).

$\alpha_1 = J$: Plusieurs machines spécialisées fonctionnant en job-shop. (*i.e.* les travaux sont décomposés en plusieurs opérations qui doivent être exécutées sur l'ensemble des machines, mes peuvent avoir des routages différents).

- Le paramètre α_2 permet de préciser le nombre de machines composant l'atelier ; il peut être égale à vide ou à un entier m. Dans le premier cas, cela signifie que le nombre de machines est quelconque. Dans le deuxième cas, cela signifie que l'atelier est composé de m machines ($m > 0$).

Le champ β : Contraintes et caractéristiques du système

Il est formé de huit sous champs : $\beta = \beta_1\beta_2\beta_3\beta_4\beta_5\beta_6\beta_7\beta_8$.

- $\beta_1 \in \{\emptyset, prem\}$ permet de préciser le mode d'exécution. $\beta_1 = \emptyset$ indique le mode sans préemption et $\beta_1 = prem$ indique le mode avec préemption.

- $\beta_2 \in \{\emptyset, res\}$ caractérise les ressources supplémentaires nécessaires à l'exécution d'un travail (outils, ressources de transport). $\beta_2 = \emptyset$ indique qu'il n'y a pas de ressources complémentaires. $\beta_2 = res\ \lambda\delta\rho$ indique la nécessité des ressources complémentaires ; cette nécessité est détaillée par les valeurs de λ, δ et ρ.

- $\beta_3 \in \{\emptyset, prec, tree, chain\}$ précise un type de précédence entre travaux, c'est-à-dire le fait qu'un travail doit être exécuté avant un autre. La valeur $\emptyset$ indique que les travaux sont indépendants. Les valeurs *prec*, *tree*, *chain* indiquent l'existence, respectivement, d'une

relation de précédence générale, d'une relation de précédence sous forme d'arbre et d'une relation de précédence sous forme de chaîne.

- $\beta_4 \in \{\emptyset, r_i\}$ décrit les dates de disponibilité (*i.e.* dates au plus tôt) des différents travaux dans le système. Ces dates peuvent être identiques et égales à zéro pour tous les travaux ($\beta_4 = \emptyset$) ou différentes suivant les travaux ($\beta_4 = r_i$).

- $\beta_5 \in \{\emptyset, p_i = p, \underline{p} \leq p \leq \overline{p}\}$ détaille les durées opératoires des différents travaux. Ces durées peuvent être fonction de la machine. Différentes restrictions peuvent également être considérées pour simplifier certains problèmes.

  $\beta_5 = \emptyset$: Les travaux ont des durées opératoires arbitraires.

  $\beta_5 = p_i = p$: Tous les travaux ont des durées opératoires égales à p.

  $\beta_5 = \underline{p} \leq p \leq \overline{p}$: Les durées opératoires des travaux sont comprises entre $\underline{p}$ et $\overline{p}$.

- $\beta_6 \in \{\emptyset, d_i, \widetilde{d_i}\}$ indique les éventuelles dates échues (ou dates de fin au plus tard) des travaux.

  $\beta_6 = \emptyset$: Les travaux n'ont de dates échues ou le critère à optimiser dépend des dates échues.

  $\beta_6 = d_i$: Chaque travail a une date échue de fin d'exécution sous peine de pénalisation.

  $\beta_6 = \widetilde{d_i}$: Chaque travail a une date échue impérative (date limite) qu'il faut absolument respecter.

- $\beta_7 \in \{\emptyset, s, s_i, s_{ij}, nwt, nmit\}$ permet de spécifier des contraintes temporelles sur les enchaînements de travaux. Ces contraintes sont très souvent introduites afin de mieux présenter les problèmes réels. Il est parfois nécessaire de considérer un temps improductif entre l'exécution de deux travaux différents sur une même machine pour présenter les changements et les réglages d'outils. Ces temps de changement peuvent être constants (s), fonction du nouveau travail (s_i) ou bien fonction de l'enchaînement des deux travaux (s_{ij}). Également, pour le cas des industries où l'on manipule de la matière de fusion, on doit pouvoir imposer que toutes les opérations d'un travail soient exécutées sans temps d'attente ($\beta_7 = nwt$ (no-wait)). Si on a besoin de spécifier que les machines, une fois qu'elles commencent à travailler, ne doivent pas s'arrêter qu'après avoir terminé tous les travaux, alors β_7 prend la valeur *no-idle*. La contrainte *nmit* (no machine idle time) indique que les temps morts sur la machine ne sont pas admis.

- Enfin, le paramètre $\beta_8 \in \{\emptyset, M_j\}$ indique, dans le cas de machines parallèles, des restrictions sur la polyvalence des machines. L'ensemble M_j représente l'ensemble des machines capables de réaliser le travail J_i. Lorsque β_8 est vide, toutes les machines sont capables d'exécuter tous les travaux.

Le champ γ : Critère à optimiser

Les critères les plus utilisés sont :

$\gamma = C_{\max}$: Makespan. Date de sortie du système (le dernier travail). ($C_{\max} = \max\limits_{1\leq i\leq n}\{C_i\}$ où C_i est la date de fin d'exécution du travail J_i),

$\gamma = \max\limits_{1\leq i\leq n}\{F_i\}$ (ou $F_{\max}$) : Flowtime. Avec $F_i = C_i - r_i$, c'est le temps de séjour du travail J_i dans l'atelier et r_i est la date de son disponibilité,

$\gamma = \max\limits_{1\leq j\leq m}\{I_j\}$ (ou $I_{\max}$), où I_j est la somme des temps d'oisiveté sur la machine M_j,

$\gamma = L_{\max}$: Décalage temporelle maximal. Ce critère mesure la plus grande violation des dates échues ($L_{\max} = \max\limits_{1\leq i\leq n}\{L_i = C_i - d_i\}$ où d_i est la date de fin au plus tard),

$\gamma = \max\limits_{1\leq i\leq n}\{T_i\}$ (ou $T_{\max}$), avec $T_i = \max\{0, C_i - d_i\}$: Le retard maximal (maximum tardiness),

$\gamma = \max\limits_{1\leq i\leq n}\{E_i\}$ (ou $E_{\max}$), avec $E_i = \max\{0, d_i - C_i\}$: L'avance maximale (maximum earliness),

Soit w_i le poids associé au travail J_i ;

$\gamma = \sum_{i=1}^{n} C_i$ (ou $\overline{C}$) : Somme des dates de fin d'exécution. La date C_i peut être pondérée et un deuxième critère pourra être défini $\gamma = \sum w_i C_i$ (ou $\overline{C}_w$),

$\gamma = \sum_{i=1}^{n} F_i$ (ou $\overline{F}$). Optimiser ce critère est équivalent à optimiser le critère $\overline{C}$. C'est le même cas pour le critère $\gamma = \sum w_i F_i$ (ou $\overline{F}_w$),

$\gamma = \sum_{i=1}^{n} T_i$ (ou $\overline{T}$). Ce critère désigne le retard total des travaux. Il peut être pondéré et un deuxième critère pourra être défini $\gamma = \sum w_i T_i$ (ou $\overline{T}_w$),

$\gamma = \sum_{i=1}^{n} E_i$ (ou $\overline{E}$). Ce critère désigne l'avance totale des travaux. Il peut être pondéré et un deuxième critère pourra être défini $\gamma = \sum w_i E_i$ (ou $\overline{E}_w$),

$\gamma = \sum_{i=1}^{n} U_i$: Somme du nombre de travaux terminés avec retard ($U_i = 1$ si le travail J_i est terminé après sa date de fin au plus tard). Un U_i peut être pondéré et un deuxième critère pourra être défini $\gamma = \sum w_i U_i$.

Remarquons enfin que plusieurs extensions ont été proposées pour les champs β et γ (voir [19], [188]) afin de prendre en compte des catégories de problèmes particuliers (stochastiques, répétitifs, cycliques, dynamiques, multicritères, etc.) et qu'il est probable que dans l'avenir, d'autres extensions soient encore créées.

2.1.3 Diagramme de Gantt

Nous présentons souvent un ordonnancement par un « *Diagramme de Gantt* ». Un tel diagramme met en évidence l'occupation des machines par les différents travaux ainsi que les temps morts. La FIG. 2.2 présente un ordonnancement pour un problème à trois machines et trois travaux.

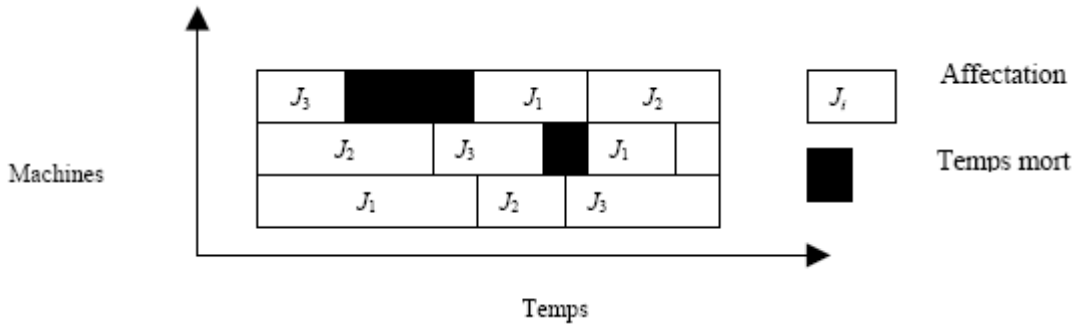


FIG. 2.2 – Exemple de diagramme de Gantt.

2.1.4 Quelques notions fondamentales

Les notions présentées dans cette section se réfèrent à la caractérisation des ensembles dominants. Un sous-ensemble d'ordonnancements est dit « *dominant* » par rapport à un critère s'il contient au moins un ordonnancement optimal relativement à ce critère. La définition 2.1.1 suivante introduit la notion de critère régulier dans le cas d'un problème de minimisation.

Définition 2.1.1. [188]
Soit $\mathcal{S}$ l'ensemble des solutions d'ordonnancements. D'une façon générale, un critère Z est dit

« *régulier* » si, et seulement si, Z est une fonction croissante des dates de fin des travaux. Autrement dit, si et seulement si :

$$\forall x, y \in \mathcal{S}, C_i(x) \leqslant C_i(y), \forall i = 1, \ldots, n,$$

$$\Rightarrow Z(C_1(x), \ldots, C_n(x)) \leqslant Z(C_1(y), \ldots, C_n(y)).$$

De cette définition, il est possible de déduire que si un ordonnancement x est meilleur qu'un ordonnancement y, alors il existe au moins un C_i tel que $C_i(x)$ est strictement inférieur à $C_i(y)$.

Corollaire 2.1.1. *[188]*
Les critères $C_{\max}$, $\overline{C}$, $\overline{C}_w$, $T_{\max}$, $\overline{T}$, $\overline{T}_w$, $\overline{U}$ *et* $\overline{U}_w$ *sont réguliers. Les critères* $I_{\max}$, $E_{\max}$, $\overline{E}$, $\overline{E}_w$ *ne sont pas réguliers.*

Par la suite, nous rappelons certaines définitions et propriétés essentielles qui permettent de caractériser le type des solutions recherchées et ainsi restreindre la taille de l'espace de recherche dans un problème d'ordonnancement. Plus précisément, l'ensemble des ordonnancement admissibles, respectant toutes les contraintes du problème, peut être décomposé en plusieurs sous-classes définies ci-dessous [62].

Ordonnancement avec insertion de temps mort

Définition 2.1.2. Un ordonnancement $x \in \mathcal{S}$ appartient à la classe des ordonnancement avec insertion de temps mort si avant d'ordonnancer chaque opération, les machines sont prêtes à paresser une période positive ou nulle.

Ordonnancement semi-actif

Définition 2.1.3. Soit $x \in \mathcal{S}$ un ordonnancement de $\mathcal{S}$ et $\mathcal{S}_x$ l'ensemble des ordonnancements dont les séquences d'opérations sur les machines sont identiques. Un ordonnancement x appartient à la classe des ordonnancements semi-actif si et seulement si $\nexists y \in \mathcal{S}_x$ tel que $\forall i \in \{1, \ldots, n\}$, $C_i(y) \leqslant C_i(x)$ et $\exists i \in \{1, \ldots, n\} | C_i(y) < C_i(x)$.

Dans un ordonnancement semi-actif, il est possible d'avancer une opération sans modifier la séquence des opérations sur la ressource : Toutes les opérations sont calées, soit sur l'opération qui la précède dans sa gamme, soit sur l'opération qui la précède sur la machine utilisée.

Ordonnancement actif

Définition 2.1.4. Un ordonnancement $x \in \mathcal{S}$ appartient à la classe des ordonnancements actifs si et seulement si $\nexists y \in \mathcal{S}$ tel que $\forall i \in \{1, \ldots, n\},\ C_i(y) \leqslant C_i(x)$ et $\exists i \in \{1, \ldots, n\} | C_i(y) < C_i(x)$.

En conséquence, dans un ordonnancement actif, il est impossible d'avancer une opération, sans reporter le début d'une autre opération.

Ordonnancement non retardé

Définition 2.1.5. Un ordonnancement $x \in \mathcal{S}$ appartient à la classe des ordonnancements non retardés si et seulement si aucune opération n'est mise en attente alors qu'une machine est disponible pour l'exécuter.

La FIG. 2.3 fait apparaître que les ordonnancements non retardés sont inclus dans le sous-ensemble des ordonnancements actifs qui sont eux-mêmes inclus dans le sous-ensemble des ordonnancements semi-actifs.

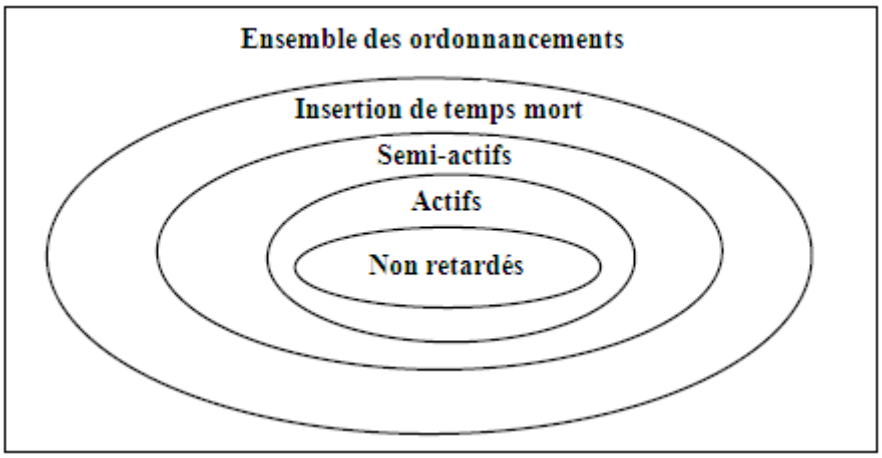


FIG. 2.3 – Inclusion des classes d'ordonnancements.

La propriété majeur suivante [9] lie les critères réguliers à la classe des ordonnancements actifs :

Proposition 2.1.2. *L'ensemble des ordonnancements semi-actifs est dominant dans les problèmes d'optimisation d'un critère régulier et le sous-ensemble des ordonnancements actifs est le plus petit ensemble dominant.*

En conséquence, dans le cas d'un critère régulier, la recherche d'une solution optimale peut être limitée à l'ensemble des ordonnancements actifs, ce qui restreint ainsi la taille de l'espace de recherche.

2.2 Complexité

Nous nous intéressons dans cette section au cadre mathématique dans lequel les problèmes peuvent être classés en problèmes faciles ou difficiles. Plusieurs ouvrages [205] et [19] développent la théorie de la complexité.

Un problème de décision est un énoncé auquel la réponse peut être uniquement *oui* ou *non*. Les problèmes de décision sont de deux types : Les problèmes *décidables* et les problèmes *indécidables*.

Un problème est dit *indécidable* s'il est impossible de décrire un algorithme qui permet de décider de la réponse pour tous les cas de figures.

Un problème de décision $P1$ est dit *réductible* à un autre problème de décision $P2$ (on note $P1 \alpha P2$) s'il existe une fonction polynomiale f qui transforme chaque énoncé de $P1$ en un autre énoncé de $P2$ de telle manière que la réponse pour $P1$ est *oui* si, et seulement, si la réponse pour $P2$ est *oui*.

Un algorithme est dit *polynomial* si sa complexité temporelle est bornée par un $O(p(x))$ où p est un polynôme et x est la longueur d'une instance du problème. Il est dit *pseudo-polynomial* si sa complexité est bornée par un polynôme en fonction de la taille de la plus grande instance du problème.

Un problème de décision décidable est dit *non polynomial* $\mathcal{NP}$ s'il existe un algorithme polynomial qui permet de reconnaître une instance positive de ce problème (qui a la réponse *oui*). Parmi les problèmes $\mathcal{NP}$, on distingue, d'une part, les problèmes *polynomiaux* et, d'autre part, les problèmes dits $\mathcal{NP}$*-complets* :

- Un problème de décision est dit *polynomial* (on dit aussi appartenant à la classe $\mathcal{P}$) s'il existe un algorithme polynomial en fonction de la taille des données qui permet de le résoudre. Un tel problème est dit *facile*.
- Un problème $\mathcal{NP}$*-complet* est un problème $\mathcal{NP}$ tel que tout problème $\mathcal{NP}$ est réductible polynomialement en ce problème.

Pour démontrer qu'un problème Q est $\mathcal{NP}$-complet, il faudra montrer qu'il est de la classe $\mathcal{NP}$ et qu'il existe un problème R connu pour être $\mathcal{NP}$-complet tel que $R\alpha Q$. On note aussi qu'un problème $\mathcal{NP}$-complet peut être *$\mathcal{NP}$-complet au sens fort* ou *au sens faible* :

- Un problème P est dit *$\mathcal{NP}$-complet au sens faible* s'il est $\mathcal{NP}$-complet et qu'il existe un algorithme pseudo-polynomial pour le résoudre.
- Un problème P est dit *$\mathcal{NP}$-complet au sens fort* s'il appartient à la classe $\mathcal{NP}$ et qu'il existe un problème Q connu pour être *$\mathcal{NP}$-complet au sens fort* tel que $Q\alpha P$.

Ainsi, il est nécessaire de connaître les problèmes classiques connus pour être $\mathcal{NP}$-complets. Le premier problème qui a été prouvé, en 1971, comme étant un problème $\mathcal{NP}$-complet au sens fort [41] est le problème de satisfiabilité (SAT) qui peut être défini comme suit : Étant donnés un ensemble de n variables booléennes $x_1, x_2, \ldots, x_n$ et un ensemble de m clauses $c_1, c_2, \ldots, c_m$ (une clause est un sous-ensemble des $2n$ littéraux $x_1, x_2, \ldots, x_n, \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ où $\bar{x}_i$ désigne la variable complémentaire de x_i, la question posée est : "Existe-t-il une affectation de la valeur *vrai* ou *faux* à chacune des variables x_i de sorte que pour chacune des m clauses, au moins l'un des littéraux la constituant ait la valeur *vrai* ?" Dans ce qui suit, nous présentons d'autres problèmes $\mathcal{NP}$-complet dont la $\mathcal{NP}$-complétude n'a pu être démontrée que grâce à l'existence d'un premier problème $\mathcal{NP}$-complet. Ce sont les six problèmes de base exposés par GAREY et JOHNSON dans [75] :

- **3-satisfiabilité :** Étant donné un ensemble de clauses de dimension 3, construites à partir d'un ensemble fini de variables, existe-t-il une affectation de ces variables qui satisfait toutes les clauses ?
- **Couplage de dimension 3 :** Étant donné un ensemble M de triplets ($M \subseteq X \times Y \times Z$, où X, Y et Z sont des ensembles disjoints et de même cardinalité q), existe-t-il un ensemble M' tel que $M' \subseteq M$, $|M'| = q$ et les triplets de M' sont deux à deux disjoints ?

- **Recouvrement :** Étant donné un graphe $G = (V, E)$ et un entier k, existe-t-il $X \subseteq V$ tel que $|X| \leqslant k$ et toute arête de G a au moins une de ses extrémités dans X ?
- **Clique :** Étant donné un graphe $G = (V, E)$ et un entier k, existe-t-il $X \subseteq V$ tel que $|X| \geqslant k$ et tous les sommets de X sont deux à deux adjacents ?
- **Cycle Hamiltonien :** Étant donné un graphe $G = (V, E)$, existe-t-il un cycle passant une fois et une seule par chacun des sommets de V ?

– **Partition :** Étant donnés n entiers positifs $s_1, s_2, \ldots, s_n$, existe-t-il un sous-ensemble $J \subseteq I = \{1, 2, \ldots, n\}$ tel que $\sum_{i \in J} s_i = \sum_{i \in I \setminus J} s_i$?

Un problème d'optimisation est un problème pour lequel la réponse à une instance est une solution ayant une valeur optimale pour une fonction objectif définie. Il est toujours possible de faire correspondre un problème d'optimisation à un problème de décision. Dans le cas, par exemple, de la minimisation d'une fonction objectif, le problème de décision peut être posé de la manière suivante : Le problème admet-t-il une solution, dont la fonction objectif, est inférieure ou égale à un seuil défini y ? De même, dans le cas de la maximisation d'une fonction objectif, le problème de décision peut être posé de la manière suivante : Le problème admet-t-il une solution, dont la fonction objectif est supérieure ou égale à un seuil défini y ?

Remarque 2.2.1. *Un problème d'optimisation est dit* $\mathcal{NP}$-difficile *si le problème de décision qui lui correspond est* $\mathcal{NP}$-complet.

2.3 Méthodes classiques de résolution

Les problèmes d'ordonnancement de la classe $\mathcal{P}$ disposent donc d'algorithmes polynomiaux spécifiques qui permettent de les résoudre. Pour appréhender les problèmes $\mathcal{NP}$-difficiles, différentes méthodes de résolution (exactes ou heuristiques) sont proposées dans la littérature. On va passer à la présentation les méthodes les plus connues en les classant en *méthodes dédiées* et *méthodes génériques*.

2.3.1 Méthodes dédiées

Dans les méthodes dédiées, on classe les méthodes qui ont été conçues pour résoudre un problème d'ordonnancement particulier.

2.3.1.1 Règle de tri

Plusieurs règles de tri, optimales ou heuristiques, ont été proposées dans la littérature. Parmi celles les plus traditionnelles, on cite :

- **La règle SPT** (*Shortest Processing Time*), qui consiste à ordonnancer les tâches par ordre croissant de leurs durées de traitement. La règle SPT est optimale lors de la résolution du

problème $1||\overline{C}$ [19]. Conversement, on trouve la **règle LPT** (*longest Processing Time*), qui consiste à classer les tâches par ordre décroissant de leurs durées de traitement.

- **La règle WSPT** (*Weighted Shortest Processing Time*), qui consiste à ordonnancer les tâches dans l'ordre croissant des rapports $\frac{p_i}{w_i}$. Elle résout optimalement le problème $1||\overline{C}_w$.
- **La règle EDD** (*Earliest Due Date*), qui consiste à ordonnancer les tâches selon l'ordre croissant de leurs dates échues d_i. Elle est optimale pour la résolution des problèmes $1||L_{\max}$ [108] et $1||T_{\max}$.

2.3.1.2 Algorithmes de liste

Une généralisation de ces règles à des problèmes avec machines parallèles nécessite l'addition d'une règle d'affectation des tâches aux machines. Dans le cas des machines identiques, nous utilisons généralement la règle d'affectation **FAM** (*First Available Machine*), qui affecte une tâche à la première machine disponible. La règle **SPT-FAM** résout polynomialement le problème $P||\overline{C}$ en considérant les tâches dans l'ordre croissant de leurs durées d'exécution et en les affectant à la première machine disponible. La règle **WSPT-FAM** et **EDD-FAM** sont utilisées dans des algorithmes heuristiques pour la résolution des problèmes $P||\overline{C}_w$ et $P||L_{\max}$, respectivement. Dans le cas des machines uniformes, nous considérons souvent la règle d'affectation **FM** (*Fastest Machine*) qui consiste à affecter une tâche à la machine la plus rapide parmi celles qui sont disponibles. La règle **SPT-FM** résout optimalement le problème $Q||\overline{C}$. Les règles **WSPT-FM** et **EDD-FM** sont utilisées dans des algorithmes heuristiques pour la résolution des problèmes $Q||\overline{C}_w$ et $Q||L_{\max}$, respectivement.

Lorsque la préemption des tâches est autorisée, la règle SPT-FM devient **SRPT-FM** (*Shortest Remaining Processing Time on fastest Machine*) et résout optimalement le problème $Q|pmtn|\overline{C}$. Elle consiste à ordonnancer parmi les tâches restantes, la tâches disponible ayant le plus petit temps de traitement restant et en l'affectant à la machine la plus rapide parmi celles qui sont disponibles, en préemptant lorsque c'est nécessaire. La règle **LRPT-FM** résout optimalement le problème $Q|pmtn|C_{\max}$.

2.3.1.3 Autres méthodes

Les problèmes d'ordonnancement sur machines parallèles recouvrent un ensemble de problèmes de gestion très vaste. L'une des méthodes de résolution élaborées pour appréhender

de tels problèmes est celle de (BRUNO & al.) [25] qui résout optimalement le problème $Q||\overline{C}$. En définissant une matrice de coûts machine-ordre/travaux, l'auteur modélise le problème d'ordonnancement sur machines parallèles uniformes sous la forme d'un problème d'affectation. Il propose ensuite de résoudre ce problème par la méthode hongroise [124] (connue pour les problèmes d'affectation) qui offre une solution à temps polynomial ; la solution est en fait recherchée pour le problème dual.

2.3.2 Méthodes génériques

Contrairement aux méthodes dédiées qui ont été conçues pour des problèmes spécifiques, les méthodes génériques proposent des approches de résolution qui sont applicables de façon générale et beaucoup plus ouverte. Ces méthodes sont classées en *méthodes exactes*, *méthodes heuristiques constructives* et *méthodes heuristiques amélioratrices*.

2.3.2.1 Méthodes exactes

Les trois familles de méthodes exactes sont : La programmation dynamique, la méthode de séparation et évaluation (Branch-and-Bound) et la résolution à partir d'une modélisation analytique. Ces méthodes se caractérisent par un temps de calcul exponentiel, ce qui explique quelles ne sont utilisables que sur des problèmes de petite taille.

2.3.2.1.1 Programmation dynamique

Dans cette famille, un problème de dimension n est décomposé en n problèmes de dimension 1. Le système est composé de n étapes que l'on résout séquentiellement, le passage d'une étape à une autre se fait à partir des lois d'évolution du système et d'une décision [37].

Cette méthode a été introduite par BELLMANN [14]. Son principe d'optimalité est basé sur l'existence d'une équation récursive permettant de décrire la valeur optimale du critère à une étape donnée en fonction de sa valeur à l'étape précédente. Ainsi, pour appliquer la programmation dynamique à un problème combinatoire, le calcul du critère pour un sous-ensemble de taille k nécessite la connaissance de ce critère pour chaque sous-ensemble de taille $k-1$, ce qui porte le nombre de sous-ensembles considérés à 2^n (où n est le nombre d'élément considérés dans le problème), d'où sa complexité exponentielle. Pourtant, pour les problème $\mathcal{NP}$-difficile au sens faible, il est souvent possible de construire un algorithme de

programmation dynamique pseudo-polynomial, pouvant être utilisé pour des problèmes de dimension raisonnable.

2.3.2.1.2 Méthode de séparation et évaluation (Branch-and-Bound)

Cette méthode est basée sur une énumération implicite et intelligente de l'ensemble des solutions réalisables. La *séparation* consiste à décomposer l'ensemble des solutions en plusieurs sous-ensembles qui sont décomposés à leur tour selon une démarche itérative. Ce processus peut ce visualiser sous la forme d'un arbre d'énumération ; les nœuds de l'arbre correspondent aux sous-ensembles et les feuilles correspondent à des solutions réalisables. Pour accélérer la recherche de la solution optimale en évitant l'exploration inutile de certains nœuds, on utilise une procédure d'*évaluation* qui calcule, à chaque niveau de l'arbre, la valeur de chaque nœud et permet ainsi de décider du nœud (sou-ensemble) à développer (décomposer). Par exemple, dans le cas d'un problème de minimisation, cela revient à calculer la borne inférieure pour tous les nœuds fils du niveau considéré et la descente dans l'arbre est poursuivie avec le nœud donnant la borne inférieure. Par ailleurs, une borne supérieure de la solution optimale est calculée et est utilisée pour éviter l'exploration de nœuds dont la valeur de la borne inférieure est supérieure à la valeur de la borne supérieure (certaines branches vont être élaguées). Cette borne supérieure est réactualisée lorsqu'une solution réalisable de valeur inférieure est atteinte.

Enfin, des remontées et descentes par d'autres nœuds de l'arbre sont effectuées tant que la borne calculée est inférieure ou égale à la valeur actualisée de la borne supérieure. Ainsi, l'exploration de certaines branches de l'arbre est évitée, ce qui permet de ne pas énumérer toutes les solutions réalisables. Il faut donc souligner que l'efficacité de la méthode (en terme du nombre de nœuds explorés) est déterminée par la qualité de la borne initiale par la procédure d'évaluation considérée.

2.3.2.1.3 Modélisation analytique et résolution

La modélisation analytique d'un problème permet, non seulement de mettre en évidence l'objectif et les différentes contraintes du problème, mais également de le résoudre. L'idéal est de modéliser le problème sous la forme d'un programme linéaire dont les variables sont réelles. Dans ce cas, il existe un bon nombre de solveurs permettant de le résoudre. Dès que le problème comporte des coûts fixes ou des décisions nécessitant l'utilisation de variables entières, les modèles deviennent beaucoup plus difficiles à résoudre. Il en est de même lorsque

le modèle n'est pas linéaire, mais quadratique par exemple.

Actuellement, plusieurs langages de modélisation existent tel que AMPL, GAMS, LINGO, MPL. Ils permettent d'écrire les programmes linéaires de façon formelle, proche de l'écriture mathématique ; lesquels les programmes peuvent être executer par des solveurs tels que CPLEX, OSL, XPRESS (selon le langage utilisé). Malheureusement, tous les problèmes ne peuvent être résolus par cette approche car la résolution de programmes linéaires avec des variables entières, par exemple, demande souvent beaucoup trop de temps de calcul.

2.3.2.2 Méthodes heuristiques constructives

Pour des problèmes $\mathcal{NP}$-difficiles de grande taille, les méthodes exactes sont peu envisageables à cause de leurs temps de calcul. Il est alors possible d'utiliser des méthodes approximatives qui donnent des solutions certes sous-optimales, mais en un temps de calcul raisonnable. La performance de telles méthodes est généralement calculée par le rapport entre la valeur de la solution fournie et la valeur de la solution optimale, ceci pour le pire des cas ou dans le cadre d'une moyenne de plusieurs instances. si la solutions optimale est non calculable, il est également possible d'étudier expérimentalement le comportement de l'heuristique en comparant ses performances soit à celles d'autres heuristiques, soit à des bornes inférieures de la solution optimale.

Ces méthodes par constructions progressives sont, en définitive, des méthodes itératives où, à chaque itération, une solution partielle est complétée. La plupart de ces méthodes sont des algorithmes gloutons car elles considèrent les éléments (tâches ou machines) dans un certains ordre sans remettre en cause un choix une fois qu'il a été effectué. De principe très simple, ces méthodes permettent de trouver une solution très rapidement.

un premier type de telles méthodes consiste, dans une première phase, à calculer une liste qui donnera l'ordre des prise en compte des éléments. Cette liste construite à priori, à partir d'un critère bien défini, n'est plus remise en cause au cours de l'ordonnancement. La deuxième phase de l'algorithme se réduit à considérer les éléments (tâches ou machines) dans l'ordre de la liste pour construire l'ordonnancement.

Un deuxième type de méthodes consiste à choisir, au cours de la construction, l'affectation d'un travail à une machine en utilisant des règles de priorité. Ces méthodes peuvent également être vue comme des méthodes sérielles dynamiques où les listes sont reconstruites à chaque étape, selon un critère qui peut évoluer dans le temps (le nombre de tâches disponibles, par

exemple). Notons que si le choix de la tâche à rajouter est guidé par un théorème de dominance, cela permet de prouver l'optimalité d'une solution si certaines hypothèses sont vérifiées [155]. Une revue détaillée des règles classiques couramment utilisées en ordonnancement est proposée par (PANWALKER & ISKANDER) dans [154]. Pour chaque problème, de nombreuses règles ont été développées et les articles présentant ces règles sont passés en revue.

2.3.2.3 Méthodes amélioratrices

Ces méthodes sont initialisées par une solution réalisable, calculée soit aléatoirement soit à l'aide de l'une des heuristiques constructives exposées précédemment. Elles recherchent, à chaque itération, une amélioration de la solution courante par des modifications locales. Cet examen se poursuit jusqu'à ce qu'un critère d'arrêt soit satisfait. L'utilisation de ces heuristiques itératives suppose que l'on puisse définir, pour toute solution S, un voisinage de solution, $N(S)$, contenant les solutions voisines (proches dans un certains sens). En général, le voisinage d'une solution est généré en appliquant, plusieurs fois et de façon différente, une petite transformation (échange de tâches par exemple). Ce voisinage est ensuite évalué et comparé à la solution courante, etc. Nous présentons ci-dessous les méthodes amélioratrices les plus connues.

2.3.2.3.1 Méthodes de descente

Cette méthode se caractérise par le fait qu'elle ne considère, à chaque itération, que des solutions évoluant dans le sens de l'amélioration.

Cette méthode ne conduit pas, en général, au minimum absolue mais seulement à un minimum local qui constitue la meilleure des solutions accessibles compte tenu de la solution initiale. Pour améliorer l'efficacité, on peut évidemment l'appliquer plusieurs fois, avec des conditions initiales différentes et retenir comme solution finale le meilleur des minimum locaux obtenus ; cependant, cette procedure augmente sensiblement le temps de calcul de l'algorithme et ne garantie pas de trouver la configuration optimale. Une autre idée pour surmonter l'obstacle des minimums locaux consiste à autoriser, de temps en temps, des mouvement de remontée lors du changement de la configuration courante tel que adopté par la méthode du recuit simulé ou tabou.

La méthode d'échange de type r-optimale en est l'exemple type. Cette méthode d'optimisation a été initialement proposée par LIN [134] pour résoudre le problème de voyageur

de commerce, mais elle s'applique également à tout problème combinatoire dont la solution consiste en une permutation de r élément parmi n.

Le terme r-optimale indique qu'une solution ne peut plus être améliorée en échangeant au plus r éléments. La méthode consiste donc à sélectionner r éléments et à voir si, en les interchangeant, on obtient une meilleure solution. Nous remarquons qu'une solution n-optimale est une solution optimale (pour un problème de taille n). Ainsi, plus r augmente, plus on se rapproche de la solution optimale, mais plus les calculs sont difficiles. En pratique, on se limite à $r = 2$ ou 3.

2.3.2.3.2 Recuit simulé

Cette méthode est due aux physiciens KIRKPATRICK, GELLAT et VECCHI [121]. Elle s'inspire des méthodes de simulation de Métropolis en mécanique statistique définies durant les année 50.

L'analogie historique s'inspire du recuit des métaux en métallurgie : Un métal refroidi trop vite présente de nombreux défauts microscopiques, c'est équivalent d'un minimum local pour un problème d'optimisation combinatoire. Si on le refroidit lentement, les atomes se réarrangent, les défauts disparaissent et le métal a alors une structure très ordonnée, équivalent à un optimum global. Pour modifier l'état d'un matériau, le physicien dispose d'un paramètre de commande : La température. Le recuit est précisément une stratégie de contrôle de la température afin d'avoir un système physique dans un état de basse énergie.

L'algorithme s'appuie sur ces deux résultats de la physique statistique :

- D'une part, lorsque l'équilibre thermodynamique est atteint à une température donnée T, la probabilité, pour un système physique, de posséder une énergie donnée E, est $\exp^{(-E/K_bT)}$ où K_b est la constante de BOLTZMANN.
- D'autre part, pour simuler l'évolution d'un système physique vers son équilibre thermodynamique à une température donnée T, on peut utilisée l'algorithme de Métropolis : Partant d'une configuration donnée, on fait subir aux système une modification élémentaire ; si cette transformation a pour effet de diminuer la fonction objectif (ou énergie) du système, elle est acceptée ; si elle provoque au contrainte une augmentation de l'énergie $\triangle E$ de la fonction objectif, elle est acceptée tôt de même avec la probabilité $\exp^{(-\triangle E/T)}$ (en pratique, cette condition est réalisée en tirant au hasard un nombre

compris entre 0 et 1, et on accepte la configuration dégradant la fonction objectif de la qualité $\triangle E$ si le nombre tiré et inférieur à $\exp^{(-\triangle E/T)}$). On répète ce processus jusqu'à ce que le système soit figé : La température a atteint la valeur nulle, ou bien plus aucun mouvement de remontée du coût n'a été accepté.

Les inconvénient du recuit simulé résident, d'une part, dans les "réglages", comme la gestion de la décroissance par paliers de la température ; de bons réglages relèvent souvent du savoir faire de l'ingénieur. D'autre part, les temps de calcul peuvent devenir selon les cas très importants. Par contre, les méthodes de recuit simulé ont l'avantage d'être souples et rapidement implémentables lorsque l'on veut résoudre des problèmes d'optimisation combinatoire, le plus souvent de grande taille. Cette méthode a l'intérêt d'admettre une preuve de la convergence. Ainsi, lorsque certaines conditions sont vérifiées, on a la garantie d'obtenir la solution optimale.

2.3.2.3.3 Recherche tabou

Cette méthode, dont l'origine remonte à 1977 [79], a été formalisé en 1986 par GLOVER [80]. La principale particularité de la méthode tient dans la mise en œuvre de mécanismes inspirés de la mémoire humaine dont le but est de tirer les leçons du passé.

Le principe de base de la méthode tabou est simple. À partir d'une solution initiale quelconque, la méthode tabou engendre une succession de solutions ou configurations qui doit aboutir à une solution optimale. À chaque itération, le mécanisme de passage d'une configuration (s) à la suivante (t) est le suivant :

- On construit l'ensemble de voisins de s, c'est-à-dire l'ensemble des configurations accessibles en un seul mouvement élémentaire à partir de s, soit $V(s)$ l'ensemble de ces voisins.
- On évalue la fonction objectif f du problème pour chacune des configurations appartenant à $V(s)$. La configuration t, qui succède s dans la chaîne construite par tabou, est la configuration de $V(s)$ pour laquelle f prend la valeur minimale.

Cependant, telle quelle, la procédure ne fonctionne généralement pas, car il y a un risque important de retourner à une configuration déjà retenue lors d'une itération précédente, ce qui provoque un cyclage. Pour éviter ce phénomène, qui concerne plutôt des configurations peu éloignées (au sens du nombre de mouvements), on tient à jour, à chaque itération, une "liste tabou" de mouvements interdits ; cette liste circulaire contient m mouvements inverses

$(t \rightarrow s)$. La recherche du successeur de la configuration courante se restreint alors aux voisins de s qui peuvent être atteints sans utiliser de movement de la liste tabou. La procédure est stoppée dès qu'on a effectué un nombre donné d'itérations sans améliorer la meilleure solution courante.

Des versions plus élaborées, permettant des recherches plus efficaces, ont été proposées par la suite [81], [82]. Cette méthode donne de très bons résultats pratiques, malgré l'inexistence de résultats théoriques garantissant la convergence de l'algorithme vers la solution optimale.

2.3.2.3.4 Algorithmes génétiques

Les algorithmes génétiques sont des techniques de recherche inspirées de l'évolution biologique des espèces et basée sur une initiation des phénomènes d'adaptation des être vivants. L'application de ces algorithmes aux problèmes d'optimisation a été formalisé par GOLDBERG [83].

On part d'une population (ensemble de solutions) initiale sur laquelle des opérations de reproduction (de croisement ou de mutation) vont être réalisées dans l'objectif d'exploiter au mieux les caractéristiques et les propriétés de cette population. Un algorithme génétique consiste à faire évoluer progressivement, par générations successives, la composition de cette population, en maintenant sa taille constante : D'une génération à la suivante, la compétence de la population doit globalement s'améliorer. Les opérations de reproduction doivent mener à une amélioration de l'ensemble de la population puisque les bonnes solutions sont encouragées à échanger par croisement leurs caractéristiques et à engendrer des solutions encore meilleures. Toutefois, des solutions de très mauvaise qualité peuvent aussi apparaître et permettent d'éviter de tomber trop rapidement dans un optimum local. En effet, dans la phase de reproduction, les individus les plus compétents ont une probabilité de selection plus élevée. Cependant, les descendants n'éliminent pas leurs parents, qui demeurent dans la population courante.

La difficulté pour appliquer les algorithmes génétiques résident dans le besoin de coder les solutions et de fixer les valeurs des différents paramètres (taille de la population, coefficients de reproduction, probabilité de mutation, ...). De plus, ces algorithmes demandent un effort de calcul très important.

2.3.2.3.5 Optimisation par colonies de fourmis (OCF)

Cette métaheuristique a été introduite pour la première fois dans la thèse de doctorat de MARCO DORIGO [50] et a été inspirée par les études sur le comportement des fourmis réelles [86], [49], [48]. À l'origine, l'optimisation par colonie de fourmis a été conçu pour résoudre le problème du voyageur de commerce (VC). L'objectif de ce problème est de trouver la tournée la plus courte pour un ensemble de ville données. Une matrice fournissant les distances, $d_{i,j}$, entre toutes les paires de villes est utilisée pour estimer la longueur d'une tournée.

Dans l'optique d'un problème de VC, chaque fourmi représente un agent [53] et lorsqu'il se déplace de la ville i à la ville j, il laisse une trace sur le chemin (ij). De plus, un agent choisit la prochaine ville à visiter à l'aide d'une probabilité $P_{i,j}^{k}(t)$ basée sur un compromis entre l'intensité de la trace $\tau_{i,j}(t)$ et la visibilité $(1/d_{i,j})$ qui prend en considération la distance entre les villes. Les coefficients α et β sont des paramètres qui permettent de contrôler l'importance relative des deux éléments. Finalement, une fourmi k possède une forme de mémoire, tabou_k, lui rappelant la liste ordonnée des villes déjà visitée afin d'obliger celle-ci à former une solution admissible. Si le nombre total de fourmis est m et le nombre de villes à visiter est n, un cycle est réalisé lorsque chacune des m fourmis complètent une tournée des n villes.

(DORIGO et GAMBARDELLA) [52] ont proposé, en 1997, certaines améliorations à la version de base [50] concernant la règle de transition, la mise à jour de la trace qui se fait globalement mais aussi localement, l'intégration de méthode d'amélioration et l'utilisation d'une liste de candidats. Plusieurs autres travaux [54], [53], [51] procurent des explications supplémentaires sur le fonctionnement de l'algorithme, sur la notation et le choix des divers paramètres.

Dans (GAGNÉ & al.) [73], il a été proposé et validé certaines modifications à l'algorithme de base d'OCF pour la résolution d'un problème d'ordonnancement avec machine unique et temps de réglages dépendants de la séquence dans un objectif de minimisation du retard total. La matrice de distance $(d_{i,j})$ est remplacée par plusieurs matrices qui tiennent compte individuellement de certains aspects importants du contexte d'ordonnancement en regard de l'objectif à optimiser.

2.4 L'Ordonnancement Multicritère

Différents états de l'art de l'ordonnancement multicritère peuvent être rencontrés dans la littérature [103], [102]. L'analyse de ces travaux remarque :

- La nécessité de connaître les résultats du domaine de l'optimisation multi-objectif afin de mieux comprendre les difficultés issues de la prise en compte des critères conflictuels.
- Le besoin d'une typologie nous permet de formaliser les différents types de problèmes et d'unifier la notation de ces problèmes.
- le besoin de connaître les résultats sur les problèmes d'ordonnancement à critère unique.

2.4.1 Exemples d'application

Dans cette section, quelques problèmes d'ordonnancement correspondants à des situations pratiques sont présentés, quoi que ce soit leur domaine d'application.

Fabrication de bouteilles en verre

Une usine fabrique des bouteilles en verre dont les couleurs sont choisies initialement à la phase de planification [189]. Un four contenant le verre fondu d'une couleur donnée serve des machines de différentes formes. Ces machines sont munies de plusieurs moules permettant de former plusieurs types de bouteilles, et qui correspondent à des ordres différents. Le temps de changement d'un moule sur une machine est négligeable par rapport au temps de production, ainsi, le changement d'un produit à un autre se fait dans un temps dissimulé. La fabrication d'un produit par machine crée un profit mesurable. L'un des objectif est alors, en donnant la production à l'horizon, affecter les travaux aux machines afin de maximiser le profit total. D'un autre côté, changer les couleur dans le four influe sur l'ensemble des machines qu'il les serve, et ce changement peut être seulement apparu lorsque toutes les machines ont complété leurs production en cours. Afin de ne pas permettre aux machines d'être oisives pour un temps trop long, qui crée un profit prohibitif, on désire que les machine doivent arrêter la production dans un temps limité. Un second critère vise alors à affecter les travaux afin de minimiser la plus grande différence de charge de travail entre deux machines. Le décideur désire de trouver un meilleur compromis entre le profit total et le temps d'arrêt le plus long des machines.

Traitement de chèques

L'organisation d'un centre de traitement spécialisé dans le traitement des chèques (débit, crédit, impression, etc.) est similaire à un centre de production. Plus précisément, elle peut être présentée comme un problème flowshop hybride à trois étages, où les travaux passent plusieurs périodes à travers d'un même étage (recirculation). On peut associer deux dates échues de fin d'exécution. La première correspond au temps de traitement d'une opération du cheminement transférant les informations aux client. La deuxième correspond au temps de traitement de la dernière opération des travaux. Si cette date n'est pas respectée, la livraison des chèques sera retardée pendant une journée complète, qui crée un coût proportionnel à la somme d'argent retardée. Deux critères différents sont associés à ces dates échues de fin d'exécution. Le premier vise à minimiser le retard maximal afin de limiter le dérangement des clients, et le second critère correspond au nombre pondéré des travaux en retard où le poids associé est le coût encouru par le retard ([16]). Ces deux critères n'ont pas la même importance pour l'entreprise qui désire sur tout de minimiser le second et au même temps le premier critère.

Problème d'ordonnancement relatif au transport

Nombreux problèmes de planning et d'ordonnancement apparaissent lorsqu'on a affaire au transport des biens et/ou des voyageurs. Entre autre, le problème de l'équipage aérien peut être vu comme un problème multicritère ([136]). Ce problème survient lorsqu'ayant affaire à l'ordonnancement des équipes d'envols dans le transport aérien. Admettant qu'il y a un ensemble d'envols à effectuer et un ensemble de rotations prédéfinies, une rotation étant une séquence d'envols. Connaissant un ensemble de pilotes, on doit les affecter aux rotations sans violer les contraintes associées à la sécurité aérienne et à la compétence du pilote. Le but est de minimiser deux critères. Le premier est l'écart relatif moyen par pilote entre le temps d'envols mensuel réel et idéal. Le second critère est l'écart absolu moyen entre le nombre réel et idéal de pays étrangers visités, par prime durant le mois, et par pilote.

Une autre application de l'ordonnancement du transport est associé au service de planification du transport ferroviaire, dans les voies ferrées à grande vitesse ([30]). Ce problème apparaît lorsqu'ayant affaire au transport interurbain, où étant donnés plusieurs stations de trains et probablement un nombre énorme de voyageurs. Admettant un ensemble de stations, le but est de déterminer un ordonnancement-d'arrêt afin de satisfaire les contraintes du problème et de minimiser le critère. Un ordonnancement-d'arrêt est une séquence de stations où un train

doit s'arrêter. On doit aussi déterminer le nombre minimal de trains requis pour satisfaire l'ordonnancement-d'arrêt. Deux critères sont à minimiser : Le coût total de mise en œuvre pour la planification à l'horizon et à la perte du temps total de déplacement d'un voyageur.

2.4.2 Définition

On appelle un « ***problème d'ordonnancement multicritère*** », le problème qui consiste à déterminer un ordonnancement Pareto optimal pour plusieurs critères conflictuels. Il peut être subdivisé en trois sous-problèmes [188] :

1. ***Modélisation du problème***, dont la résolution mène à la détermination de la nature du problème d'ordonnancement sous considération aussi bien que la définition des critères à être pris en compte.
2. ***La prise en compte des critères***, dont la résolution mène à l'indication du contexte de résolution et la manière en laquelle on veut prendre en compte les critères. L'analyste détermine un module d'aide à la décision pour un problème multicritère, aussi appelé un module pour la prise en compte des critères.
3. ***Ordonnancement***, dont la résolution mène à trouver une solution pour le problème. L'analyste détermine un algorithme pour la résolution du problème d'ordonnancement, aussi appelé un module de résolution pour le problème d'ordonnancement.

2.4.3 Notation des problèmes d'ordonnancement multicritères

Dans la phase de *prise en compte des critères*, et suivant les données disponibles, l'analyste choisit une approche de résolution pour le problème d'ordonnancement et ainsi détermine un problème d'ordonnancement. En prenant en compte les différentes méthodes de détermination des optima de Pareto (citées dans le chapitre précédant), les fonctions à optimiser pour le problème d'ordonnancement peuvent prendre des différentes formes. Chacune traduit une méthode de détermination d'un optimum de Pareto. Les critères ne changent pas et ils correspondent à ceux définis durant la phase de *modélisation du problème.*

Après la phase de modélisation, un problème d'ordonnancement multicritère peut être noté dans une manière générale en utilisant la notation des trois champs, ou le champ γ comporte la liste des critères : $\alpha/\beta/Z_1, Z_2, \ldots, Z_K$; *i.e.* seulement le champ γ soit étendu et des nouvelles fonctions sont définies comme suit :

- Z si l'objectif est de minimiser l'unique critère Z (problème à un seul critère). Z peut être C_{max}, T_{max}, etc.

- $F_\ell(Z_1, \ldots, Z_K)$ si l'objectif est de minimiser une combinaison linéaire convexe des K critères. Par exemple, ce cas peut ce présenter si le décideur peut associer un poids à chaque critère.

- $\epsilon(Z_u/Z_1, \ldots, Z_{u-1}, Z_{u+1}, \ldots, Z_k)$ indique que seulement le critère Z_u est à minimiser, sujet à tous les autres critères bornés supérieurement par des valeurs connues. L'analyste est dans le cas de l'approche $\epsilon-$contrainte.

- $P(Z_1, \ldots, Z_K)$ indique une fonction non décroissante à minimiser, si on considère que tous les critères sont bornés supérieurement par des valeurs connues. L'analyste est dans le cas de l'analyse paramétrique.

- $F_T(Z_1, \ldots, Z_K)$ indique une fonction objectif étant une expression d'une distance à une solution idéale connue. La distance est calculée en utilisant la métrique de Tchebycheff. Cette solution idéale peut ne pas être atteinte.

- $F_{Tp}(Z_1, \ldots, Z_K)$ indique une fonction objectif étant une expression d'une distance à une solution idéale connue. La distance est calculée en utilisant la métrique pondérée de Tchebycheff. Cette solution idéale peut ne pas être atteinte.

- $F_{Tpa}(Z_1, \ldots, Z_K)$ indique une fonction objectif étant une expression d'une distance à une solution idéale connue. La distance est calculée en utilisant la métrique pondérée augmentée de Tchebycheff. Cette solution idéale peut ne pas être atteinte.

- $F_s(Z_1, \ldots, Z_K)$ indique une fonction très particulière qui prend en compte une solution idéale connue pour trouver la solution recherchée. L'analyste est dans le cas de l'approche du but à atteindre.

- $GP(Z_1, Z_2, \ldots, Z_K)$, si il y a des buts à atteindre pour chaque critère dans le problème d'ordonnancement (goal programming). Le problème n'est pas à optimiser les critères, mais à trouver une solution qui satisfait les buts, même si cette solution ne correspond pas à un optimum de Pareto.

- $Lex(Z_1, Z_2, \ldots, Z_K)$ indique que le décideur ne permet pas l'agrégation des critères. Les critères sont ordonnés selon leurs importances, le plus important est classé le premier. L'analyste utilise l'ordre lexicographique et optimise les critères l'un après l'autre.

- $\#(Z_1, Z_2, \ldots, Z_K)$ indique le problème d'énumération de tous les optima de Pareto. Par conséquent, on associe seulement à ce problème un algorithme de résolution à postériori qui n'utilise aucune méthode d'agrégation présentée dans le chapitre précédant.

Remarque 2.4.1. *Parmi ces approches, les F_ℓ, Lex, ϵ, GP et # approches sont les plus reconnues dans la littérature.*

2.4.4 Complexité

Pour un problème à K critères, il suffit que l'optimisation d'un seul critère soit $\mathcal{NP}$-difficile pour que le problème multicritère le soit aussi. La complexité des problèmes d'ordonnancement bicritères sur une seule machine a été considérée par [102], [34] (où CHEN et BULFIN ont posé des règles nous permettant de déduire la complexité des problèmes d'ordonnancement bicritères à partir des problèmes à un seul critère correspondants). La complexité des problèmes bicritères sur des machines multiples est adressée par [35].

CHAPITRE 3

Stratégies d'Approches de Résolution

Dans ce chapitre, nous présentons quelques stratégies d'approches de résolution des problèmes d'ordonnancement multicritère. Nous dressons un constat des différentes techniques utilisées dans ce cadre, telles que la programmation mathématique, les techniques de recherche arborescente et les métaheuristiques.

3.1 Méthodes exactes

Les problèmes bicritères ont reçu une attention particulière. Des méthodes exactes telles que le Branch-and-Bound et la programmation dynamique ont subi des révision afin de les adapter au cas multicritère. Cependant ces approches sont destinées à des problèmes de petites tailles, et leur efficacité est mise en question dès que la taille du problème augmente et le nombre de critères retenus croît. La procédure du Simplex a aussi reçu des modifications afin de répondre aux exigences de l'optimisation multi-objectif. On trouve dans [181] une étude assez détaillée des travaux effectués dans ce sens.

3.1.1 Programmation dynamique

L'application de la programmation dynamique dans le cadre de l'optimisation multicritère est rare. La difficulté est liée au principe de monotonie exigée par la méthode. Ce principe

réclame que la préférabilité d'une solution partielle reste préservée par recursion. Ce principe est difficilement vérifiable lorsque le nombre d'objectifs à optimiser est élevé (> 2).

3.1.2 Méthode de séparation et évaluation (Branch-and-Bound)

(RAMESH & al.) [159] proposent une variante multicritère de l'algorithme de Branch-and-Bound appliqué aux problèmes d'optimisation linéaire à variables entières. Le problème est résolu en combinant la procédure de Branch-and-Bound avec une spécification interactive des préférences du décideur.

3.2 Approches de résolution approchées

Les années soixante ont vu se multiplier les situations de conflit entre critères. Des situations où les mots compromis et équilibre représentent la clé de la réussite. Les techniques alors élaborées procèdent soit par transformation du problème en un problème mono-critère, soit en se basant sur la notion d'optimalité de Pareto ou finalement par traitement séparé des différents critères.

3.2.1 Transformation du problème vers le mono-critère

Cette approche [186] procède en transformant le problème du cas multicritère vers le mono-critère. Entre autres, les méthodes relatées dans le premier chapitre font l'objet de cette sous-section, à savoir les techniques d'agrégation, l'approche paramétrique, l'approche ϵ-contrainte, l'utilisation de la métrique de Tchebycheff et la programmation par but.

3.2.2 Approches par traitement séparé des objectifs

L'optimisation des différents critères est réalisée en traitant les différents objectifs séparément. Par exemple, dans les algorithmes génétiques, la prise en compte de différents critères apparaît au niveau de la phase de sélection (sélection parallèle, sélection lexicographique) ou de la phase de reproduction (reproduction multi-sexuelle).

3.3 Métaheuristiques

Cette section présente quelques techniques de l'approche Pareto (c'est-à-dire que le principe de dominance est utilisé) mais aucun essai n'est fait afin de présenter un état de l'art exhaustif sur ces techniques. En particulier, les algorithmes évolutionnaires et autres approches basées sur une recherche locale sont présentés. Pour une étude d'autres techniques classiques pour l'optimisation multi-objectif se référer à [181], alors que les états de l'art de compréhension sur les différentes approches pour l'optimisation multi-objectif en incluant les méthodes classiques et évolutionnaires peuvent être trouvés dans [39]. Après avoir décrire quelques métaheuristiques multi-objectifs les plus récentes, cette section présente quelques observations concernant les similarités et les différences entre ces approches en leurs applications à des problèmes d'optimisation multi-objectif.

3.3.1 Algorithmes évolutionnaires multi-objectifs

Il n'y a pas une définition universelle d'un algorithme évolutionnaire, mais dans un sens strict, un algorithme évolutionnaire manipule une population de solutions, développe cette population à l'aide de coopération (recombinaison, croisement) et de mutation. Cet algorithme utilise une représentation codée des solutions [99]. Un nombre d'algorithmes évolutionnaires multi-objectifs ont été proposés ces dernières années et le grand intérêt concernant ces méthodes a motivé l'extension des algorithmes évolutionnaires, proposés initialement pour des problèmes d'optimisation mono-objectifs, à des variantes multi-objectifs. Quelques uns de ces algorithmes évolutionnaires multi-objectifs sont décrits brièvement par la suite.

3.3.1.1 Vector Evaluated Genetic Algorithm (VEGA) [168]

Ça peut être le premier algorithme génétique dans lequel le concept de dominance a été implanté pour l'évaluation et la sélection des individus. Cet algorithme considère une population de N individus. Ces individus sont répartis en k sous-populations, chaque valeur de k représentant un objectif à optimiser. À chaque génération, un nombre de sous-populations est généré par selection en fonction de l'objectif k. Ensuite, ces sous-populations sont regroupées pour former une nouvelle population de N individus et les opérateurs de croisement et de mutation sont appliqués.

3.3.1.2 Multi-objectif Genetic Algorithm (MOGA) [66]

En 1993, (FONSECA & FLEMING) ont proposé un procédé de *ranking* basé sur la notion d'optimalité de Pareto. Dans la *Non Dominated Sorting* (NDS), le rang d'un individu est égal au nombre de solutions qui le domine dans la population plus un. Le rang de l'individu jouera alors le rôle de la fonction d'utilité globale. Les étapes suivantes décrient le principe d'évaluation dans cette approche.

1. Pour chaque individu i de la population Pop, calculer $r(i) = 1 + |\{j/\ j \in Pop$ et j domine $i\}|$;
2. Tirer la population sur la base des rangs r ;
3. Affecter à chaque individu i un niveau d'adéquation $f(i)$ par interpolation depuis les meilleurs ($r(i) = 1$) jusqu'aux plus mauvais ($r(i) \leq Taille - Pop$).

3.3.1.3 Niched Pareto Genetic Algorithm (NPGA) [105]

(HORN & al.) ont proposé le NPGA, un algorithme utilisant une selection par tournoi, basée principalement sur la dominance de Pareto. Le NPGA exécute les mêmes étapes que l'AG standard, la seule chose qui les différencie étant la méthode de selection. À chaque tournoi, deux individus candidats, A et B, sont sélectionnés aléatoirement dans la population initiale. Au lieu de limiter la comparaison aux deux individus (comme c'est le cas pour l'AG standard), un ensemble d'individus (ou ensemble de comparaison) est également sélectionné aléatoirement dans la population. Les deux candidats sélectionnés sont comparés à chaque individu de l'ensemble de comparaison. Si l'un des deux candidats est dominé par l'ensemble, et le second ne l'est pas, ce dernier est sélectionné pour la reproduction. Si les deux candidats sont dominés ou non dominés par l'ensemble de comparaison, alors il faut utiliser la technique basée sur la fonction de partage (*fitness sharing*) pour choisir le candidat gagnant. Le résultat du tournoi est décidé lors du calcul du compteur de niche. La taille de l'ensemble de comparaison permet de contrôler la pression de selection ou de dominance. Une version améliorée de cet algorithme, appelée *NPGA-2*, est décrite par (ERICKSON & al.) [61].

3.3.1.4 Non Dominated Sorting Genetic Algorithm (NDSGA) [180]

Cet algorithme classe aussi les individus par rapport à un procédé de ranking basé sur la notion d'optimalité de Pareto comme le cas de [66]. (SRINIVAS & DEB) [180] ont proposé une

nouvelle version de l'algorithme NDSGA, appelé *NDSGA-II*, qui est considéré comme étant plus efficace que son prédécesseur car :

- Il utilise une approche élitiste qui permet de sauvegarder les meilleures solutions trouvées lors des générations précédentes.
- Il utilise une procédure de tri basée sur la non-dominance, plus rapide.
- Il ne nécessite aucun réglage de paramètres.
- Il utilise un opérateur de comparaison basé sur un calcul de la distance de crowding.

3.3.1.5 Strength Pareto Evolutionary Algorithm (SPEA) [211]

La méthode dite *Strength Pareto Evolutionary Algorithm* est basée sur la combinaison d'anciennes et de nouvelles techniques afin de générer plusieurs solutions Pareto optimales en parallèle. Comme pour les autres techniques :

- Une population additionnelle est maintenue, elle archive les solutions Pareto trouvées toute au long de la recherche.
- Le concept de dominance est utilisé afin d'associer une valeur d'adéquation scalaire aux individus.
- Un clustering des solutions archivées est réalisé afin de réduire leur nombre sans perdre la forme de la frontière Pareto.

D'autre part, la technique se distingue par :

- La combinaison de trois méthodes en un seul algorithme.
- L'adéquation d'un individu est calculée à partir des solutions archivées seulement.
- Les solutions archivées participent aussi à la phase de selection.
- Une nouvelle technique de maintien de niches est proposée, ne requérant aucune valeur de paramètre et se basant sur la notion d'optimlité de Pareto.

La version améliorée de cette technique, appelée *SPEA-2*, introduite par [210].

3.3.1.6 Autres algorithmes évolutionnaires multi-objectifs

Les algorithmes cités ci-dessus ne représentent qu'un échantillon d'un tas de méthodes proposées dans la littérature ces dernières années. Autres approches incluent le multi-objectif Messy Genetic Algorithm (*MOMGA*) I et II [194] et le Pareto Covering Genetic Algorithm

(*PCGA*) [125]. Un autre algorithme génétique multi-objectif a été proposé par [146] particulièrement appliqué aux problèmes d'ordonnancement sur machines. Dans ces dernières années, plusieurs autres extensions des algorithmes évolutionnaires pour l'optimisation multi-objectif ont été proposés. À titre d'exemples, les Micro-Genetic algorithms, les Cellular Genetic Algorithms, les méthodes Particle swarm optimisation, agent-based algorithms et autres algorithmes pouvant être rencontrés dans les conférences (*Evolutionary Multi-Criterion Optimisation "EMO", 2001*), (*Congress on Evolutionary Computation "CEC", 2002*). Pour une étude détaillée sur les principes de l'optimisation évolutionnaire multi-objectif, on se réfère à [194], [46].

3.3.2 Métaheuristiques alternatives multi-objectifs

Une autre classe des métaheuristiques pour l'optimisation de Pareto sont celles utilisant explicitement une recherche ou une exploration du voisinage (au lieu des opérateurs génétiques) afin de guider la recherche ou comme étant un outil important du processus (approches hybrides). Identiquement à la section précédente, plusieurs métaheuristiques multi-objectifs utilisant une recherche locale ont été mises au point dans la littérature. Quelques unes de ces métaheuristiques multi-objectifs sont décrites brièvement ci-dessous :

3.3.2.1 Simulated Annealing for Multi-objective Optimization [174]

Ça peut être la première extension du recuit simulé pour l'optimisation multi-objectif reportée dans la littérature. L'idée proposée a été la modification des critères d'acceptance des solutions candidates dans l'algorithme original. Des différents critères alternatifs ont été examinés afin de croître la probabilité d'acceptation des solutions non dominées. Une règle spéciale donnée par la combinaison de plusieurs critères a été proposée afin de concentrer la recherche presque exclusivement sur les solution non dominées.

3.3.2.2 Multi-objective Tabu Search (MOTS) [97]

Cet algorithme est une extension à base de population de la métaheuristique de recherche locale utilisant un ensemble de poids afin de guider la recherche vers la frontière de Pareto. Chaque solution maintient sa propre liste tabou et les poids sont ajustés afin de garder les solutions en dehors de leurs voisines et par conséquent essayer de couvrir entièrement la surface de compromis.

3.3.2.3 Pareto Simulated Annealing (PSA) [43]

C'est une extension à base de population du recuit simulé proposé à des problèmes d'optimisation combinatoires multi-objectifs. La population de solutions explorent leur voisinage identiquement au recuit simulé classique, mais des poids sont attribués à chaque objectif et à chaque itération afin d'assurer une tendance pour couvrir la surface de compromis. Les poids sont ajustés à chaque solution de manière à croître la probabilité de les garder en dehors de leurs voisines comme étant le cas dans l'algorithme de recherche tabou [97]. À partir du recuit simulé, cette métaheuristique hybride prend l'idée de la recherche voisine, probabilité d'acceptation des solutions candidates et la dépendance de cette acceptation d'un paramètre température. À partir des algorithmes génétiques, l'approche prend l'idée de l'utilisation d'un échantillon de population de solutions interactantes.

3.3.2.4 Multi-objective Simulated Annealing (MOSA) [193]

Cette approche est une autre extension du recuit simulé dans lequel une fonction d'agrégation est utilisée pour évaluer la fitness des solutions afin d'essayer d'approcher les différentes régions de la surface de compromis. L'algorithme se déroule avec une seule solution courante mais maintient une population de solutions non dominées trouvées durant la recherche.

3.3.2.5 Evolutionary Local Search Algorithm (ELSA) [140]

Ceci est un algorithme évolutionnaire utilisant une sélection locale comme étant l'outil principal dans le but de minimiser l'interaction entre les individus de la population. L'idée de cette approche est qu'une population d'individus compétents peuvent chercher l'espace d'une manière parallèle. Cet algorithme n'utilise pas le croisement mais le seul opérateur pour générer des nouvelles solutions est la mutation. Les auteurs soulignent que l'efficacité majeure de cet algorithme est sa potentialité d'être implanté en parallèle et qu'il maintient la diversification de la population d'une manière identique que la fonction de partage (fitness sharing) mais plus efficacement.

3.3.2.6 Genetic Local Search (GLS) [112]

Cet algorithme est proposé pour la génération approximative d'un ensemble de solutions non dominées dans l'optimisation combinatoire multi-objectif. Cet algorithme est une hybridation entre les algorithmes génétiques et la recherche locale dont une fonction d'agrégation

pondérée est générée aléatoirement à chaque itération. Cette fonction est utilisée pour sélectionner les solutions qui seront croisées afin de former des solutions actuelles et pour guider l'optimisation locale de celles-ci. Les premières implantations de cet algorithme ont été faites pour des problèmes d'ordonnancement multicritère [107].

3.3.2.7 Simulated Annealing for Multi-objective Optimisation (SAMO) [183]

C'est une autre extension du recuit simulé dans lequel une température est associée à chaque objectif du problème. L'algorithme utilise seulement une solution et le processus recuit ajuste chaque température indépendamment par rapport à la performance de la solution en chaque critère durant la recherche. Un archive est utilisé pour sauvegarder toutes les solutions non dominées visitées.

3.3.2.8 Autres Métaheuristiques multi-objectifs utilisant une recherche locale

Plusieurs autres approches issues de cette classe ont été proposées et examinées dans la littérature. Par exemple, la recherche tabou variante de [13] maintient une solution unique mais des listes additionnelles de solutions non dominées trouvées durant la recherche sont garder afin de guider la recherche. Autres variantes multi-objectifs de l'optimisation par colonies de fourmis hybrident entre la recherche tabou et les algorithmes évolutionnaires et autres implantations de la recherche locale génétique multi-objectif peuvent être rencontrées dans les conférences (*Evolutionary Multi-Criterion Optimisation "EMO", 2001*), (*Congress on Evolutionary Computation "CEC", 2002*).

3.3.3 Quelques observations sur les métaheuristiques multi-objectifs

Presque tous les algorithmes évolutionnaires multi-objectifs utilisent la notion d'optimalité de Pareto pour la sélection des individus pendant la recherche. Cependant, plusieurs métaheuristiques alternatives multi-objectifs (celles utilisant une recherche locale) emploient une méthode différente, dans la plus part des cas, une fonction d'agrégation pondérée. Il est discuté dans [111] que les fonctions d'agrégation ont plus d'avantages car elles forcent les solutions à explorer certaines régions de l'espace des solutions et ces fonctions ne sont pas nécessairement interprétées comme étant une transformation des objectifs multiples en un seul objectif, *i.e.* elles sont juste un outil pour guider la recherche dans l'optimisation multi-objectif et ça ne veut pas dire qu'elles représentent les préférences du décideur. À son avis, la

dominance assure seulement la convergence mais pas la dispersion tout au long du front. Cependant, les fonctions d'agrégation sont capables de produire des points sur toutes les régions du front. Dans [112], l'auteur suggère que la dominance de Pareto n'est pas convenue à la combiner avec une recherche locale.

Généralement, les algorithmes évolutionnaires multi-objectifs utilisent un opérateur génétique standard et les différences entre ces algorithmes résident dans les stratégies utilisées pour la sélection et la diversification. Les approches alternatives emploient une recherche voisinage, qui a besoin d'être conçue par rapport au problème. Plusieurs algorithmes évolutionnaires multi-objectifs utilisent les mécanismes de diversification basés sur un clustering ou une fonction de partage (fitness sharing) pour disperser la population tout au long de la surface de compromis. En revanche, les métaheuristiques alternatives multi-objectifs emploient la variation des poids pour ce but.

CHAPITRE 4

Un État de l'Art sur Quelques Types de Problèmes d'Ordonnancement Multicritère

Dans ce chapitre, nous décrivons quelques plus importants travaux effectués dans le domaine de l'ordonnancement multicritère à une seule machine, sur machines parallèles et problèmes d'ateliers.

4.1 Problèmes d'ordonnancement à une seule machine

La littérature sur l'ordonnancement multicritère vise principalement les problèmes d'ordonnancement à une seule machine. Ils sont en quelques sorte intéressants car leurs propriétés peuvent être généralisées pour plusieurs situations compliquées ou utilisées afin de décrire quelques algorithmes heuristiques pour des problèmes multi-machines.

un résultat général est apporté par HOOGEVEEN [102] montrant que le problème $1||\epsilon(f^1_{\max}/f^2_{\max}, \ldots, f^K_{\max})$ est résolu en $O(n^2)$ si $k = 2$ et en $O(n^{k(k+1)-6})$ si $k \geqslant 3$, lorsque $f^i_{\max}(S) = \max\limits_{1\leq j\leq n}(f^i_j(C_j(S)))$ avec f^i_j des fonctions croissantes en dates de fin d'exécution C_j. L'auteur montre que la cardinalité de l'ensemble E est au plus $(\frac{n(n-1)}{2}+1)^{K-1}$.

4.1.1 Minimiser la somme des dates de fin d'exécution des tâches

Dans cette sous section, nous présentons les problèmes en lesquels la moyenne des dates de fin d'exécution des tâches est considérée comme étant l'un des critères.

• Le problème $1||\epsilon(\overline{C}/L_{\max})$ avec $L_{\max} = 0$ est résolu par SMITH [177]. L'algorithme de SMITH [177] est étendu au problème $1||\epsilon(\overline{C}/L_{\max})$ par (HECK & ROBERTS) [98], qui proposent un algorithme à priori. L'algorithme 1 à posteriori est proposé par (VAN WASSENHOVE & GELDERS) [202]. Cet algorithme représente une étape majeure en ordonnancement multicritère et il aboutit à plusieurs algorithmes similaires, exacts ou heuristiques. Son principe est comme suit : pour une valeur fixée ϵ, un optimum de Pareto strict est déterminé en utilisant un algorithme glouton combinant les deux règles SPT et EDD. La contrainte $L_{max} \leq \epsilon$ est équivalente à imposer des dates limites aux tâches, et tel que, l'algorithme se déroulant à l'envers, commençant par la dernière position. À chaque position, une liste des tâches choisies est déterminée et parmi eux la règle de priorité SPT/EDD est appliquée pour sélectionner la tâche à ordonnancer. La nouvelle valeur de ϵ est déduite à partir de l'ordonnancement construit.

Algorithme 1 : L'algorithme de (VAN WASSENHOVE & GELDERS, 1980)

Données : T étant l'ensemble des tâches à ordonnancer ;

Supposer que $p_1 \leq p_2 \leq \cdots \leq p_n$;

début

$\underline{\text{Étape 1}}$: ***/* Initialisation de l'algorithme */***

$\epsilon := \sum_{i=1}^{n} p_i$;

/* Initialisation des dates limites */

$\widetilde{d}_i := d_i + \epsilon, \forall i = 1, \ldots, n$;

$End := Faux$; $E := \emptyset$;

$\underline{\text{Étape 2}}$: ***/* Détermination de l'ensemble E */***

tant que $(End = Faux)$ **faire**

$L := T$; $S := \emptyset$

/* Utiliser la version modifiée de la règle SPT */

tant que $((End = Faux)$ ***et*** $(L \neq \emptyset))$ **faire**

$F := \{J_i \in L / \widetilde{d}_i \geqslant \sum_{J_k \in L} p_k\}$;

si $(F = \emptyset)$ **alors**

$End := Vrai$;

sinon

Soit $J_i \in F$ tel que $p_i = \max_{J_k \in F}(p_k)$;

/* Choisir la tâche possédant la plus grande date échue */

$S := \{J_i\} // S$;

$L := L - \{J_i\}$;

si $L = \emptyset$ **alors**

$E := E + \{S\}$;

$\epsilon := -1$;

$\widetilde{d}_i := d_i + \epsilon, \forall i = 1, \ldots, n$;

$End := Faux$;

fin

Résultat : E étant l'ensemble des solutions efficaces déterminées par cet algorithme.

Exemple illustratif

Nous considérons un problème avec $n = 5$ tâches :

J_i	J_1	J_2	J_3	J_4	J_5
p_i	3	5	6	7	9
d_i	23	22	24	22	18

☞ $\epsilon = 30$, $\widetilde{d}_i = [53; 52; 54; 52; 48]^T$, $End = Faux$ et $E = \emptyset$.

☞ $L = \{J_1, J_2, J_3, J_4, J_5\}$, $S_1 = \emptyset$.

$F = \{J_1, J_2, J_3, J_4, J_5\}$, $i = 5$, $S_1 = (J_5)$.

$F = \{J_1, J_2, J_3, J_4\}$, $i = 4$, $S_1 = (J_4, J_5)$.

$F = \{J_1, J_2, J_3\}$, $i = 3$, $S_1 = (J_3, J_4, J_5)$.

$F = \{J_1, J_2\}$, $i = 2$, $S_1 = (J_2, J_3, J_4, J_5)$.

$F = \{J_1\}$, $i = 1$, $S_1 = (J_1, J_2, J_3, J_4, J_5)$, $End = Vrai$, $L_{\max}(S_1) = 12$ et $\overline{C}(S_1) = 76$.

$E = \{(J_1, J_2, J_3, J_4, J_5)\}$.

$End = Faux$, $\epsilon = 11$, $\widetilde{d}_i = [34; 33; 35; 33; 29]^T$

☞ $L = \{J_1, J_2, J_3, J_4, J_5\}$, $S_2 = \emptyset$.

$F = \{J_1, J_2, J_3, J_4\}$, $i = 4$, $S_2 = (J_4)$.

$F = \{J_1, J_2, J_3, J_5\}$, $i = 5$, $S_2 = (J_5, J_4)$.

$F = \{J_1, J_2, J_3\}$, $i = 3$, $S_2 = (J_3, J_5, J_4)$.

$F = \{J_1, J_2\}$, $i = 2$, $S_2 = (J_2, J_3, J_5, J_4)$.

$F = \{J_1\}$, $i = 1$, $S_2 = (J_1, J_2, J_3, J_5, J_4)$, $End = Vrai$, $L_{\max}(S_2) = 8$ et $\overline{C}(S_2) = 78$.

$E = \{(J_1, J_2, J_3, J_4, J_5), (J_1, J_2, J_3, J_5, J_4)\}$.

$End = Faux$, $\epsilon = 7$, $\widetilde{d}_i = [30; 29; 31; 29; 25]^T$

☞ $L = \{J_1, J_2, J_3, J_4, J_5\}$, $S_3 = \emptyset$.

$F = \{J_1, J_3\}$, $i = 3$, $S_3 = (J_3)$.

$F = \{J_1, J_2, J_4, J_5\}$, $i = 5$, $S_3 = (J_5, J_3)$.

$F = \{J_1, J_2, J_4\}$, $i = 4$, $S_3 = (J_4, J_5, J_3)$.

$F = \{J_1, J_2\}$, $i = 2$, $S_3 = (J_2, J_4, J_5, J_3)$.

$F = \{J_1\}$, $i = 1$, $S_3 = (J_1, J_2, J_4, J_5, J_3)$, $End = Vrai$, $L_{\max}(S_3) = 6$ et $\overline{C}(S_3) = 80$.

$E = \{(J_1, J_2, J_3, J_4, J_5), (J_1, J_2, J_3, J_5, J_4), (J_1, J_2, J_4, J_5, J_3)\}$.

$End = Faux$, $\epsilon = 5$, $\widetilde{d}_i = [28; 27; 29; 29; 23]^T$

☞ $L = \{J_1, J_2, J_3, J_4, J_5\}$, $S_4 = \emptyset$.

$End = Vrai$, $L \neq \emptyset$.

Remarque 4.1.1. *Des résultats numériques montrent que pour un problème avec* $n = 50$ *tâches, ils existent plus de* 29 *optima de Pareto stricts.*

(NELSON & al.) [149] étudient aussi ce problème d'énumération pour lequel ils proposent un algorithme branch-and-bound qui détermine un sous ensemble de FE. Des conditions de dominance sont utilisées pour améliorer l'efficacité de cet algorithme. (ESSWEN & al.) [63] propose aussi une version modifiée de l'algorithme.

• Le problème $1||F_\ell(T_{\max}, \overline{C})$ appartient à la classe $\mathcal{P}$ et est étudié par (SEN & GUPTA) [172]. Ils proposent un algorithme branch-and-bound qui énumère tous les optima supportés de Pareto faibles.

• EMMONS [58] étudie le problème $1||Lex(f_{\max}, \overline{C})$ qui appartient à la classe $\mathcal{P}$. Pour sa résolution, EMMONS propose un algorithme glouton basé sur celui de LAWLER [127] destiné pour le problème $1|prec|f_{\max}$.

• JOHN [113] étudie le problème $1||\epsilon(\overline{C}/f_{\max})$ pour lequel il propose un algorithme en $O(n^2|E|)$ déterminant l'ensemble E. L'auteur montre que $|E| < \frac{1}{4}(n^2 - 1)(p_{\max} - p_{\min})$ où $p_{\max} = \max_{i=1,\ldots,n}(p_i)$ et $p_{\min} = \min_{i=1,\ldots,n}(p_i)$. (HOOGEVEEN & VAN DE VELDE) [104] s'intéressent aussi à ce problème. Ils proposent un algorithme basé sur une méthode gloutonne pour déterminer un optimum de Pareto strict lorsque ϵ est fixé et ils décrivent en suite un algorithme à posteriori qui détermine l'ensemble E en variant ϵ. Les auteurs montrent que $|E| < \frac{n(n-1)}{2} + 1$. Cette borne supérieure est clairement plus précise que celle proposée par JOHN [113].

• Le problème $1|nmit|\epsilon(\overline{C}/E_{\max})$ pour lequel le temps arrêt machine n'est pas autorisé est fortement $\mathcal{NP}$-difficile. (AZIZOGLU & al.) [5] proposent une heuristique qui approxime un sous ensemble de FE.

• Le problème $1|nmit|F_\ell(\overline{C}/L_{\max} - L_{\min})$ pour lequel la complexité est inconnue est abordé par (SEN & al.) [173]. Ils proposent un algorithme branch-and-bound qui énumère l'ensemble des optima supportés de Pareto stricts.

• Le problème $1||Lex(\overline{U}, \overline{C})$ est $\mathcal{NP}$-difficile. Pour sa résolution, EMMONS [59] propose une heuristique. L'énumération des optima de Pareto des deux critères $\overline{C}$ et $\overline{U}$ est étudiée par (NELSON & al.) [149] qui proposent un algorithme à posteriori pour le problème $1||\epsilon(\overline{C}/\overline{U})$. C'est un algorithme branch-and-bound qui détermine un sous ensemble des optima de Pareto faibles. Il consiste à déterminer itérativement une valeur ϵ et en suite minimiser le critère $\overline{C}$ sous la contrainte $\overline{U} \leq \epsilon$. Les auteurs proposent également deux heuristiques basées sur l'algorithme optimal précédant. Ce problème est réabordé par (KIRAN & UNAL) [120] qui proposent des conditions générales pour le calcul de ces optima. Ces conditions sont notamment reliées aux propriétés de la séquence obtenue en appliquant la règle SPT et l'algorithme de SMITH [177] destiné pour le problème $1||\epsilon(\overline{C}/L_{\max})$.

• Le problème $1||\#(\overline{T}, \overline{C})$ est abordé par LIN [133] qui propose un algorithme à posteriori. Ce problème est $\mathcal{NP}$-difficile tant que le problème $1||\overline{T}$ est pareil. LIN propose des conditions de dominance entre les tâches, qui sont valides pour tous les ordonnancements de Pareto stricts. Pour la détermination de l'ensemble E, un algorithme de programmation dynamique, intégrant ces conditions de dominance, est proposé. Il est basé sur l'algorithme de programmation dynamique multicritère introduit par YU [208] et (YU & SEIFORD) [209].

• (FRY & LEONG) [71] étudient le problème $1||F_\ell(\overline{E}, \overline{C})$ dont la complexité est inconnue et pour lequel ils proposent un programme en nombres entiers. Des résultats expérimentaux montrent que la solution optimale peut être déterminée en moins de 100 secondes pour des problèmes à 10 tâches.

• (NELSON & al.) [149] étudient le problème $1||\epsilon(\overline{C}/\overline{U}, T_{\max})$ pour lequel ils proposent un algorithme branch-and-bound déterminant un sous ensemble de FE. Pour augmenter l'efficacité de cet algorithme et réduire l'espace de recherche, NELSON, SARIN et DANIELS utilisent des conditions de dominance.

4.1.2 Minimiser la somme des dates de fin d'exécution pondérées des tâches

• (CHEN & BULFIN) [33] étudient quelques problèmes d'ordonnancement bicritères avec des durées opératoires unitaires. Parfois, ce type d'étude permet de trouver des résultats sur la complexité de quelques problèmes. Il est possible de trouver une application pour tels problèmes, et d'après (CHEN & BULFIN), dans les ateliers de fabrication de voiture comme étant le cas de Toyota (voir [144]). (CHEN & BULFIN) s'intéressent aux problèmes $1|p_i = 1|Lex(\overline{C}_w, Z_2)$ avec $Z_2 \in \{\overline{T},\ \overline{T}_w,\ \overline{U},\ \overline{U}_w,\ T_{\max}\}$, pour lesquels ils proposent des algorithmes gloutons, en $O(nlog(n))$, basés sur la règle WSPT. En outre, (CHEN & BULFIN) étudient le problème $1|p_i = 1|\epsilon(\overline{C}_w/T_{\max})$ pour lequel ils proposent un algorithme qui énumère l'ensemble E. Les problèmes $1|p_i = 1|F_\ell(\overline{C}_w, Z_2)$ avec $Z_2 \in \{\overline{T},\ \overline{T}_w,\ \overline{U},\ \overline{U}_w\}$, sont également abordés. (CHEN & BULFIN) notent que l'algorithme de (ANEJA & NAIR) [3] résolvant le problème d'affectation bicritère, peut être aisément modifié pour la résolution de ces problèmes. Ainsi, ils proposent de déterminer l'ensemble E tandis que l'approche F_ℓ permet seulement de calculer les solutions de l'ensemble E_s dans le cas où l'ensemble des solutions n'est pas convexe. (CHEN & BULFIN) abordent aussi les problèmes $1|p_i = 1|Lex(Z_1, \overline{C}_w)$ avec $Z_1 \in \{\overline{T}_w,\ \overline{U},\ \overline{U}_w,\ T_{\max}\}$ et montrent comment réduire ces problèmes à un problème d'affectation résolu en $O(n^3)$. Pour la résolution du problème $1|p_i = 1|Lex(\overline{T}, \overline{C}_w)$, les auteurs proposent un algorithme glouton en $O(nlog(n))$.

• (CHAND & SCHNEEBERGER) [28] proposent un algorithme de programmation dynamique pour résoudre le problème $1||Lex(L_{\max}, \overline{C}_w)$. Ça complexité est également étudiée par HOOGEVEEN [102] qui montrent que ce problème est $\mathcal{NP}$-difficile.

• Le problème $1||\epsilon(\overline{C}_w/L_{\max})$ avec $L_{\max} = 0$ est fortement $\mathcal{NP}$-difficile. SMITH [177] propose un algorithme basé sur les règles EDD et WSPT, qu'il conjecture d'être optimal. EMMONS [58] montre par un contre exemple que cette conjecture est fausse. Ce problème est repris par BANSAL [10] qui propose un algorithme branch-and-bound pour la détermination de la solution optimale, si elle existe.

(CHAND & SCHNEEBERGER) [29] montrent que pour un certain nombre de problèmes et si une solution existe avec $L_{\max} = 0$, l'algorithme proposé par SMITH est optimal.

Notamment, c'est le cas des problèmes où les poids w_i sont définis par $w_i = f(p_i)$, où f est une fonction décroissante.

(HECK & ROBERTS) [98] étudient le problème général de minimisation du critère $\overline{C}_w$ sous la contrainte $L_{\max} \leq 0$. Ce problème est fortement $\mathcal{NP}$-difficile car : (i) lorsque ϵ est fixé, il se réduit au problème $1|\widetilde{d}_i|\overline{C}_w$, et (ii) le problème $1|\widetilde{d}_i|\overline{C}_w$ est fortement $\mathcal{NP}$-difficile ([129]). Pour sa résolution, ils proposent une heuristique basée sur un résultat présenté dans [177]. BURNS [26] proposent aussi une heuristique basée sur la méthode du voisinage. Par la suite, MIYAZAKI [142] remit ce problème en question et montre qu'il existe une réduction polynomiale du problème $1|L_{\max} \leq \epsilon|-$ vers le problème $1|\widetilde{d}_i|-$ (ou $1|d'_i, L_{\max} = 0|-$) et ça revient au problème étudié par [177]. MIYAZAKI propose alors des conditions pour améliorer une séquence en permutant des tâches adjacentes. En utilisant l'heuristique de [177] pour l'obtention d'une séquence initiale, une heuristique de voisinage en $O(n^3)$ est mise en œuvre. Des résultats expérimentaux montrent son efficacité par rapport à celles proposées par [177] et [26]. Aucune comparaison n'est faite avec celle de [98] ou l'algorithme branch-and-bound de [10].

• (VAN WASSENHOVE & GELDERS) [201] s'intéressent à la minimisation des deux critères $\overline{C}_w$ et $\overline{T}_w$ via le problème $1||F_\ell(\overline{C}_w, \overline{T}_w)$ qui est fortement $\mathcal{NP}$-difficile tant que le problème $1||\overline{T}_w$ est ainsi. Ils proposent des conditions de dominance basées sur le résultat démontré par (LAWLER & al.) [128] et quatre algorithmes de résolution. Les trois premiers sont des algorithmes branch-and-bound qui se diffèrent seulement par la borne inférieure utilisée en chaque nœud. Dans le premier algorithme, une borne inférieure d'un nœud est calculée en résolvant un problème de transport appliqué aux tâches non ordonnancées de ce nœud. Les deux auteurs montrent comment construire la matrice des coûts de transport. Dans le second algorithme branch-and-bound proposé, la borne inférieure de chaque nœud est calculée en résolvant un problème d'affectation appliqué aux tâches non ordonnancées du nœud courant. Des conditions de dominance montrées par ces deux auteurs sont aussi utilisées. Une méthode proposée par FISHER [65] pour le calcul des bornes inférieures est utilisée dans le troisième algorithme. Ces bornes sont calculées par un algorithme de programmation dynamique, qui résout le problème initial et utilise la condition de dominance présentée. Des résultats expérimentaux montrent que cet algorithme est plus rapide mais exige beaucoup plus de l'espace mémoire que les algorithmes de branch-and-bound. Ils conjecturent que pour des problèmes de plus de 40 tâches, l'algorithme combinant celui de branch-and-bound et celui

de programmation dynamique est plus efficace que les autres.

4.1.3 Minimiser les coûts des durées de dépréciation pondérées de toutes les tâches

VICKSON [196] étudie le problème $1|p_i \in [\underline{p}_i;\overline{p}_i]|F_\ell(\overline{C}_w, \overline{CC}_w)$. VICKSON ne présente pas une étude de sa complexité mais conjecture qu'il est $\mathcal{NP}$-difficile, contrairement aux problèmes $1|p_i \in [\underline{p}_i;\overline{p}_i]|F_\ell(\overline{C}, \overline{CC}_w)$ et $1|p_i \in [\underline{p}_i;\overline{p}_i]|F_\ell(\overline{T}_{\max}, \overline{CC}_w)$, qui appartiennent à la classe $\mathcal{P}$ [197]. Le critère $\overline{CC}_w$ est le coût total du temps dépréciation, défini par :

$$\overline{CC}_w = \sum_{i=1}^{n}\sum_{j=1}^{m} w_{i,j}x_{i,j},$$

avec $w_{i,j}$ le coût d'une unité du temps de compression de la tâche J_i sur la machine M_j. $x_{i,j}$ est le nombre d'unités du temps de compression de la tâche J_i sur la machine M_j, ce qui donne $p_{i,j} = \overline{p}_{i,j} - x_{i,j}$.

L'auteur propose un algorithme branch-and-bound et une heuristique gloutonne basée sur la règle WSPT. Un cas d'étude montre que l'heuristique est capable d'ordonnancer et calculer le temps de traitement optimal de certaines tâches. Pour les autres tâches, l'algorithme de branch-and-bound doit être utilisé. Le problème $1|p_i \in [\underline{p}_i;\overline{p}_i]|\epsilon(\overline{CC}_w/T_{\max})$ est résolu par (VAN WASSENHOVE & BACKER) [200]. Les deux auteurs montrent que leurs algorithme polynomial calcule seulement l'ensemble des optima supportés de Pareto stricts. Cet algorithme est alors étendu pour la résolution des problèmes avec une fonction de coût maximum générale $f_{\max}$ au lieu le critère $T_{\max}$.

4.1.4 Minimiser les coûts de changement d'équipements

• (GUPTA & al) [92] étudient un problème en lequel $\mathcal{M}$ ordres de clients sont pris en compte. Chacun de ces ordres, noté par $\mathcal{O}_j$, $\forall j = 1,\ldots,\mathcal{M}$, est supposé de n_j tâches. Le nombre de tâches à ordonnancer est égale à $n = \sum_{j=1}^{\mathcal{M}} n_j$. En plus, supposant que k classes de tâches prédéfinies B_ℓ existent et $|\mathcal{O}_j \cap B_\ell| = 1$, $\forall j = 1,\ldots,\mathcal{M}$, $\forall \ell = 1,\ldots,k$. En outre, le traitement de deux tâches $J_{[i]}$ et $J_{[i+1]}$ ($[i]$ la tâche dans la $i^{\text{ième}}$ position) appartenant aux différentes classes induit un coût setup, noté par $SC_{[i],[i+1]}$, dépendant de la classe $J_{[i+1]}$ et qui est égal au temps setup correspondant. Le but est de minimiser deux critères :

- Le coût de changement d'équipements défini par $\overline{SC} = \sum_{i=2}^{n} SC_{[i-1],[i]}$. Notons que la minimisation de ce critère implique la minimisation du makespan $C_{\max}$,
- Le coût d'exécution, défini par $\overline{AC} = \sum_{j=1}^{\mathcal{M}} \max_{J_\ell, J_i \in \mathcal{O}_i} (0; C_i - C_\ell)$.

Le problème adressé est noté par $1|classes, orders, S_{sd}|Lex(\overline{SC}, \overline{AC})$. Un ordonnancement optimal pour le critère $\overline{SC}$ est tel que toutes les tâches dans une même classe sont traitées consécutivement et la minimisation du critère $\overline{AC}$ mène alors à ordonnancer les classes d'un côté, et les tâches incluses dans ces classes d'un autre côté. Un algorithme optimal en $O(nlog(\mathcal{M}))$ est proposé.

• (GUPTA & al) [92] s'intéressent ensuite au problème lexicographique opposé, noté par $1|classes, orders, S_{sd}|Lex(\overline{AC}, \overline{SC})$. Ce problème peut être réduit à un cas particulier d'un problème de voyageur de commerce pour lequel ils proposent un algorithme optimal en $O(n)$.

• (BARNES & VANSTON) [12] considèrent le problème $\mathcal{NP}$-difficile noté par $1|S_{sd}|F_\ell(\overline{SC}, \overline{C}_w)$. Les auteurs montrent qu'il peut être réduit à un problème du voyageur de commerce et ils proposent une heuristique utilisant la règle *« choix de la plus proche ville non visitée »* pour déterminer un ordonnancement. Ils proposent aussi deux algorithme optimaux. Le premier est un algorithme branch-and-bound et le second est un algorithme de programmation dynamique. Ce dernier utilise une borne inférieure pour chaque décision afin de réduire l'espace de recherche. Des résultats expérimentaux montrent que l'algorithme de programmation dynamique est plus rapide que celui de branch-and-bound. Ces résultats montrent en plus que la déviation relative entre la solution heuristique et la solution optimale est en moyen de $0,66\%$.

• (BOURGADE & al) [22] s'intéressent au problème d'ordonnancement industriel relative aux emballages en verre. Le but de ce problème est de déterminer un ordonnancement qui minimise le retard maximum $T_{\max}$ et les coûts de changements d'équipements sur la machine. Ces coûts, notés par $SC_{[i-1],[i]}$ avec $[i]$ la tâche dans la $i^{\text{ième}}$ position, sont supposés d'être dépendants dans la séquence. Le problème considéré est noté par $1|S_{sd}|F(\overline{SC}, T_{\max})$. Les auteurs proposent un programme en nombres entiers avec deux formes différentes de la fonction F. Afin de résoudre ces deux problèmes, ils proposent un algorithme branch-and-bound. Des résultats expérimentaux montrent que les solutions obtenues en minimisant l'une des deux

fonctions peuvent être des solutions dominées.

4.1.5 Minimiser des critères avec des dates échues imposées

• (CHEN & BULFIN) [33] s'intéressent aux problèmes où les tâches ont des temps de traitement unitaires. Ils considèrent les problèmes $1|p_i = 1|Lex(\overline{T}_w, Z_2^1)$, avec $Z_2^1 \in \{\overline{U}, \overline{U}_w\}$ et les problèmes $1|p_i = 1|Lex(\overline{U}, Z_2^2)$ et $1|p_i = 1|Lex(\overline{U}_w, Z_2^2)$ avec $Z_2^2 \in \{\overline{T}, \overline{T}_w\}$ qui sont modélisés sous forme des problèmes d'affectation résolus en $O(n^3)$. (CHEN & BULFIN) étudient aussi les problèmes $1|p_i = 1|Lex(\overline{T}, Z_2^1)$ pour lesquels ils proposent un algorithme glouton optimal. En plus, ils notent que pour les problèmes $1|p_i = 1|F_\ell(\overline{U}, Z_2^2)$ et $1|p_i = 1|F_\ell(\overline{U}_w, Z_2^2)$, l'algorithme de (ANEJA & NAIR) [3] peut être aisément modifié pour déterminer l'ensemble E. En effet, seulement les solutions de l'ensemble E_s peuvent être déterminées. (CHEN & BULFIN) s'intéressent aussi à la minimisation du critère $T_{\max}$ via les problèmes $1|p_i = 1|Lex(T_{\max}, Z_2^3)$, avec $Z_2^3 \in \{\overline{U}, \overline{U}_w, \overline{T}_w\}$ qui se réduisent à des problèmes d'affectation. De même concernant le problème $1|p_i = 1|\epsilon(Z_2^3/T_{\max})$.

• Le problème $1||\epsilon(\overline{U}/T_{\max})$ est considéré par (NELSON & al) [149] et pour lequel ils présentent un algorithme branch-and-bound qui détermine un sous ensemble de FE. Ils présentent aussi une heuristique qui approxime ce sous ensemble.

• SHANTIKUMAR [175] est le premier qui aborde la minimisation de deux critères $\overline{U}$ et $T_{\max}$ lorsqu'en résolvant le problème $1||Lex(\overline{U}, T_{\max})$. Ils proposent une heuristique et un algorithme branch-and-bound dans lequel des conditions de dominance sont utilisées pour éliminer des nœuds dans l'arbre de recherche. Chaque nœud est évalué par une borne inférieure, le calcul se fait en utilisant l'algorithme présenté par MOORE [145] et destiné pour le problème $1||\overline{U}$. Aucun résultat expérimental n'est représenté par SHANTIKUMAR, cependant plus tard, des travaux effectués par (GUPTA & al) [89] montrent que cet algorithme exact n'est pas capable de résoudre des problèmes de plus de 30 tâches. Plus tard, (GUPTA & RAMNARAYANAN) [90] proposent un algorithme heuristique basé sur celui de MOORE destiné pour le problème $1||\overline{U}$, et une procédure d'amélioration interchangeable. Jusqu'à maintenant, la complexité du problème $1||Lex(\overline{U}, T_{\max})$ est ouverte mais supposée d'être non polynomiale par (GUPTA & al) [89] qui proposent un algorithme de branch-and-bound pour le résoudre.

4.1.6 Minimisation des critères Juste-à-Temps

La notion d'ordonnancement *« Juste à Temps »* (JAT) est de plus en plus importante dans les systèmes de production modernes car elle permet de produire des modèles qui permettent de diminuer les quantités de produits devant être stockées. Ces modèles se distinguent des modèles classiques de l'ordonnancement dans le fait qu'ils pénalisent toute tâche se terminant en avance par rapport à une date de fin idéale. La plupart des modèles actuels se basent sur la combinaison linéaire des coût d'avance et de retard de chaque tâche.

• Les problèmes JAT constituent une partie plus importante dans la littérature des problèmes d'ordonnancement multicritère. HOOGEVEEN [102] s'intéresse aux problèmes $1|s_i \in [d_i - p_i, d_i], nmit|\epsilon(L_{\max}/P_{\max})$ et $1|s_i \in [d_i - p_i, d_i]|\epsilon(L_{\max}/P_{\max})$, où l'avance des tâches est exprimée par une relation en fonction de la date du début désirée s_i et la date du début réelle t_i où $P_{\max} = \max_{1 \leq i \leq n}(s_i - t_i)$. L'auteur montre que ces problèmes sont polynomialement résolus. Lorsque la contrainte $d_i - s_i \leq p_i$ ne se présente pas, ces problèmes deviennent $\mathcal{NP}$-difficiles au sens fort. Un algorithme en $O(nlog(n))$ est proposé pour le problème $1|s_i \in [d_i - p_i, d_i], nmit|\epsilon(L_{\max}/P_{\max})$. En suite, Un algorithme calculant l'ensemble E est introduit et l'auteur montre que la cardinalité de cet ensemble est tout au plus n. Pour le problème $1|s_i \in [d_i - p_i, d_i]|\epsilon(L_{\max}/P_{\max})$, un algorithme en $O(n)^2log(n))$ est proposé.

• Le problème $1||F_\ell(\overline{E}, \overline{T})$ avec $F_\ell(\overline{E}, \overline{T}) = \overline{E} + \overline{T}$, est $\mathcal{NP}$-difficile. (GAREY & al) [76] proposent un algorithme en $O(nlog(n))$ pour calculer les dates des débuts des tâches lorsque la séquence des ces dernières est fixée. En outre, les auteurs montrent que cet algorithme peut être utilisé dans la résolution des problèmes avec des contraintes supplémentaires.

• Le problème $1|d_i = d \geq \sum_{i=1}^{n} p_i, nmit|F_\ell(\overline{E}, \overline{T})$ avec $F_\ell(\overline{E}, \overline{T}) = \overline{E} + \overline{T}$, est étudié par KANET [116] qui propose un algorithme en $O(n^2)$.

• Le problème $1|d_i = d \geq \delta, nmit|F_\ell(\overline{E}, \overline{T})$ avec $F_\ell(\overline{E}, \overline{T}) = \overline{E} + \overline{T}$, est abordé par (BAGCHY & al) [6]. Ils considèrent le cas où $d \geq \delta$ avec $\delta = p_1 + p_3 + \cdots + p_n$ si n est impair ou $\delta = p_2 + p_4 + \cdots + p_n$ si n est pair. Les auteurs proposent un algorithme en temps polynomial pour la résoudre.

• Pour résoudre les problèmes $1|p_i = 1, nmit|\epsilon(\overline{E}/\overline{U})$, $1|p_i = 1, nmit|\epsilon(E_{\max}/\overline{U})$ et $1|p_i = 1, nmit|\epsilon(F_\ell(\overline{E}, \overline{T})/\overline{U})$ avec $F_\ell(\overline{E}, \overline{T}) = \overline{E} + \overline{T}$, (KONDAKSY & al) [122] proposent quelques programmes linéaires. En suite, des algorithmes sont mis en œuvre pour énumérer l'ensemble des optima de Pareto faibles en considérant d'abord que la valeur du critère $\overline{U}$ correspond à une solution optimale du problème monocritère dont $\overline{E}$, $E_{\max}$ ou $\overline{E} + \overline{T}$ étant son objectif. Le problème bicritère est alors résolu en réduisant la valeur de borne supérieure du critère $\overline{U}$ en chaque itération.

• (AHMED & SUNDARARAGHAVAN) [1] étudient le problème de minimisation de l'avance et du retard pondérés lorsque les poids sont *symétriques*, dépendant des tâches et sont égaux au durées d'exécution. Ce problème est noté par $1|d_i = d \geq \sum_{i=1}^{n} p_i, nmit|F_\ell(\overline{E}_w, \overline{T}_w)$ avec $F_\ell(\overline{E}_w, \overline{T}_w) = \overline{E}_w + \overline{T}_w$. Les deux auteurs proposent un algorithme optimal pour le résoudre.

• (GAREY & al.) [76] étudient un problème JAT particulier dont lequel les tâches ont des durées d'exécution unitaires. Ce problème est noté par $1|p_i = 1|F(E_i, T_i)$ avec $F(E_i, T_i) = \max_{1 \leq i \leq n}(E_i + T_i)$. Pour sa résolution, GAREY, TARJAN et WILFONG proposent un algorithme optimal en $O(nlog(n))$.

• Peu de problèmes d'ordonnancement JAT avec plus de deux critères sont adressés dans la littérature. (SEIDMANN & al.) [170] traite le problème tricritère en lequel les dates échues sont inconnues mais doivent être plus proches d'une date échue commune. Le problème est noté par $1|d_i \; inconnue, \; nmit, A|F_\ell(\overline{E}, \overline{T}, \overline{A})$ avec $F_\ell(\overline{E}, \overline{T}, \overline{A}) = \alpha\overline{E} + \beta\overline{T} + \gamma\overline{A}$. Le critère $\overline{A}$ est défini par $\overline{A} = \sum_{i=1}^{n} \max\{0, d_i - A\}$, où A est une date échue dont nous voulons pas la dépasser. Pour résoudre ce problème, les auteurs propose un algorithme optimal en $O(nlog(n))$. Les tâches sont ordonnées suivant la règle SPT. Les dates échues optimales sont alors calculées de la manière suivante, $\forall i = 1, \ldots, n$:

$$d_i = \begin{cases} \sum_{j=1}^{i} p_j & \text{si } \gamma \leq \beta \\ \min(A; \sum_{j=1}^{i} p_j) & \text{sinon} \end{cases}$$

Un problème similaire, mais avec une date échue commune et les critères $\overline{U}_w$, $\overline{A}$, est abordé

par (DE & al.) [45]. le problème est noté par $1|d_i = d, A|F_\ell(\overline{U}_w, \overline{A})$ avec $F_\ell(\overline{U}_w, \overline{A}) = \overline{U}_w + \gamma\overline{A}$. Soit $M = \sum_{j/\frac{w_j}{p_j}>\gamma} p_j$, si $A < M$ alors le problème est $\mathcal{NP}$-difficile. Dans ce cas, DE, GHOSH et WELLS propose une heuristique basée sur une relaxation d'un modèle du problème. Si $A \geq M$ alors le problème est résolu en temps polynomial. Un algorithme optimal est proposé par les auteurs.

• Le problème considéré dans la littérature comme étant la base des problèmes d'ordonnancement JAT est noté par $1||F_\ell(\overline{T}, \overline{E})$. Il est $\mathcal{NP}$-difficile tant que le problème $1||\overline{T}$ est ainsi. SZWARC [185] étudie le l'enchaînement de deux tâches si elles doivent être accomplies consécutivement et sans temps arrêt machine entre deux traitements. Les tâches du problème original sont mises en blocs où les tâches de chaque bloc sont ordonnancées consécutivement et deux blocs successifs sont séparés par un temps arrêt machine. Les conditions suffisantes proposées sont alors utilisées pour ordonnancer les tâches à l'intérieur des blocs. SZWARC propose une procédure branch-and-bound qui ne considère pas les ordonnancements dominés par les conditions précédentes. Il considère aussi le cas particulier où $d_i = d$, $\forall i = 1, \ldots, n$. En outre, (AZIZOGLU & al.) [4] proposent pour le problème $1|nmit|F_\ell(\overline{T}, \overline{E})$ une adaptation de l'heuristique présentée par (OW & MORTON) [153] $1|nmit|F_\ell(\overline{T}^\alpha, \overline{E}^\beta)$. Une procédure branch-and-bound est ainsi proposée et des résultats expérimentaux montrent que plusieurs problèmes avec plus de 20 tâches peuvent être résolus.

(KIM & YANO) [119] traitent le problème $1||F_\ell(\overline{E}, \overline{T})$ avec $F_\ell(\overline{E}, \overline{T}) = \overline{E} + \overline{T}$, et proposent des heuristiques et un algorithme exact pour le résoudre. Des résultats expérimentaux montrent que cet algorithme exact est limité aux problèmes contenant environs 20 tâches.

Ce problème est réabordé par (FRY & al.) [69] qui proposent une procédure branch-and-bound basée sur la subdivision du problèmes en blocs. Des résultats expérimentaux montrent que cet algorithme résout des problèmes avec plus de 25 tâches.

Lorsque l'insertion d'un temps arrêt machine est interdit (contrainte *nmit*), (FRY & LEONG) [72] propose un programme linéaire en nombres entiers pour résoudre ce problème.

• (SUNDARARAGHAVAN & AHMED) [182] étudient le problème à date échue commune, noté par $1|d_i = d,\ nmit|F_\ell(\overline{T}, \overline{E})$. La date échue d peut être restrictive qui rend le problème $\mathcal{NP}$-difficile, car le problème $1|d_i = d < \sum_{i=1}^{n} p_i,\ nmit|F_\ell(\overline{T}, \overline{E})$ est ainsi. Les deux auteurs

proposent une heuristique basée sur une méthode gloutonne inspirée de l'algorithme proposé par (KANET) [117] pour le cas $d \geq \sum_{i=1}^{n} p_i$.

Pour le problème avec une date échue restrictive commune notée par $1|d_i = d < \sum_{i=1}^{n} p_i,\ nmit|F_\ell(\overline{T}, \overline{E})$, (BAGCHI & al.) [7] proposent une procédure branch-and-bound. Notons que la notions de date échue restrictive commune est mentionnée dans (BAGCHI & al.) [6] qui s'intéressent au problème $1|d_i = d < \delta,\ nmit|F_\ell(\overline{T}, \overline{E})$ avec $F_\ell(\overline{T}, \overline{E}) = \overline{T} + \overline{E}$. La borne δ est définie par $\delta = p_1 + p_3 + \cdots + p_n$ si n est impair et $\delta = p_2 + p_4 + \cdots + p_n$ si n est pair (en supposant que $p_1 \geq \cdots \geq p_n$). Cette quantité représente la valeur de la date d ci-dessous pour laquelle le problème est restrictif. Clairement, nous avons $\delta < \sum_{i=1}^{n} p_i$.

Pour résoudre le problème $1|d_i = d, t_0 = 0,\ nmit|F_\ell(\overline{E}, \overline{T})$ avec t_0 : la date du début de l'ordonnancement, (SUNDARARAGHAVAN & AHMED) proposent une procédure branch-and-bound. SZWARC [184] traite encore ce problème et fournit ensuite une procédure branch-and-bound. Des résultats expérimentaux montrent que l'algorithme peut résoudre des problèmes avec au moins 25 tâches. Finalement, SZWARC s'intéresse au cas où la date du début de la séquence n'est pas fixée. Il montre, à l'aide d'un exemple, que contrairement aux revendications de (BAGCHI & al.) [6], des ordonnancements optimaux existent et pour lesquels $t_0 \neq 0$.

• Lorsque l'avance est mesurée en comparaison avec les dates du début désirées, KOULAMAS [123] étudie le problème $1|s_i|F_\ell(\overline{T}, \overline{P})$ avec $F_\ell(\overline{T}, \overline{P}) = \overline{T} + \overline{P}$. L'auteur montre que ce problème est $\mathcal{NP}$-difficile et il propose sept heuristiques pour le résoudre aussi bien qu'un algorithme optimal basé sur une méthode d'énumération. Des conditions de dominance permettant à l'algorithme d'énumération d'être plus efficace, sont aussi décrites. Des résultats expérimentaux montrent que deux de ces heuristiques donnent des résultats proches de la solution optimale.

• Le problème $1||F_\ell(\overline{E}^{\alpha}, \overline{T}^{\beta})$ est la généralisation du problème d'ordonnancement JAT de base. Ce problème est fortement $\mathcal{NP}$-difficile. Afin de calculer une solution optimale, (FRY & al.) [68] proposent une heuristique basée sur un algorithme d'amélioration d'une séquence initiale par permutation des tâches. (FRY & BLACKSTONE) [70] reprennent

ce problème dans le contexte de la méthode d'organisation de la production « Optimised Production Technology ». Il proposent un programme linéaire en variables mixtes à un problème d'ordonnancement en une seule machine. (JAMES & BUCHANAN) [109] et [110] proposent plusieurs implantations de deux approches différentes en utilisant des algorithmes de recherche tabou.

• Lorsque les tâches ont des dates de disponibilité distinctes, le problème est noté par $1|r_i|F_\ell(\overline{E}^\alpha, \overline{T}^\beta)$ et il est fortement $\mathcal{NP}$-difficile. (YANO & KIM) [206] considère un cas particulier où la fonction objectif est définie par $F_\ell(\overline{E}^\alpha, \overline{T}^\beta) = \sum_{i=1}^{n}(\alpha_i E_i + \beta_i T_i)$ avec α_i et β_i sont en fonction des durées d'exécution. Les deux auteurs proposent cinq heuristiques et un algorithme branch-and-bound. (MAZZINI & ARMENTANO) [138] traitent le problème général avec des poids ordinaires et proposent une heuristique gloutonne en $O(n^3)$ et une heuristique de recherche locale. Cette dernière utilise comme ordonnancement initial la solution obtenue par l'heuristique gloutonne. Des résultats expérimentaux montrent que l'heuristique de recherche locale n'améliore pas beaucoup l'ordonnancement obtenu par l'heuristique gloutonne. L'amélioration est en moyenne 0,3%. Ce problème est réabordé par (SOURD & KEDAD SIDHOUM) [179]. Les deux auteurs proposent un nouveau algorithme branch-and-bound pouvant résoudre des instances avec plus de 50 tâches. Il peut aussi résoudre même des problèmes avec plusieurs fonctions coût générales non-convexes. L'algorithme est basé sur la combinaison d'une relaxation lagrangienne des contraintes de ressources et des nouvelles règles de dominance.

• (GUPTA & SEN) [95] s'intéressent au problème $1|nmit|F(E_i, T_i)$ avec $F(E_i, T_i) = \sum_{i=1}^{n}(E_i + T_i)^2$. Pour sa résolution, ils proposent une procédure branch-and-bound où chaque nœud est évalué par une borne inférieure calculée en utilisant la règle SPT et un algorithme de voisinage. En outre, une heuristique présentant un algorithme branch-and-bound, où certains nœuds ne sont pas explorés, est ainsi proposée. Quelques résultats expérimentaux montrent que cette heuristique est très efficace (en terme de temps et de qualité).

(BAGCHI & al.) [8] s'intéressent à ce problème lorsque toutes les tâches ont la même date échue qui est à déterminer. Le problème obtenu est noté par $1|d_i = d,\ nmit|F(E_i, T_i)$. Deux cas peuvent être distingués ; *i.e.* le problème avec $d \geq \sum_{i=1}^{n} p_i$ (cas non restrictif) et le problème

avec $d < \sum_{i=1}^{n} p_i$ (cas restrictif). Pour chaque cas, une procédure branch-and-bound calculant l'ordonnancement optimal et la date échue est proposée. Des conditions de dominance sont aussi présentées et elles permettent de résoudre le second problème d'une manière efficace. (GUPTA & SEN) montrent que dans le cas où $d = \frac{1}{n}\sum_{i=1}^{n} p_i$ leur algorithme est plus rapide que celui présenté par (EILON & CHOWDHURY) [57].

Problèmes ouverts

Peu de problèmes d'ordonnancement JAT ont une complexité ouverte et dans ce cas quelques problèmes très particuliers seront considérés :

• La minimisation de la plus grande déviation entre les tardivités algébriques mène à la détermination d'un ordonnancement JAT. Le problème $1|nmit|F_\ell(L_{\max}, L_{\min})$ avec $F_\ell(L_{\max}, L_{\min}) = L_{\max} - L_{\min}$ est traité par (GUPTA & SEN) [96]. Les deux auteurs considèrent seulement les ordonnancements semi-actifs qui n'est pas dominant pour ce problème et proposent un algorithme branch-and-bound pour le résoudre. une amélioration des bornes utilisées dans ce problème est proposée par (TEGZE & VLACH) [187].

(LIAO & HUANG) [131] proposent à ce problème un algorithme en $O(n^2\bar{p}\log(n))$ où $\bar{p} = \sum_{i=1}^{n} p_i$.

• (SEN & al.) [173] qui s'intéressent au problème $1|nmit|F_\ell(\overline{C}, L_{\max} - L_{\min})$ proposent un algorithme branch-and-bound afin d'énumérer l'ensemble des optima supportés de Pareto stricts. Des résultats expérimentaux montrent que pour un problème avec 9 tâches, le nombre moyen des optima de Pareto stricts est entre 5.2 et 8.3 pour un temps d'exécution moyen entre 2.4 et 38.1 seconds.

4.2 Problèmes d'ordonnancement sur machines parallèles

Ce sont les problèmes les plus importants via leur apparition dans la pratique, *i.e.* il existe souvent des ressources multiples dédiées aux traitement de quelques opérations (en industrie par exemple). En outre, les algorithmes de résolution pour tels problèmes peuvent

être utilisés dans des algorithmes plus généraux destinés à des problèmes d'ateliers multi-étages. Peu de problèmes d'ordonnancement multicritère sur machines parallèles ont été traités dans la littérature.

4.2.1 Problèmes avec machines parallèles identiques

4.2.1.1 Le problème $P2|pmtn|\epsilon(L_{\max}/C_{\max})$

(MOHRI & al.) [143] s'intéressent au problème d'ordonnancement bicritère où deux machines sont disponibles pour l'exécution de n tâches indépendantes qui pouvant être interrompues en importe quel moment. Chaque tâche J_i est définie par une durée de traitement p_i et une date échue d_i. Sans aucune perte de généralités, nous supposons que $d_1 \leq d_2 \leq \cdots \leq d_n$. Le but est d'ordonnancer ces tâches de telle manière que le makespan $C_{\max}$ et le décalage temporel maximal $L_{\max}$ soient minimisés. En considérant l'approche ϵ-contrainte, ils fournissent une caractérisation de vecteurs critères strictement non dominés. Ce problème est résolu en temps polynomial.

D'abord, MOHRI, MASUDA et ISHII abordent le problème $P2|pmtn|L_{\max}$ qui peut être résolu itérativement en considérant les problèmes $P2|pmtn, \widetilde{d}_i|-$, en utilisant l'algorithme de SAHNI [164]. À chaque itération, un problème $P2|pmtn, \widetilde{d}_i = d_i + L|-$ est résolu : Si une solution réalisable de ce problème existe, alors un ordonnancement pour lequel la valeur du critère $L_{\max}$ est égale à L existe. Autrement, il n'existe pas un tel ordonnancement. Commençant par l'algorithme de SAHNI, les trois auteurs montrent le résultat suivant :

Lemme 4.2.1. *[143]*
La valeur optimale du critère $L_{\max}$ dans le problème $P2|pmtn|L_{\max}$ est donnée par :

$$L^*_{\max} = \frac{1}{2} \max_{i=1,\ldots,n} \Big(\sum_{j=1}^{i} p_j - \min_{k=1,\ldots,i} \big(d_k + \sum_{j=k+1}^{i} p_j\big) - \min_{k=1,\ldots,i-1} \big(d_k + \sum_{j=k+1}^{i-1} p_j\big)\Big),$$

sous la supposition que $p_i \leq d_i$, $\forall i = 1, \ldots, n$.

Ce résultat est étendu au problème $P2|pmtn|\epsilon(L_{\max}/C_{\max})$ lorsque la contrainte sur le critère $C_{\max}$ est fixée ; *i.e* lorsque $C_{\max} \leq \epsilon$. Dans ce cas, le résultat suivant intervient :

Lemme 4.2.2. *[143]*
La valeur optimale du critère $L_{\max}$ dans le problème $P2|pmtn, C_{\max} \leq \epsilon|L_{\max}$, notée $L^{\epsilon}_{\max}$, est

donnée par :

$$L^{\epsilon}_{\max} = \max\{L^{*}_{\max},\ \max_{i=1,\ldots,n}(\sum_{j=1}^{i} p_j - \epsilon - \min_{k=1,\ldots,i-1}(d_k + \sum_{j=k+1}^{i-1} p_j))\},$$

sous la supposition que $p_i \leq d_i$, $\forall i = 1, \ldots, n$.

La résolution du problème $P2|pmtn|\epsilon(L_{\max}/C_{\max})$ avec une valeur fixe de ϵ revient à la résolution du problème $P2|pmtn|L_{\max}$. La seule différence réside dans la construction des dates limites à chaque itération. Pour le problème bicritère, nous avons $\widetilde{d_i} = \min\{d_i + L, \epsilon\}$, $\forall i = 1, \ldots, n$. Alors, en utilisant l'algorithme de SAHNI, nous déterminons un ordonnancement réalisable pour ces dates limites. Par conséquent, un ordonnancement existe pour lequel le makespan est inférieur à ϵ et la valeur du décalage temporel maximal $L_{\max}$ égale à L.

MOHRI, MASUDA et ISHII proposent une caractérisation de l'ensemble E. Ils identifient une condition suffisante pour l'existence d'un vecteur critère unique strictement non dominé. Cette condition peut être considérée comme étant une conséquence de l'expression mathématique de $L^{\epsilon}_{\max}$.

Théorème 4.2.3. *[143]*
Soit $F = \max_{i=1,\ldots,n}(\sum_{j=1}^{i} p_j - \min_{k=1,\ldots,i-1}(d_k + \sum_{j=k+1}^{i-1} p_j))$. *Si* $L^{*}_{\max} \geq F - C^{*}_{\max}$, *alors il existe un vecteur critère unique strictement non dominé défini par* $[C^{*}_{\max}; L^{*}_{\max}]^T$. $C^{*}_{\max}$ *est la valeur optimal du makespan dans le problème* $P2|pmtn|C_{\max}$, *donnée par [139] :*

$$C^{*}_{\max} = \max(\max_{i=1,\ldots,n} p_i; \frac{1}{2}\sum_{i=1}^{n} p_i).$$

Si la condition du théorème 4.2.3 n'est pas vérifiée, l'ensemble E se restreint, dans l'espace de critères, à un segment de droite limité par les vecteurs critères $[C^{*}_{\max}; F - C^{*}_{\max}]^T$ et $[C^{+}_{\max}; L^{*}_{\max}]^T$, où $C^{+}_{\max}$ est la valeur minimale du makespan calculée à partir d'un ordonnancement optimal par rapport au critère $L_{\max}$. La valeur $C^{+}_{\max}$ peut être calculée à partir d'un ordonnancement réalisable déterminé par l'algorithme de SAHNI en considérant le problème $P2|pmtn, \widetilde{d_i} = d_i + L^{*}_{\max}|-$. On outre, pour n'importe quel vecteur critère $[C; L]^T$ sur le segment de droite, un ordonnancement correspondant est obtenu en utilisant l'algorithme de SAHNI avec $\widetilde{d_i} = \min\{d_i + L; C\}$, $\forall i = 1, \ldots, n$.

Exemple illustratif

Nous considérons un problème avec $n = 5$ tâches :

J_i	J_1	J_2	J_3	J_4	J_5
p_i	3	7	4	8	10
d_i	5	10	12	13	15

☞ Nous avons

$C^*_{\max} = \max\{\max\{3,7,4,8,10\}, \frac{1}{2}(3+7+4+8+10)\} = \max\{10,16\} = 16$ et,

$L^*_{\max} = \frac{1}{2}\max\{3-5, 10-\min\{5+7,10\}-5, 14-\min\{5+7+4,10+4,12\}-\min\{5+7,10\}, 22-\min\{5+7+4+8,10+4+8,12+8,13\}-\min\{5+7+4,10+4,12\}, 32-\min\{5+7+4+8+10,10+4+8+10,12+8+10,13+10,15\}-\min\{5+7+4+8,10+4+8,12+8,13\}\} = \frac{1}{2}\max\{-2,-5,-8,-3,4\} = 2.$

☞ $F = \max\{3, 10-5, 14-\min\{5+7,10\}, 22-\min\{5+7+4,10+4,12\}, 32-\min\{5+7+4+8,10+4+8,12+8,13\}\} = \max\{3,5,4,10,19\} = 19.$

Nous n'avons pas $L^*_{\max} = 2 \geq F - C^*_{\max} = 19 - 16 = 3$, alors l'ensemble E est défini, dans l'espace des critères, par un segment de droite. La première extrémité de ce segment est $[C^*_{\max}; F - C^*_{\max}]^T = [16,2]^T$. Pour obtenir la deuxième extrémité, nous avons besoin de calculer $C^+_{\max}$ en utilisant l'algorithme de SAHNI avec $\widetilde{d}_i = d_i + 2$, $\forall i = 1,\ldots,n$. L'ordonnancement obtenu est présenté sur la FIG.4.1 suivante, avec $C^+_{\max} = 17$:

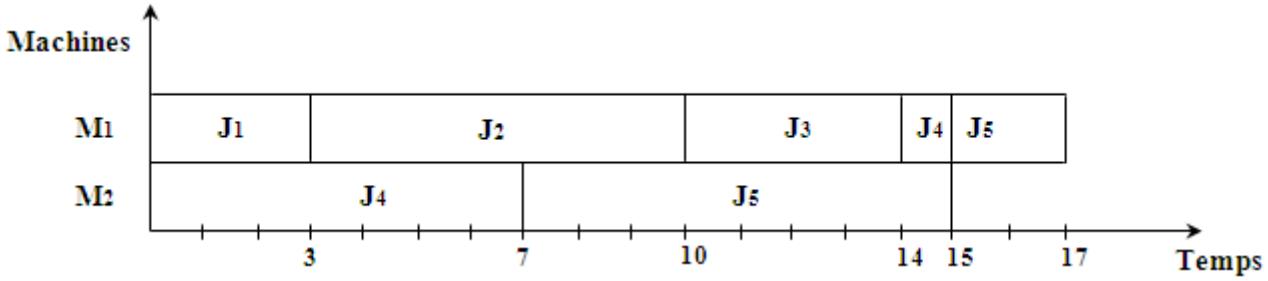


FIG. 4.1 – L'ordonnancement déterminé pour l'obtention de $C^+_{\max}$.

La deuxième extrémité est alors $[17,2]^T$ et le segment de droite est illustré par la FIG.4.2 suivante :

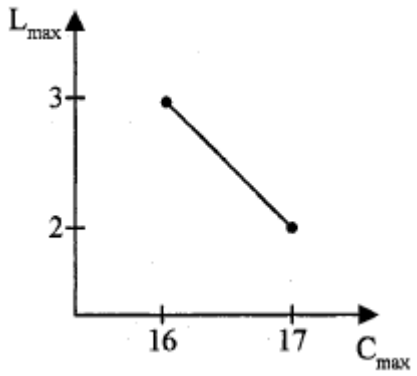


FIG. 4.2 – L'ensemble E dans l'espace des critères.

4.2.1.2 Le problème $P3|pmtn|\epsilon(L_{\max}/C_{\max})$

(MOHRI & al.) [143] considèrent l'extension du problème précèdent à un cas de trois machines. n tâches indépendantes doivent être traitées et pouvant être interrompues en import quel moment.Chaque tâche J_i est définie par une durée de traitement p_i et une date échue d_i. Sans aucune perte de généralités, nous supposons que $d_1 \leq d_2 \leq \cdots \leq d_n$. Comme pour le problème à deux machines, ils fournissent une caractérisation de l'ensemble des optima de Pareto stricts, en considérant le problème $\epsilon(L_{\max}/C_{\max})$. Ce problème est résolu en temps polynomial.

4.2.1.3 Le problème $P2||Lex(T_{\max}, \overline{U})$

(SARIN & HARIHARAN) [166] abordent le problème d'ordonnancement de n tâches indépendantes sur deux machines parallèles et lorsque aucune préemption n'est autorisée. Le but est de minimiser le retard maximum et ensuite le nombre de tâches en retard. Sans aucune perte de généralité, nous supposons que les tâches sont de telle manière que $d_1 \leq d_2 \leq \cdots \leq d_n$ (une tâche est indexée première celle possédant la plus petite durée p_i). Comme le problème $P2||T_{\max}$ est $\mathcal{NP}$-difficile, le problème bicritère est pareil. Les deux auteurs proposent une heuristique pour le résoudre.

D'abord, ils considèrent seulement la minimisation du critère $T_{\max}$ et proposent une heuristique. Ils rappellent qu'il existe un ordonnancement optimal pour ce critère dont lequel, sur chaque machine, les tâches sont ordonnancées suivant la règle EDD. Commençant par ce résultat, ils construisent un ordonnancement initial en utilisant la règle EDD-FAM. Cet ordonnancement est ensuite amélioré par une procédure de voisinage.

4.2.1.4 Le problème $P||\#(\overline{C}, \overline{U})$

La minimisation des deux critères $\overline{C}$ et $\overline{U}$ sur m machines identiques est abordée par (RUIZ-TORRES & al.) [163] qui s'intéressent à l'optimisation des optima de Pareto stricts. Comme le problème $P||\overline{U}$ est $\mathcal{NP}$-difficile, le problème bicritère est ainsi et RUIZ-TORRES, ENSCORE et BARTON proposent quatre heuristiques pour le résoudre.

Les quatre heuristiques procèdent par des améliorations itératives des ordonnancements initiaux. Pour chacune de ces heuristiques, deux recherches sont faites commençant par deux ordonnancements différents. Le premier ordonnancement initial est obtenu en appliquant la règle SPT-FAM. Le second ordonnancement initial est obtenu en appliquant les heuristiques de (HO & CHANG) [101] au problème $P||\overline{U}$. Pour chacune de ces quatre heuristiques, le voisinage de la solution courante est obtenu en effectuant des permutations de deux-tâches et des insertions d'une seule tâche.

4.2.1.5 Le problème $P|pmtn|Lex(\overline{C}, C_{\max})$

(LEUNG & YOUNG) [130] s'intéressent au problème bicritère où les n tâches peuvent être interrompue en importe quel moment. le but est de les ordonnancer de telle manière que le makespan $C_{\max}$ et la moyenne des dates de fin d'exécution $\overline{C}$ sont à optimiser. Plus précisément, un ordonnancement $\mathcal{S}$ est recherché qui est optimal pour l'ordre lexicographique $Lex(\overline{C}, C_{\max})$. Ce problème est résolu en temps polynomial.

4.2.2 Problèmes avec machines parallèles uniformes

4.2.2.1 Le problème $Q|p_i = p|\epsilon(f_{\max}/g_{\max})$

(TUZIKOV & al.) [192] étudient un problème d'ordonnancement où n tâches doivent être ordonnancées sur m machines M_j possédant des différentes vitesses de traitement. La vitesse de traitement de la machine M_j est notée k_j et le temps de traitement de J_i sur M_j est égale à $\frac{p_i}{k_j}$. La particularité de ce problème est que toutes les tâches possèdent la même durée de traitement p. Le but est de minimiser deux fonctions maximum $f_{\max}$ et $g_{\max}$ définies par $f_{\max} = \max_{i=1,\ldots,n}(\Phi_i(C_i))$ et $g_{\max} = \max_{i=1,\ldots,n}(\Psi_i(C_i))$, où Φ_i, Ψ_i sont des fonctions croissantes. En basant sur l'approche ϵ-contrainte, les auteurs proposent un algorithme optimal en temps polynomial pour l'énumération de l'ensemble des optima de Pareto stricts.

4.2.2.2 Le problème $Q|p_i = p|\epsilon(\overline{g}/f_{\max})$

(TUZIKOV & al.) [192] étudient un problème d'ordonnancement où n tâches doivent être ordonnancées sur m machines M_j possédant des différentes vitesses de traitement notée par k_j. En particulier, toutes les tâches possèdent la même durée de traitement p. Le but est de minimiser deux fonctions $f_{\max}$ et $\overline{g}$ définies par $f_{\max} = \max_{i=1,\ldots,n}(\Phi_i(C_i))$ et $\overline{g} = \sum_{i=1}^{n}(\Psi_i(C_i))$, où Φ_i, Ψ_i sont des fonctions croissantes. Les auteurs proposent un algorithme optimal en temps polynomial pour l'énumération de l'ensemble des optima de Pareto.

4.2.2.3 Le problème $Q|pmtn|\epsilon(\overline{C}/C_{\max})$

(MC CORMICK & PINEDO) [42] étudient un problème d'ordonnancement où une vitesse de traitement est associée à chaque machine M_j de vitesse k_j. Le but est de minimiser le makespan et la somme des dates de fin de traitement des tâches. La préemption des tâches est autorisée en importe quel moment t ($t \in \mathbb{R}$). Nous supposons que les tâches et les machines sont numérotées telles que $p_1 \leq p_2 \leq \cdots \leq p_n$ et $k_1 \leq k_2 \leq \cdots \leq k_m$. Le problème $Q|pmtn|\overline{C}$ peut être résolu optimallement par la règle SRPT-FM. le problème $Q|pmtn|C_{\max}$ peut être résolu optimallement par la règle LRPT-FM. Contrairement à l'algorithme précédant, les tâches peuvent être interrompues pendant un temps supérieurs à zéro.

Pour résoudre ce problème bicritère, (MC CORMICK & PINEDO) déterminent l'ensemble des optima de Pareto stricts. Pour ce faire, ils considèrent l'approche ϵ-contrainte et ils résolvent le problème suivant :

$$\begin{cases} \min \overline{C}(s) \\ \qquad C_{\max} \leq \epsilon; \\ \qquad s \in \mathcal{S}. \end{cases}$$

(MC CORMICK & PINEDO) proposent une implantation en $O(mn)$.

4.2.3 Problèmes avec machines parallèles générales

4.2.3.1 Le problème $R|p_{i,j} \in [\underline{p}_{i,j}; \overline{p}_{i,j}], prmu|F_\ell(\overline{C}, \overline{CC}_w)$

(ALIDAEE & AHMADIAN) [2] s'intéressent au problème d'ordonnancement où les durées d'exécution dépendent de la machine sur laquelle les tâches sont exécutées. On outre, ces

dates d'exécution ne sont pas données et nous avons seulement $p_{i,j} \in [\underline{p}_{i,j}; \overline{p}_{i,j}]$, $\forall i = 1, \ldots, n$, $\forall j = 1, \ldots, m$. La valeur exact $p_{i,j}$ est à déterminer en considérant la minimisation d'une combinaison linéaire des critères $\overline{C}$ et $\overline{CC}_w$. Le critère $\overline{CC}_w$ est le coût total du temps dépréciation, défini par :

$$\overline{CC}_w = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{i,j} x_{i,j},$$

avec $w_{i,j}$ le coût d'une unité du temps de compression de la tâche J_i sur la machine M_j. $x_{i,j}$ est le nombre d'unités du temps de compression de la tâche J_i sur la machine M_j, ce qui donne $p_{i,j} = \overline{p}_{i,j} - x_{i,j}$. La complexité de ce problème est ouverte.

4.3 Problèmes d'ateliers

4.3.1 Problèmes Flowshop à deux machines

Dans cette sous section, nous nous intéressons aux problèmes d'ordonnancement flowshop multicritère à deux machines. Chaque tâche J_i est exécutée sur la machine M_1 pour une durée $p_{i,1}$, en suite sur la machine M_2 pour une durée $p_{i,2}$. Dans ce contexte, le problème d'ordonnancement le plus adressé dans la littérature concerne la minimisation des critères $\overline{C}$ et $C_{\max}$. Des différents problèmes d'ordonnancement sont présentés par rapport à la forme de la fonction objectif.

4.3.1.1 Le problème $F2|prmu|Lex(C_{\max}, \overline{C})$

• Ce problème est fortement $\mathcal{NP}$-difficile [35], et RAJENDRAN [156] proposent deux heuristiques et un algorithme exact pour le résoudre. Ces deux heuristiques utilisent l'algorithme de JOHNSON [114] afin de déterminer une séquence initiale optimale par rapport au critère $C_{\max}$. En suite, elles utilisent des permutations des tâches adjacentes qui sont choisies par deux indicateurs généralement définis pour une séquence $\mathcal{S}$ et une position r. le but de ces indicateurs est de réduire la valeur du critère $\overline{C}$ de l'ordonnancement $\mathcal{S}$. La prochaine tâche à permuter est déterminée par rapport à ces indicateurs.

RAJENDRAN propose aussi un algorithme branch-and-bound pour la détermination d'une solution optimale. des résultats numériques montrent que cet algorithme est limité aux

problèmes avec au plus 15 tâches.

L'une des heuristiques proposées a l'avantage d'être optimale si $\min_{i=1,\ldots,n}(p_{i,1}) > \max_{i=1,\ldots,n}(p_{i,2})$. Cette configuration correspond au cas où *la machine* M_1 *domine la machine* M_2.

• (GUPTA & al) [93] proposent neuf heuristiques et un algorithme optimal pour résoudre le problème $F2|prmu|Lex(C_{\max}, \overline{C})$. Ils étudient deux cas particuliers pour lesquels un ordonnancement optimal peut être construit en temps polynomial.

Les neuf heuristiques présentées par les trois auteurs sont des algorithmes gloutons et de voisinage. Des résultats numériques montrent qu'une de ces heuristiques domine les autres. elle est de complexité $O(n^4)$. Autres résultats numériques montrent aussi que cette heuristique est meilleures que les deux heuristiques proposées par RAJENDRAN [156] et (CHEN & BULFIN) [35].

Des comparaisons avec l'algorithme optimal sont présentées pour des problèmes avec au plus 10 tâches.

• (T'KINDT & al) [190] proposent une heuristique pour la résolution du problème $F2|prmu|Lex(C_{\max}, \overline{C})$, qui est une extension de celle présentée (NAGAR & al) [147] et en suite par (BILLAUT & al) [18]. Cette heuristique procède en deux étapes : Dans la première étape, de complexité $O(n^3)$, un ordonnancement est construit par un algorithme glouton. Dans la deuxième étape, cet ordonnancement est amélioré par une recherche locale.

T'KINDT, GUPTA et BILLAUT modélisent le problème sous forme d'un programme en variables mixtes et étudient aussi plusieurs méthodes exactes . Ils proposent un algorithme de programmation dynamique et un algorithme de branch-and-bound. Ce dernier est le plus performant.

Des résultats numériques montrent que l'heuristique de (T'KINDT & al) [190] est meilleure que celle de (GUPTA & al) [93] en terme de temps et de qualité. En outre, l'algorithme de branch-and-bound proposé résout des problèmes contenant au plus 35 tâches dans les cas les plus difficiles.

• (T'KINDT & al) [191] exploitent l'efficacité des algorithme des colonies de fourmis à ce problème. L'idée de base de ces algorithmes vient de l'habileté des fourmis à trouver le plus

court chemin entre leurs nids et les localisations de nourriture.

• **Autres résultats**

Plusieurs heuristiques de voisinage à base de population ont été aussi présentées dans la littérature. (NEPPALLI & al) [150] proposent des algorithmes génétiques pour la résolution de ce problème. L'algorithme le plus performant est obtenu en considérant que chaque chromosome de la population est évalué par une combinaison convexe des critères $C_{\max}$ et $\overline{C}$ avec un poids égale à n pour le premier et égale à 1 pour le deuxième.

(GUPTA & al) [91] implantent plusieurs heuristiques de voisinage : recherche tabu, le recuit simulé et une recherche locale bi-niveau. Des résultats numériques montrent que l'algorithme génétique de (NEPPALLI & al) [150] est dominé en temps et en qualité par l'heuristique de recherche locale bi-niveau.

4.3.1.2 Le problème $F2|prmu|F_\ell(C_{\max}, \overline{C})$

• (NAGAR & al) [147] s'intéressent au problème où les critères $C_{\max}$ et $\overline{C}$ sont à minimiser par une combinaison convexe. Ce problème est fortement $\mathcal{NP}$-difficile puisque le problème lexicographique $F2|prmu|Lex(C_{\max}, \overline{C})$ est aussi fortement $\mathcal{NP}$-difficile. les trois auteurs lui proposent une heuristique gloutonne. Dans le cas particulier où les tâches possèdent les mêmes durées de traitement sur les deux machines, *i.e* $p_{i,1} = p_{i,2}$, $\forall i = 1, \ldots, n$, cette heuristique détermine un ordonnancement optimal.

Algorithme 2 : L'heuristique de (NAGAR & al, 1995)

Données : T étant l'ensemble des tâches à ordonnancer ;

début

Étape 1 : ***/* Initialisation de l'algorithme */***

$C_1 := 0$; $C_2 := 0$; $S = \emptyset$;

Étape 2 : ***/* Phase de recherche gloutonne */***

tant que $(T \neq \emptyset)$ **faire**

Trier T suivant les valeurs croissantes de $\max\{C_1 + p_{i,1}, C_2\} + p_{i,2}$;

$C_1 := C_1 + p_{T[1],1}$; $C_2 := \max\{C_1, C_2\} + p_{T[1],2}$;

$S := S//\{T[1]\}$; $T := T - \{T[1]\}$;

fin

Résultat : S étant l'ordonnancement déterminé par cet algorithme possédant les critères de performance $C_{\max}(S)$ et $\overline{C}(S)$.

Exemple illustratif

Nous considérons un problème avec $n = 10$ tâches :

J_i	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
p_i	5	6	7	10	10	8	13	7	10	2
d_i	10	8	11	10	9	7	5	4	2	1

☞ $C_1 = C_2 = 0$ et $S = \emptyset$.

☞ $T = \{J_{10}, J_8, J_9, J_2, J_1, J_6, J_3, J_7, J_5, J_4\}$,

$S = \{J_{10}\}$, $C_1 = 2$, $C_2 = 3$.

☞ $T = \{J_8, J_9, J_2, J_1, J_6, J_3, J_7, J_5, J_4\}$,

$S = \{J_{10}, J_8\}$, $C_1 = 9$, $C_2 = 13$.

☞ $T = \{J_9, J_2, J_1, J_6, J_3, J_7, J_5, J_4\}$,

$S = \{J_{10}, J_8, J_9\}$, $C_1 = 19$, $C_2 = 21$.

☞ $T = \{J_2, J_1, J_6, J_3, J_7, J_5, J_4\}$,

$S = \{J_{10}, J_8, J_9, J_2\}$, $C_1 = 25$, $C_2 = 33$.

☞ $T = \{J_6, J_1, J_7, J_3, J_5, J_4\}$,

$S = \{J_{10}, J_8, J_9, J_2, J_6\}$, $C_1 = 33$, $C_2 = 40$.

☞ $T = \{J_1, J_3, J_7, J_5, J_4\}$,

$S = \{J_{10}, J_8, J_9, J_2, J_6, J_1\}$, $C_1 = 38$, $C_2 = 50$.

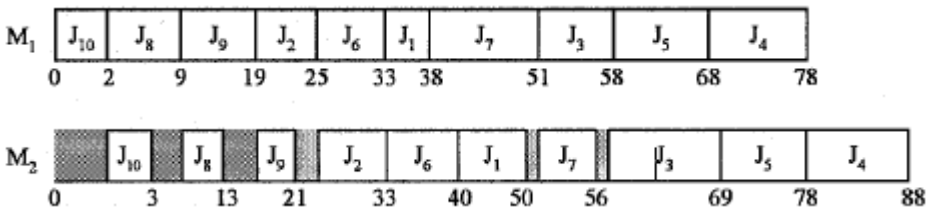
☞ $T = \{J_7, J_5, J_4, J_3\}$,

$S = \{J_{10}, J_8, J_9, J_2, J_6, J_1, J_7\}$, $C_1 = 51$, $C_2 = 56$.

☞ $T = \{J_3, J_5, J_4\}$,

$S = \{J_{10}, J_8, J_9, J_2, J_6, J_1, J_7, J_3\}$, $C_1 = 58$, $C_2 = 69$.

☞ $T = \{J_5, J_4\}$. Nous obtenons l'ordonnancement S illustré par la FIG.4.3



$C_{\max}(S) = 88$ et $\overline{C}(S) = 451$

FIG. 4.3 – L'ordonnancement déterminé par l'heuristique de (NAGAR & al, 1995).

Un algorithme branch-and-bound est aussi proposé par les auteurs. À chaque noeud, une tâche est ordonnancée après les tâches déjà ordonnancées. Une borne inférieure est calculée en utilisant une combinaison linéaire d'une borne inférieure du critère $C_{\max}$ et une borne inférieure du critère $\overline{C}$.

• (SIVRIKAYA-SERIFOGLU & ULUSOY) [176] s'intéressent aussi à ce problème pour lequel lui proposent une heuristique. En outre, les deux auteurs proposent trois algorithmes branch-and-bound pour sa résolution.

• YEH [207] s'intéresse à ce problème pour lequel lui propose une amélioration de l'heuristique élaborée par (NAGAR & al) [147] et un algorithme branch-and-bound. Des résultats numériques montrent que ce dernier est limité à des problèmes avec au plus 14 tâches.

4.3.1.3 Le problème $F2|prmu, r_i|F_\ell(C_{\max}, \overline{C})$

(CHOU & LEE) [38] considèrent que les tâches ont des dates de disponibilités et présentent le problème sous forme d'un programme linéaire à variables mixtes. Les auteurs proposent aussi une heuristique similaire à la procédure de recherche par faisceau (filtered beam search). Des résultats numériques sur des petites instances montrent sont efficacité.

4.3.1.4 Le problème $F2|prmu|\epsilon(\overline{C}/C_{\max})$

(SAYIN & KARABATI) [167] s'intéressent à la détermination de l'ensemble des optima de Pareto stricts des critères $C_{\max}$ et $\overline{C}$ en utilisant l'approche ϵ-contrainte. Ce problème est $\mathcal{NP}$-difficile au sens fort. SAYIN et KARABATI proposent deux algorithmes branch-and-bound qui sont utilisés itérativement.

Deux résultats numériques sont présentés par SAYIN et KARABATI. D'abord, les durées d'exécution sont générées entre 1 et 10 suivant une loi uniforme. Les résultats obtenus montrent que l'algorithme peut résoudre n'importe quel problème avec au plus 22 tâches. Le nombre moyen de vecteurs critères non dominés est entre 1.5 et 2.1. Le second résultat, concerne les problèmes pour lesquels les durées d'exécution sont générées entre 1 et 100. Ces problèmes sont plus difficiles à résoudre car l'algorithme ne peut pas résoudre quelques problèmes comportant 20 tâches.

4.3.1.5 Le problème $F2|prmu|\#(C_{\max}, T_{\max})$

(DANIELS & CHAMBERS) [44] s'intéressent à la détermination de l'ensemble des optima de Pareto stricts. Ce problème est $\mathcal{NP}$-difficile.

DANIELS et CHAMBERS proposent un algorithme branch-and-bound déterminant l'ensemble E. Les deux auteurs présentent aussi des règles d'élimination des nœuds dans l'arbre de recherche, ainsi que des conditions de dominance. Les deux auteurs proposent aussi une heuristique qui approxime l'ensemble E. Cette heuristique est inspirée de l'algorithme de (VAN WASSENHOVE & GELDERS) [202] qui résout le problème $1||\epsilon(\overline{C}/L_{\max})$. Cette heuristique se compose de deux parties. Dans la première partie, la valeur de ϵ est variée sur un certain intervalle. Pour chaque valeur, la deuxième partie détermine un ordonnancement qui minimise le critère $C_{\max}$ sous la contrainte $T_{\max} \leq \epsilon$. Des résultats numériques concernant le nombre d'optima de Pareto obtenus, sont présentés.

4.3.1.6 Le problème $F2|prmu|\#(C_{\max}, \overline{U})$

(LIAO & al) [132] s'intéressent à la détermination de l'ensemble des optima de Pareto stricts des critères $C_{\max}$ et $\overline{U}$. Les auteurs n'ont pas montré la complexité de ce problème mais connaissant la complexité du problème $F2|prmu|L_{\max}$, le problème bicritère est fortement $\mathcal{NP}$-difficile.

LIAO, YU et JOE proposent un algorithme branch-and-bound pour sa résolution. Des résultats numériques montrent que cet algorithme permet de traiter des problèmes allant jusqu'aux 30 tâches. En outre, le cas où les dates échues d_i sont les plus inférieures, les problèmes sont les plus difficiles à résoudre. Enfin, le nombre moyen de vecteurs critères non dominés est entre 1.1 et 1.8.

4.3.1.7 Le problème $F2|prmu|\#(C_{\max}, \overline{T})$

Lorsque les critères $C_{\max}$ et $\overline{T}$ sont pris en compte, la détermination de l'ensemble des optima de Pareto est proposée par (LIAO & al) [132]. Ce problème peut être démontré fortement $\mathcal{NP}$-difficile.

LIAO, YU et JOE proposent un algorithme branch-and-bound pour sa résolution. Des résultats numériques montrent que cet algorithme permet de traiter des problèmes allant jusqu'aux 30 tâches. En outre, le cas où les dates échues d_i sont les plus inférieures, les problèmes sont les plus difficiles à résoudre. Enfin, le nombre moyen de vecteurs critères non dominés est entre 1.2 et 3.1.

4.3.1.8 Le problème $F2|prmu|F_\ell(C_{\max}, \overline{C}, \overline{T})$

(EREN & GÜNER) [60] s'intéressent au problème d'ordonnancement tricritère flowshop à deux machines. L'objectif est de minimiser une combinaison convexe des critères $C_{\max}$, $\overline{C}$ et $\overline{T}$. Les deux auteurs modélisent ce problème, étant $\mathcal{NP}$-difficile, sous forme d'un programme linéaire en nombre entiers. L'algorithme modifié de (NAWAZ & al) [148], une heuristique basée sur la recherche tabu, la recherche aléatoire et la règle EDD, sont utilisés pour résoudre des problèmes avec au plus 2500 tâches. Des expériences numériques sont faites pour évaluer la performance des heuristiques. Des résultats montrent que les heuristiques sont assez efficaces, et la performance de l'heuristique basée sur la recherche tabu est meilleure par rapport aux autres en terme de la qualité de la solution.

4.3.2 Problèmes Flowshop à m machines

Dans cette section, nous considérons les problèmes du flowshop où m machines sont disponibles pour exécuter des tâches. Les machines sont utilisées suivant leurs indices, *i.e* M_1 en suite M_2, ..., jusqu'à M_m.

4.3.2.1 Le problème $F|prmu|Lex(C_{\max}, \overline{C})$

(SELEN & HOT) [171] et WILSON [204] résolvent ce problème en utilisant la programmation en variables mixtes. Puisque le cas $m = 2$ est $\mathcal{NP}$-difficile alors ce problème est pareil. Le modèle requière : $(2mn + n - m + 2)$ contraintes, n^2 variables en $0 - 1$ et $(mn + 1)$ variables entières. D'abord, le problème mathématique minimisant le makespan est résolu afin

de déterminer la valeur optimale de ce dernier. Ensuite, les auteurs résolvent le problème bicritère.

4.3.2.2 Le problème $F|prmu|\#(C_{\max}, \overline{C})$

• (GANGADHARAN & RAJENDRAN) [74] s'intéressent à la minimisation des critères $C_{\max}$ et $\overline{C}$. Ils restreignes leur étude à l'ensemble des permutations des ordonnancements. Aucune fonction objectif n'est définie explicitement et le problème est de trouver une solution appartenant à l'ensemble O défini par :

$$O = \{S/\ \forall S' \neq S,\ \frac{C_{\max}(S) - C_{\max}(S')}{\min(C_{\max}(S); C_{\max}(S'))} + \frac{\overline{C}(S) - \overline{C}(S')}{\min(\overline{C}(S); \overline{C}(S'))} \leq 0\}$$

L'heuristique proposée est basée sur la méthode du recuit simulé qui est exécutée avec deux séquences initiales. La meilleure solution calculée est retenue.

• RAJENDRAN [157] et RAJENDRAN [158] propose une heuristique qui calcule une solution appartenant à l'ensemble des séquences défini par :

$$O = \{S/\ \forall S' \neq S,\ \frac{C_{\max}(S) - C_{\max}(S')}{\min(C_{\max}(S); C_{\max}(S'))} + \frac{\overline{C}(S) - \overline{C}(S')}{\min(\overline{C}(S); \overline{C}(S'))} < 0\}$$

Cette heuristique est basée sur une méthode de voisinage ayant les mêmes démarche que celle présentées par RAJENDRAN [156] pour résoudre le problème $F2|prmu|Lex(C_{\max}, \overline{C})$.

RAJENDRAN généralise l'heuristique au problème avec trois critères $F|prmu|C_{\max}, \overline{C}, \overline{I}$, où $\overline{I}$ est la somme des temps d'arrêt sur toutes les machines, *i.e* $\overline{I} = \sum_{k=1}^{m} I_k$ avec I_k la somme des temps d'arrêt sur la machine M_k. L'heuristique élaborée est la même que celle citée précédemment sauf que l'ensemble de séquences O devient :

$$O = \{S/\ \forall S' \neq S,\ \frac{C_{\max}(S) - C_{\max}(S')}{\min(C_{\max}(S); C_{\max}(S'))} + \frac{\overline{C}(S) - \overline{C}(S')}{\min(\overline{C}(S); \overline{C}(S'))} + \frac{\overline{I}(S) - \overline{I}(S')}{\min(\overline{I}(S); \overline{I}(S'))} < 0\}$$

En faisant des comparaisons avec l'heuristique de (HO & CHANG) [100], qui résout le problème $F|prmu|C_{\max}$, les trois heuristiques donnent de meilleurs résultats pour les deux ou les trois critères.

4.3.2.3 Le problème $F|[\underline{p}_{i,j};\overline{p}_{i,j}]prmu|F_\ell(C_{\max},\overline{CC}_w)$

NOWICKI [151] s'intéresse au problème où le temps de traitement des tâches est à déterminer. Les coûts des durées de d'appréciation sont mesurées par le critère défini par :

$$\overline{CC}_w = \sum_{i=1}^{n}\sum_{j=1}^{m} w_{i,j}x_{i,j}$$

où $x_{i,j}$ représente la compression de l'opération $O_{i,j}$ définie par $p_{i,j} = \overline{p}_{i,j} - x_{i,j}$. Ce problème est $\mathcal{NP}$-difficile.

Pour le problème particulier à deux machines, NOWICKI propose un algorithme d'approximation avec une performance de garantie de $\frac{4}{3}$. Il est similaire à celui présenté par (NOWICKI & ZDRZALKA) [152]. En considérant un vecteur de compression initial x^0 tel que, $\forall i = 1,\ldots,n,\ \forall j = 1,\ldots,m,\ x^0_{i,j} \in [0;\underline{p}_{i,j} - \overline{p}_{i,j}]$. les Les durées opératoires étant fixées, l'algorithme de JOHNSON donne une séquence des tâches initiales. La seconde étape de cet algorithme considère que cette séquence étant fixée et cherche un vecteur x qui minimise la fonction objectif $F_\ell(C_{\max},\overline{CC}_w)$ en faisant la résolution d'un programme linéaire.

4.3.2.4 Le problème $F|p_{i,j} = p_i \in [\underline{p}_i;\overline{p}_i], prmu|\#(C_{\max},\overline{CC}_w)$

(CHENG & SHAKHLEVICH) [36] s'intéressent au problème flowshop particulier où toutes les opérations d'une tâche ont la même durée de traitement, *i.e* $p_{i,j} = p_i,\ \forall i = 1,\ldots,n$. En outre, les durées de traitement sont des variables à déterminer, et nous avons $p_i \in [\underline{p}_i;\overline{p}_i]$, $\forall i = 1,\ldots,n$. Les coûts des durées d'appréciation sont mesurées par le critère $\overline{CC}_w$ défini par : $\overline{CC}_w = \sum_{i=1}^{n} w_i x_i$ où $x_i \in [0;\underline{p}_i - \overline{p}_i]$ et il représente la compression de la tâche J_i, *i.e* $p_i = \overline{p}_i - x_i$.

(CHENG & SHAKHLEVICH) s'intéressent à la détermination des optima de Pareto stricts, qui est un problème polynomialement résolu.

(CHENG & SHAKHLEVICH) proposent un algorithme à posteriori basé sur l'énumération

des points extrêmes de Pareto du polyèdre des solutions dans l'espace des critères. Le problème peut être modélisé par un programme linéaire.

4.3.3 Problèmes Jobshop et Openshop

4.3.3.1 Les problèmes Jobshop

Peu de problèmes jobshop multicritère sont adressés dans la littérature.

• (HUCKERT & al) [106] étudient le problème $J||F_T(C_{\max}, \overline{C}, \overline{I}, T_{\max}, \overline{U})$ qui est fortement $\mathcal{NP}$-difficile. Ils proposent un algorithme intéractif inspiré de la méthode de STEM [15].

• (DECKRO & al) [47] étudient le problème $J||GP(C_{\max}, \overline{C}, \overline{E} + \overline{T})$ et ils proposent un programme à variables mixtes pour le résoudre.

4.3.3.2 Le problème $O_2||Lex(C_{\max}, \overline{C})$

• (GUPTA & WERNER) [94] montrent que si la valeur optimale du critère $C_{\max}$ est égale à $\max_{i=1,\ldots,n}(p_{i,1} + p_{i,2})$, alors un ordonnancement optimal pour le problème bicritère peut être obtenu en temps polynomial.

• (KYPARISIS & KOULAMAS) [126] étudient des cas particuliers pour lesquels le problème lexicographique est résolu en temps polynomial. Chaque cas est représenté sous forme d'un lemme [126].

(KYPARISIS & KOULAMAS) proposent plusieurs heuristiques pour le problème bicritère en des différentes configurations. Lorsque $\min_{i=1,\ldots,n} p_{i,1} > \min_{i=1,\ldots,n} p_{i,2}$, *i.e la machine* M_1 *domine la machine* M_2, un algorithme en $O(nlogn)$ avec une performance de garantie au pire des cas égale à $(1 + \frac{1}{n})$, est présenté.

4.3.3.3 Le problème $O_3||Lex(C_{\max}, \overline{C})$

(KYPARISIS & KOULAMAS) [126] étudient le problème $\mathcal{NP}$-difficile à trois machines. Ils proposent une heuristique en $O(nlogn)$ pour le cas où $\min_{i=1,\ldots,n} p_{i,1} > \max_{i=1,\ldots,n}(p_{i,2}, p_{i,3})$, *i.e la machine* M_1 *domine les machines* M_2 *et* M_3. Cette heuristique est similaire à celle présentée pour le problème à deux machines. Sa performance de garantie au pire des cas est égale à $1 + \frac{2(n+2)}{n(n+1)}$.

4.3.4 Problèmes d'ateliers avec affectation de ressources

4.3.4.1 Problème Flowshop hybride à trois étages

(FORTEMPS & al) [67] s'intéressent au problème d'ordonnancement apparaissant en chimie industrielle, noté par $HF3,(P_6,P_3,1)|constr|F_\ell(C_{\max},\gamma(T_i),\delta(VPI))$, où *constr* présente un ensemble de contraintes. L'objectif est de déterminer un ordonnancement qui minimise une combinaison convexe des trois critères, dont :

1. $C_{\max}$ est le makespan.
2. $\gamma(T_1,\ldots,T_n)$ est la fonction pénalité des retards des tâches.
3. $\delta(VPI)$ est la fonction pénalité qui exprime la violation des période d'indisponibilité des machines sur le premier étage.

Puisque le problème est $\mathcal{NP}$-difficile, l'algorithme proposé pour sa résolution est une heuristique en deux phases. La première étape détermine une séquence de tâches $\mathcal{S}$, qui présente l'ordre dont lequel ces dernières sont introduites dans l'atelier. La deuxième étape détermine l'ordonnancement final et affecte les tâches aux ressources en chaque étage.

Deux heuristiques sont proposées pour la détermination de la séquence $\mathcal{S}$. La première est l'algorithme du recuit simulé et la deuxième est celui de la recherche tabu. L'idée générale de l'heuristique d'affectation est d'ordonnancer les tâches tant est possible sur les machines suivant la règle FAM. Ces affectations sont faites en respectant les contraintes imposées sur chaque étage.

4.3.4.2 Problèmes Flowshop hybride à k étages

4.3.4.2.1 Le problème $HFk,(PM^{(\ell)})_{\ell=1}^{k}||F_\ell(C_{\max},\overline{C})$

(RIANE & al) [160] s'intéressent au problème d'ordonnancement où l'atelier est composé de k étages et chaque étage ℓ possède $M^{(\ell)}$ machines identiques. L'objectif est de minimiser le makespan et la somme des dates de fin de traitement des tâches. Pour ce faire, RIANE, MESKENS et ARTIBA minimisent une combinaison convexe des critères. Ce problème est $\mathcal{NP}$-difficile.

Les trois auteurs modélisent le problème sous forme d'un programme linéaire à variables mixtes. En outre, deux heuristiques de recherche tabu sont proposées. La première génère aléatoirement une séquence de tâches. L'affectation sur le premier étage est faite suivant la

règle FAM et les tâches sont ordonnancées tant est possible. En supposant que les affectations sur les étages suivants sont faites suivant la même règle et en considérant la séquence de tâches obtenue en un ordre non décroissant des dates de fin de traitement des tâches sur l'étage précédant. Le voisinage considéré est obtenu en permettant n'importe quelle paire de tâches. Dans la deuxième heuristique, l'ordonnancement est déterminé suivant les mêmes étapes. Seulement, la séquence initiale utilisée sur le premier étage est générée en utilisant l'heuristique de [27].

4.3.4.2.2 Le problème $HFk, (PM^{(\ell)})_{\ell=1}^{k}||\epsilon(\overline{C}/C_{\max})$

(RIANE & al) [160] s'intéressent au problème d'ordonnancement où l'atelier est composé de k étages et chaque étage ℓ possède $M^{(\ell)}$ machines identiques. L'objectif est de minimiser le makespan et la somme des dates de fin de traitement des tâches, en considérant la minimisation du critère $\overline{C}$ sous la contrainte $C_{\max} \leq \epsilon$.

Les auteurs modélisent le problème sous forme d'un programme linéaire à variables mixtes. En outre, trois heuristiques de recherche tabu sont présentées. Le principe des deux premières est identique à celles proposées pour la résolution du problème $HFk, (PM^{(\ell)})_{\ell=1}^{k}||F_{\ell}(C_{\max}, \overline{C})$. Le troisième algorithme de recherche tabu améliore une solution initiale déterminée en résolvant le problème $Lex(C_{\max}, \overline{C})$.

4.3.4.2.3 Le problème $HFk, (PM^{(\ell)})_{\ell=1}^{k}|r_i^{(1)}, d_i^{(k)}|\epsilon(C_{\max}/T_{\max})$

(VIGNIER & al) [198] s'intéressent au problème où les contraintes d'indisponibilité sont imposées sur les ressources. L'atelier est composé de k étages, chaque machine possède sa propre période où elle est non disponible à travailler. Le nombre de machines disponibles à l'instant t et à l'étage ℓ est noté par $M^{(\ell)}(t)$, et en supposant que les machines ont un profile « *zig-zag* », *i.e* $M^{(\ell)}(t) \in [M^{(\ell)} - 1; M^{(\ell)}]$, $\forall t$ (voir [165]). Chaque tâche J_i est définie par une date de disponibilité sur le premier étage et une date échue sur le dernier étage. La préemption des tâches est interdite sauf lorsqu'une tâche ne peut être complétée avant le début de la période d'indisponibilité de la machine sur laquelle la tâche est ordonnancée. L'objectif est de minimiser le makespan sous la contrainte $T_{\max} \leq \epsilon$. Ce problème est équivalent au problème $HFk, (PM^{(\ell)})_{\ell=1}^{k}|r_i^{(1)}, d_i^{(k)}|C_{\max}$, où $\widetilde{d}_i^{(k)} = d_i^{(k)} + \epsilon$, $\forall i = 1, \ldots, n$

Pour résoudre ce problème, un algorithme de branch-and-bound, inspiré de celui proposé par (BRAH & HUNSUCKER) [24], est présenté.

CHAPITRE 5

Une Solution Faiblement Efficace pour le Problème $P||Lex(C_{\max}, L_{\max})$

Dans ce chapitre, nous nous sommes intéressés à la minimisation du makespan $C_{\max}$ et le décalage temporel maximal $L_{\max}$ dans le problème $P||Lex(C_{\max}, L_{\max})$. Nous présentons l'extension d'un algorithme optimal basé sur la méthode de branch–and–bound développé par BOUDHAR [21] déterminant la valeur optimale du critère $C_{\max}$ dans le problème $P||C_{\max}$. À une certaine étape de la procédure de séparation, nous rajoutons la procédure optimisant un deuxième critère $L_{\max}$ par l'application de la règle de tri EDD aux sous problèmes $1||L^j_{\max}$, $j = 1, \ldots, m$. Un exemple d'application est illustré à la fin de ce chapitre.

5.1 Le problème $P||Lex(C_{\max}, L_{\max})$

Nous abordons le problème d'ordonnancement de n tâches indépendantes sur m machines parallèles identiques, lorsque aucune préemption n'est autorisée. Le but est de minimiser le makespan $C_{\max}$ et le décalage temporel maximal $L_{\max}$. Supposons que le décideur ne permet pas l'agrégation des deux critères et ces derniers sont classés selon leurs importances $C_{\max} \longrightarrow L_{\max}$. Tenons compte de cet ordre lexicographique, nous optimisons ces critères l'un après l'autre.

Théorème 5.1.1.
Le problème $P||Lex(C_{\max}, L_{\max})$ est $\mathcal{NP}$-difficile.

Preuve.
Ce problème est $\mathcal{NP}$-difficile puisque même le problème $P||C_{\max}$ est pareil [1]. □

5.2 Une Solution Faiblement Efficace pour le Problème $P||Lex(C_{\max}, L_{\max})$

Dans ce qui suit, nous allons décrire un algorithme exact (évidemment non polynomial) basé sur la méthode de séparation et évaluation (branch-and-bound). La procédure de séparation partitionne le problème en plusieurs sous problèmes identiques et indépendants de taille inférieure ou égale. La procédure d'évaluation calcule une borne inférieure pour la valeur optimale de chaque sous problème, ce qui permet de faire des coupes dans l'arborescence. La procédure d'évaluation nous permet de construire un ordonnancement de longueur inférieure ou égale à g avec :

$$g = \max\{\max_{j=1,\ldots,m}\{H_j\},\ \max_{i=1,\ldots,n}\{p_i\},\ \lceil\frac{1}{m}\sum_{i=1}^{n} p_i\rceil\};$$

où H_j est la date de fin de traitement des tâches déjà affectées à la machine M_j.

Pour cela, nous supposons qu'une liste de priorité est donnée pour les tâches non encore affectées (nous pouvons choisir la règle LPT) et à chaque étape, la première machine libre est sélectionnée pour traiter la première tâche libre de la liste telle que le temps de traitement global sur la machine considérée soit inférieur ou égale à g.

Si une tâche J_i ne peut pas être affectée à une machine M_j de sorte que $H_j + p_i \leq g$ alors la tâche J_i est fractionnée sur plusieurs machines et la solution obtenue n'est pas réalisable. L'ordonnancement est donc de longueur égale à g.

[1] Le problème $P||C_{\max}$ est $\mathcal{NP}$-difficile puisque même le problème restraint à deux machines $P2||C_{\max}$ a été démontré comme étant $\mathcal{NP}$-difficile [118].

Étant sur un nœud de l'arborescence. Soit C la meilleure évaluation exacte (associée à une solution réalisable) trouvée. Avec g l'évaluation calculée sur le nœud considéré.

Si $g \geq C$, il n'est pas nécessaire d'explorer plus à fond ce nœud de l'arborescence, puisqu'il ne peut y avoir une valeur optimale de coût inférieur à $C - 1$ à partir de ce nœud. Sinon, (c'est-à-dire $g < C$), nous vérifions si la solution obtenue associée à l'évaluation g est réalisable, si c'est le cas nous affectons à C la valeur de g et nous appliquons la règle EDD sur chaque machine afin de minimiser les $L^j_{\max}$ correspondant aux problèmes $1||L^j_{\max}$, $j = 1, \ldots, m$. Nous considérons que $L^j_{\max}$ est le maximum des décalages temporels des tâches à traiter sur la $j^{\text{ème}}$ machine, $j = 1, \ldots, m$, alors le décalage temporel maximal $L_{\max}$ correspondant au problème $P||Lex(C_{\max}, L_{\max})$ est

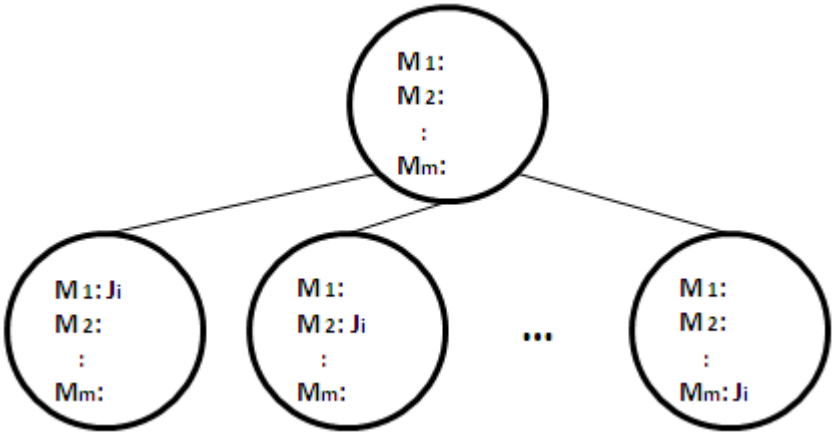
$$L_{\max} = \max(L^j_{\max})_{j=1,\ldots,m};$$

Sinon nous séparons comme suit :

Soit J_i la tâche qui a causé l'irréalisabilité de la solution, à partir du nœud considéré, nous allons créer m nœuds fils tels que :

- La tâche J_i est affectée à la machine M_1 pour le premier nœud.
- La tâche J_i est affectée à la machine M_2 pour le deuxième nœud.

  ⋮ ⋮ ⋮
- La tâche J_i est affectée à la machine M_m pour le $m^{\text{ème}}$ nœud.



Et nous continuons à avancer dans l'arborescence en utilisant une exploration par profondeur d'abord (par backtracking).

5.3 Les différentes procédures de l'algorithme par séparation et évaluation

Les différentes procédures de l'algorithme par séparation et évaluation sont :

Procédure Initialisation;

début

$M_1 := \emptyset\,;\ M_2 := \emptyset\,;\ \ldots\ ;\ M_m := \emptyset$;

$JC := J$;

$C := +\infty$;

fin

Procédure Évaluation($M_1\,;\ \ldots\ ;\ M_m\,;\ JC\,;\ g\,;\ S\,;\ \textit{Réalisable}$);

début

$g = \max\{\max\limits_{j=1,\ldots,m}\{H_j\},\ \max\limits_{i=1,\ldots,n}\{p_i\},\ \lceil\frac{1}{m}\sum_{i=1}^{n} p_i\rceil\}$;

- Affecter les tâches de JC aux m machines suivant la règle LPT de sorte que $H_j \leq g$, $\forall j = 1, \ldots, m$;

si *Une tâches J_i ne peut pas être affectée entièrement à une machine (*i.e *la tâche est morcelée)* **alors**

$S := J_i$;

Réalisable:= $Faux$

sinon

Réalisable:= $Vrai$

fin

Procédure EDD(M_1 ; ... ; M_m ; L);
début
  $j := 1$;
  tant que $(j \leq m)$ **faire**
    - Appliquer la règle EDD sur les tâches affectées à la machine M_j ;
    - Calculer L^j ;
    - j :=j+1 ;
  $L := \max(L^j)_{j=1,...,m}$
fin

Procédure SE(M_1 ; ... ; M_m ; JC ; C);
début
  Évaluation(M_1 ; ... ; M_m ; JC ; g ; S ; *Réalisable*);
  si $g < C$ **alors**
    si *Réalisable* **alors**
      $C := g$;
      $\overline{M}_1 := M_1$; ... ; $\overline{M}_m := M_m$;
      EDD($\overline{M}_1$; ... ; $\overline{M}_m$; L) ;
    sinon `/* séparation */`
      $JC := JC \setminus S$;
      $j := 1$;
      tant que $(j \leq m) \wedge (g < C)$ **faire**
        $M_j := M_j \cup \{S\}$;
        SE(M_1 ; ... ; M_m ; JC ; C);
        $M_j := M_j \setminus \{S\}$;
        $j := j + 1$
fin

L'algorithme se résume ainsi :

Algorithme 3 : Le branch-and-bound pour le problème $P||Lex(C_{\max}, L_{\max})$

Données : J étant l'ensemble des tâches à ordonnancer ;

JC étant l'ensemble des tâches non encore affectées aux machines ;

n étant le nombre de tâches ;

m étant le nombre de Machines identiques en parallèle ;

H_j étant la date de fin de traitement des tâches déjà affectées à la machine M_j.

Résultat : $(C, L)^T$ étant la solution faiblement efficace déterminée par cet algorithme.

début

Initialisation ;

Évaluation(M_1 ; ... ; M_m ; JC ; g ; S ; *Réalisable*) ;

SE(M_1 ; ... ; M_m ; JC ; C)

fin

5.4 Résultats théoriques

Théorème 5.4.1. *[21]*

L'algorithme 3 se termine en un nombre fini d'étapes.

Preuve.

Il n'est pas difficile de voir que l'algorithme 3 est fini puisqu'il n'y a pas de fausses séparations.

□

Théorème 5.4.2.

La solution donnée par l'algorithme 3 est faiblement efficace.

Preuve.

L'algorithme 3 détermine d'abord la valeur optimale du critère $C_{\max}$ en considérant le problème $P||C_{\max}$ [21], en suite, la valeur optimale du critère $L_{\max} = \max(L^j_{\max})_{j=1,...,m}$, où $L^j_{\max}$ étant la solution optimale du sous problème $1||L^j_{\max}$ en appliquant la règle EDD aux tâches affectées à la machine M_j. Puisque la valeur optimale du problème $P||C_{\max}$ peut correspondre à plusieurs séquences, alors, la solution trouvée par l'algorithme 3 $(C_{\max}, L_{\max})^T$ est faiblement efficace.

□

5.5 Exemple illustratif

Soit à ordonnancer, sans préemption, cinq tâches sur deux machines parallèles identiques. Le but est de minimiser $C_{\max}$ et $L_{\max}$ dans cet ordre (l'agrégation des critères n'est pas permise). Cela revient à proposer un ordonnancement Pareto optimal pour le problème $P2||Lex(C_{\max}, L_{\max})$ ayant les données suivantes :

J_i	J_1	J_2	J_3	J_4	J_5
p_i	10	8	7	4	3
d_i	15	13	10	12	5

Nous avons $\max\limits_{i=1,\ldots,n}\{p_i\} = 10$ et $\lceil \frac{1}{m}\sum_{i=1}^{n} p_i \rceil = 16$.

☞ Au premier nœud, nous avons :

- $H_1 = H_2 = 0$, $g = \max\{0, 10, 16\} = 16$ et $C = +\infty$;
- Les tâches J_1, J_2, J_3 et J_4 sont affectées comme suit :

  $M_1 = \{J_1, J_4\}$

  $M_2 = \{J_2, J_3\}$

  Alors que la tâche J_5 doit être partagée entre les deux machines pour avoir une date de fin de traitement égale à 16 et ainsi la solution n'est pas réalisable. C'est donc la tâche J_5 qui a causé l'irréalisabilité de la solution.

☞ Au deuxième nœud, nous avons :

- $H_1 = 3, H_2 = 0$, $g = \max\{3, 10, 16\} = 16$ et $C = +\infty$;
- Les tâches J_1 et J_2 sont affectées comme suit :

  $M_1 = \{J_5, J_2\}$

  $M_2 = \{J_1\}$

  Alors que la tâche J_3 doit être partagée entre les deux machines pour avoir une date de fin de traitement égale à 16 et ainsi la solution n'est pas réalisable. C'est donc la tâche J_3 qui a causé l'irréalisabilité de la solution.

☞ Au troisième nœud, nous avons :

- $H_1 = 10, H_2 = 0$, $g = \max\{10, 10, 16\} = 16$ et $C = +\infty$;
- La tâche J_1 est affectée comme suit :

$M_1 = \{J_5, J_3\}$

$M_2 = \{J_1\}$

Alors que la tâche J_2 doit être partagée entre les deux machines pour avoir une date de fin de traitement égale à 16 et ainsi la solution n'est pas réalisable. C'est donc la tâche J_2 qui a causé l'irréalisabilité de la solution.

☞ Au quatrième nœud, nous avons :

- $H_1 = 18, H_2 = 0$, $g = \max\{18, 10, 16\} = 18$ et $C = 18$;
- Les tâches J_1 et J_4 sont affectées comme suit :

$M_1 = \{J_5, J_3, J_2\}$

$M_2 = \{J_1, J_4\}$

• **Application de la règle EDD sur la machine M_1 :**

Nous obtenons la suite de tâches J_5, J_3, J_2 pour $L^1_{\max} = +5$:

J_i	J_5	J_3	J_2
p_i	3	7	8
d_i	5	10	13
C_i	3	10	18
L_i	-2	0	+5

• **Application de la règle EDD sur la machine M_2 :**

Nous obtenons la suite de tâches J_4, J_1 pour $L^2_{\max} = -1$:

J_i	J_4	J_1
p_i	4	10
d_i	12	15
C_i	4	14
L_i	-8	-1

Donc, $L_{\max} = \max(L^1_{\max}, L^2_{\max}) = \max(+5, -1) \Longrightarrow L_{\max} = +5$.

☞ Au cinquième nœud, nous avons :

- $H_1 = 10, H_2 = 8$,$g = \max\{10, 10, 16\} = 16$ et $C = 18$;

- $M_1 = \{J_5, J_3\}$

  $M_2 = \{J_2\}$

  Alors que la tâche J_1 doit être partagée entre les deux machines pour avoir une date de fin de traitement égale à 16 et ainsi la solution n'est pas réalisable. C'est donc la tâche J_1 qui a causé l'irréalisabilité de la solution.

☞ Au sixième nœud, nous avons :

- $H_1 = 20, H_2 = 8$, $g = \max\{20, 10, 16\} = 20$ et $C = 18$;
- Les tâches J_1 et J_2 sont affectées comme suit :

  $M_1 = \{J_5, J_3, J_1\}$

  $M_2 = \{J_2\}$

  C'est pas nécessaire d'explorer plus à fond ce nœud.

☞ Au septième nœud, nous avons :

- $H_1 = 10, H_2 = 18$, $g = \max\{18, 10, 16\} = 18$ et $C = 18$;
- La tâche J_1 est affectée comme suit :

  $M_1 = \{J_5, J_3\}$

  $M_2 = \{J_2, J_1\}$

  C'est pas nécessaire d'explorer plus à fond ce nœud.

☞ Au huitième nœud, nous avons :

- $H_1 = 3, H_2 = 7$, $g = \max\{7, 10, 16\} = 16$ et $C = 18$;
- Les tâches J_1 et J_2 sont affectées comme suit :

  $M_1 = \{J_5, J_1\}$

  $M_2 = \{J_3, J_2\}$

  Alors que la tâche J_4 doit être partagée entre les deux machines pour avoir une date de fin de traitement égale à 16 et ainsi la solution n'est pas réalisable. C'est donc la tâche J_4 qui a causé l'irréalisabilité de la solution.

☞ Au neuvième nœud, nous avons :

- $H_1 = 7, H_2 = 7$, $g = \max\{7, 10, 16\} = 16$ et $C = 18$;
-

$M_1 = \{J_5, J_4\}$

$M_2 = \{J_3\}$

Alors que la tâche J_1 doit être partagée entre les deux machines pour avoir une date de fin de traitement égale à 16 et ainsi la solution n'est pas réalisable. C'est donc la tâche J_1 qui a causé l'irréalisabilité de la solution.

☞ Au dixième nœud, nous avons :

- $H_1 = 17, H_2 = 7$, $g = \max\{17, 10, 16\} = 17$ et $C = 17$;
- La tâche J_2 est affectée comme suit :

$M_1 = \{J_5, J_4, J_1\}$

$M_2 = \{J_3, J_2\}$

• Application de la règle EDD sur la machine M_1 :

Nous obtenons la suite de tâches J_5, J_4, J_1 pour $L^1_{\max} = +2$:

J_i	J_5	J_4	J_1
p_i	3	4	10
d_i	5	12	15
C_i	3	7	17
L_i	-2	-5	+2

• Application de la règle EDD sur la machine M_2 :

Nous obtenons la suite de tâches J_3, J_2 pour $L^2_{\max} = +2$:

J_i	J_3	J_2
p_i	7	8
d_i	10	13
C_i	7	15
L_i	-3	+2

Donc, $L_{\max} = \max(L^1_{\max}, L^2_{\max}) = \max(+2, +2) \Longrightarrow L_{\max} = +2$.

Remarque 5.5.1. *Si nous continuons l'exploration de l'arborescence, nous ne améliorons pas la solution obtenue. (Voir* FIG.*5.2 et* FIG.*5.3)*

• **Un ordonnancement Pareto optimal :**

Enfin, nous déterminons un ordonnancement Pareto optimal ayant les critères de performance $(C_{\max}, L_{\max})^T = (17, 2)^T$

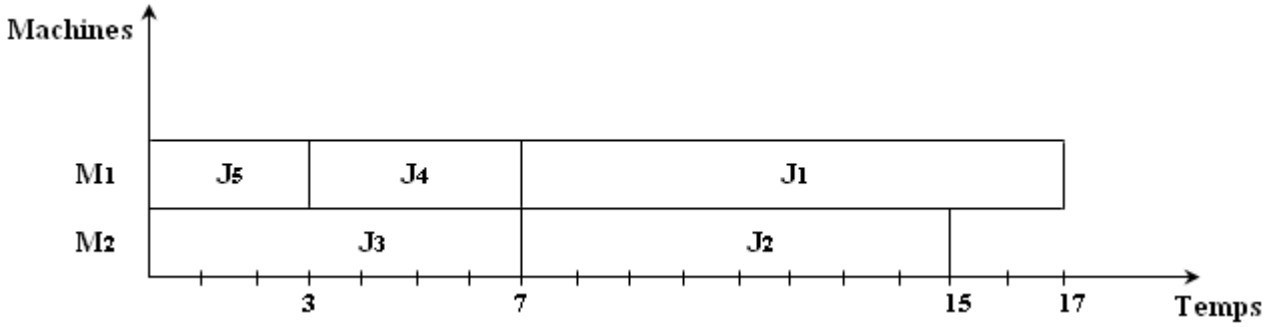


FIG. 5.1 – Un ordonnancement Pareto optimal pour l'exemple illustratif.

5.6 Conclusion

Nous avons tenté dans cette partie de faire une extension de l'algorithme optimal pour le problème $P||C_{\max}$ développé par BOUDHAR [21] et basé sur la méthode de branch-and-bound, en lui rajoutant une procédure optimisant le critère $L_{\max}$ par la règle de tri EDD afin de déterminer un ordonnancement Pareto optimal aux problème $P||Lex(C_{\max}, L_{\max})$.

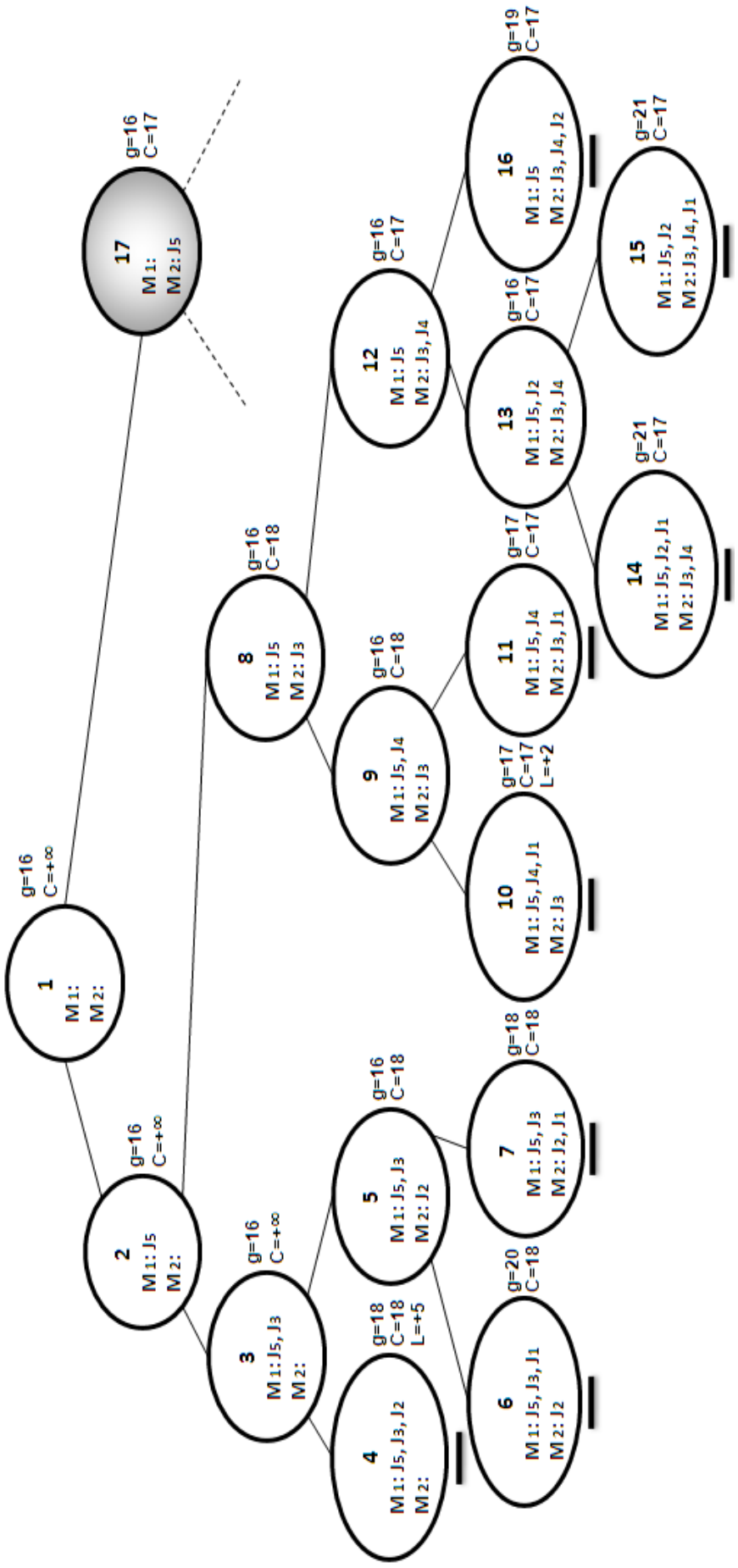


FIG. 5.2 – L'arborescence construite par l'algorithme 3 appliqué sur l'exemple illustratif (à suivre . . .).

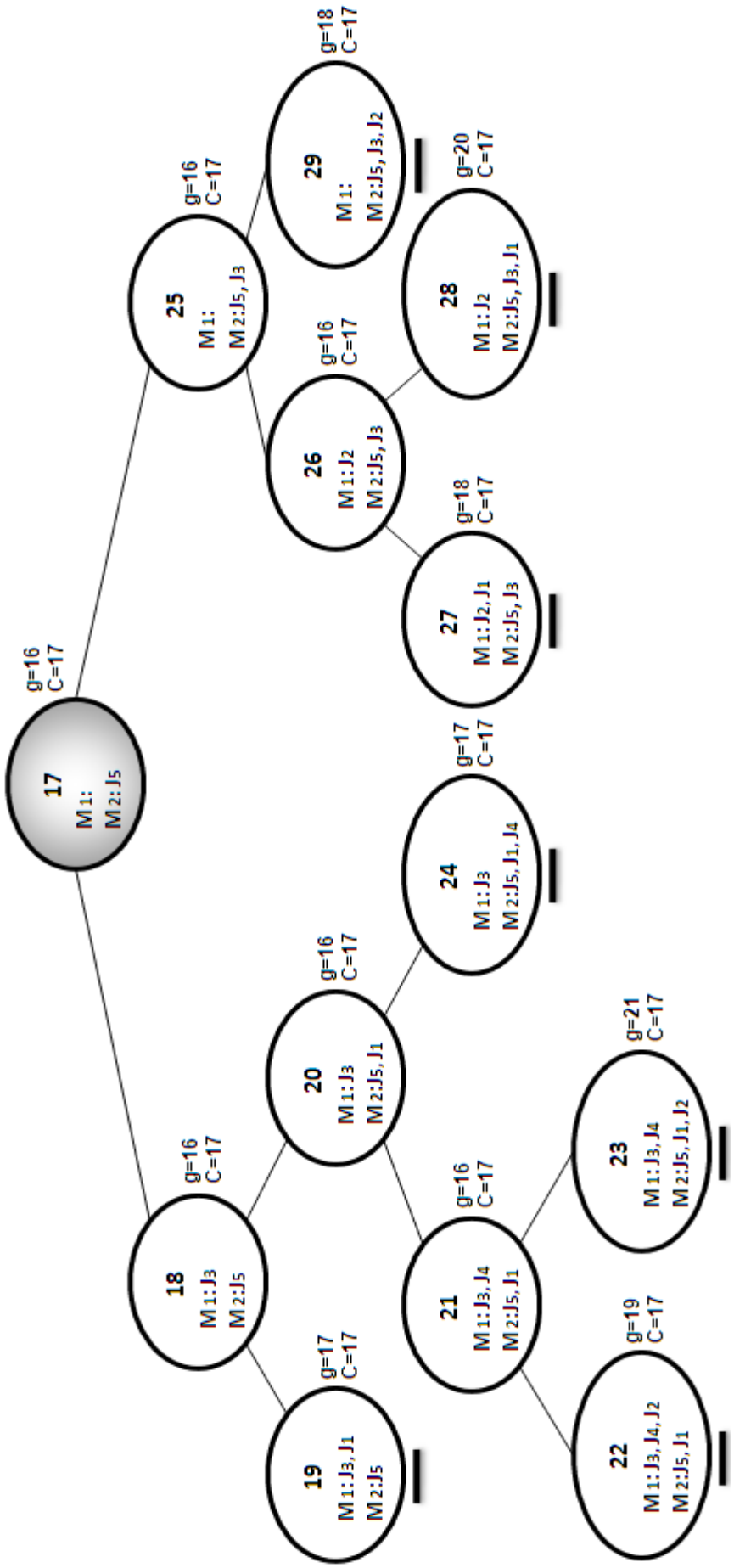


FIG. 5.3 – (... suite) L'arborescence construite par l'algorithme 3 appliqué sur l'exemple illustratif.

Conclusion

« L'ordonnancement concerne l'affectation de ressources limitées aux tâches dans le temps. C'est un processus de prise de décision dont le but est d'optimiser un ou plusieurs objectifs ».

L'objectif de ce mémoire a été la présentation d'un état de l'art sur quelques types de problèmes d'ordonnancement les plus importants et l'introduction à la nouvelle définition de ces problèmes. Après avoir présenté les résultats de base de l'optimisation multi-objectif. Nous avons exposé également la démarche, conforme aux principes de l'aide multicritère à la décision, pour aborder les problèmes d'ordonnancement multicritères. Ces problèmes sont décomposés en trois sous-problèmes. Le premier concerne la modélisation du problème d'ordonnancement considéré. La résolution du second sous-problème conduit à répondre à des questions : Comment prendre en compte les critères pour calculer des optima de Pareto ? Quel type d'algorithme faut-il mettre au point ? Le troisième sous-problème concerne la résolution du problème d'ordonnancement qui découle des deux sous-problèmes précédants. Nous avons présenté également dans ce mémoire l'extension de la notation classique des problèmes d'ordonnancement au cas multicritère. Nous avons cité en suite quelques stratégies d'approches de résolution avant de détailler notre état de l'art sur quelques problèmes d'ordonnancement multicritère à une seule machine, à machines parallèles et de type d'ateliers. Ce mémoire est clôturé par l'extension d'un algorithme optimal pour le problème $P||C_{\max}$ développé par BOUDHAR [21] et basé sur la méthode de branch-and-bound, en lui rajoutant une procédure optimisant le critère $L_{\max}$ par la règle de tri EDD afin de déterminer un ordonnancement Pareto optimal au problème $P||Lex(C_{\max}, L_{\max})$, où un ordre lexicographique est défini entre

les critères.

Dans ce travail, nous avons pu déterminer une solution faiblement efficace pour le problème $P||Lex(C_{\max}, L_{\max})$. Il serait intéressant, d'abord, de généraliser l'algorithme de branch-and-bound afin d'identifier d'autres solutions Pareto optimales, en suite, d'utiliser une métaheuristique pour la résolution du problème considéré. Il serait aussi intéressant de considérer le problème $P|pmtn|Lex(C_{\max}, L_{\max})$ et de faire en suite une étude comparative avec les travaux effectués par (MOHRI & al.) [143] concernant les deux problèmes $P2|pmtn|\epsilon(L_{\max}/C_{\max})$ et $P3|pmtn|\epsilon(L_{\max}/C_{\max})$.

On espère vivement que ce travail motive d'autres chercheurs pour développer de meilleurs méthodes de résolution dans ce domaine.

BIBLIOGRAPHIE

[1] M. S. AHMED and P. S. SUNDARARAGHAVAN. Minimising the Weighted Sum of Late and Early Completion Penalties in a Single Machine. *IIE Transactions*, 22(3) :288–290, 1990.

[2] B. ALIDAEE and A. AHMADIAN. Two Parallel Machine Sequencing Problems Involving Controllable Job Processing Times. *European Journal of Operational Research*, 70 :335–341, 1993.

[3] Y. P. ANEJA and K. P. K. NAIR. Bicriteria Transportation Problem. *Management Science*, 25 :73–78, 1979.

[4] M. AZIZOGLU, S. K. KONDAKCI, and O. KIRCA. Bicriteria Scheduling Problem Involving Total Earliness and Total Tardiness Penalties. *International Journal of Production Economics*, 23 :17–24, 1991.

[5] M. AZIZOGLU, S. K. KONDAKCI, and M. KÖKSALAN. *Bicriteria Scheduling : Minimising Flowtime and Maximum Earliness on a Single Machine, pages 279-288.* edited by J. Climaco, Multicriteria Analysis. Springer-Verlag, 1997.

[6] U. BAGCHI, R.S. SULLIVAN, and Y. L. CHANG. Minimising Mean Absolute Deviation of Completion Times about a Commun Due Date. *Naval Research Logistics Quarterly*, 33 :227–240, 1986.

[7] U. BAGCHI, R.S. SULLIVAN, and Y. L. CHANG. Minimising Absolute and Squared Deviations of Completion Times with Different Earliness and Tardiness Penalities and a Commun Due Date. *Naval Research Logistics Quarterly*, 34 :739–751, 1987.

[8] U. BAGCHI, R.S. SULLIVAN, and Y. L. CHANG. Minimising Mean Squared Deviation of Completion Times about a Commun Due Date. *Management Science*, 33(7) :894–906, 1987.

[9] K. R. BAKER. *Introduction to Sequencing and Scheduling.* John Wiley & sons, Inc, 1974.

[10] S. P. BANSAL. Single Machine Scheduling to Minimise Weighted Sum of Completion Times with Secondary Criterion - a Branch and Bound Approach. *European Journal of Operational Research*, 5(3) :177–181, 1980.

[11] V. BARICHARD. *Approches Hybrides pour les Problèmes Multiobjectifs.* Thèse de Doctorat, Université d'Angers, France, 2003.

[12] J. W. BARNES and L. K. VANSTON. Scheduling Jobs with Linear Delay Penalties and Sequence Dependent Setup Costs. *Operations Research*, 29(1) :146–160, 1981.

[13] A. BAYKASOGLU, S. OWEN, and N. GINDY. A Tabu Search Based Approach to Find the Pareto Optimal Set in Multiple Objective Optimization. *Engineering Optimization*, 31 :731–748, 1999.

[14] R. BELLMANN. *Dynamic Programming.* Princeton University Press, Princeton, 1957.

[15] R. BENAYOUN, J. DE MONGOLFIER, J. TERGNY, and O. LARITCHEV. Linear Programmig with Multiple Objective Functions : Step Method (STEM). *Mathematical Programming*, 1(3) :366–375, 1971.

[16] S. BERTEL and J. C. BILLAUT. A Genetic Algorithm for an Industrial Multiprocessor Flowshop Scheduling Problem with Reciculation. *European Journal of Operational Research*, 159 :651–662, 2004.

[17] J. C. BILLAUT. Recherche Opérationnelle et Aide à la Décision pour les Problèmes d'Ordonnancement. Habilitation à diriger des recherche, Laboratoire d'Informatique, E3i, Université François Rabelais, Tours, France, 1999.

[18] J. C. BILLAUT, V. T'KINDT, P. RICHARD, and C. PROUST. Three Exact Methods and an Efficient Heuristic for Solving a Bicriteria Flowshop Scheduling Problem. *In Multiconference on Computational Engineering in Systems Applications (CESA'98), IMACS/IEEE*, pages Pages 371–377, Nabeul–Hammamet, Tunisia, 1998.

[19] J. BLAZEWICZ, K. H. ECKER, E. PESCH, G. SCHMIDT, and J. WEGLARZ. *Scheduling in Computer and Manufacturing Processes.* Springer Verlag, 1996.

[20] J. BLAZEWICZ, G. FINKE, M. L. ESPINOUSE, and G. PAWLAK. *Scheduling Vehicles in a Cyclic Hybrid Flowshop*, volume 1, 611-619. Proceedings of International Conference on Industrial Ingineering and Production Management (IEPM'97), 1997.

[21] M. BOUDHAR. *Sur Quelques Problèmes d'Ordonnancement d'Atelier*. Thèse de Magister, Université des Science et de le Technologie Houari Boumediene, Alger, 1991.

[22] V. BOURGADE, L. M. AGUILERA, B. PENZ, and Z. BINDER. Problème Industriel d'Ordonnancement Bicritère sur Machines Unique : Modélisation et Aide à la Décision. *RAIRO-APII*, 29(3) :331–341, 1995.

[23] V. J. BOWMAN. *On the Relationship of the Tchebycheff Norm and the Efficient Frontier of Multiple-Criteria Objectives, pages 76-85. In [H. THIRIEZ and S. ZIONTS, "Multiple Criteria Decision Making", 1976]*. Springer, 1976.

[24] S. A. BRAH and J. L. HUNSUCKER. Branch and Bound Algorithm for the Flow Shop with Multiple Processors. *European Journal of Operational Research*, 51 :88–99, 1991.

[25] J. BRUNO, E. G. COFFMAN, and J. R. SETHI. Scheduling Indepedent Tasks to Reduce Mean Finish Time. *Comm. ACM*, 17 :382–387, 1974.

[26] R. N. BURNS. Scheduling to Minimise the Weighted Sum of Completion Times with Secondary Criteria. *Naval Research Logistics Quarterly*, 23(1) :125–129, 1976.

[27] H. G. CAMPBELL, R. A. DUDECK, and M. L. SMITH. A Heuristic Algorithm for the n-job, m-machine Sequencing Problem. *Management Science*, 16 :630–637, 1970.

[28] S. CHAND and H. SCHNEEBERGER. Single Machine Scheduling to Minimise Weighted Completion Time with Maximum Allowable Tardiness. Technical report, Technical Report, University of Purdue, U.S.A, 1984.

[29] S. CHAND and H. SCHNEEBERGER. A note on the Single Machine Scheduling Problem with Minimum Weighted Completion Time and Maximum Allowable Tardiness. *Naval Research Logistics Quarterly*, 33(3) :551–557, 1986.

[30] Y. CHANG, C. YEH, and C. SHEN. A Multiobjective Model for Passengers Train Services Planning : Application to Taiwan's High-speed Rail Line. *Transportation Research Part B : Policy and Practice*, 34 :91–106, 2000.

[31] A. CHARNES and W. W. COOPER. *Management Models and Industrial Applications of Linear Programming*, volume I and II. Edition John Wiley & Sons, New York, 1961.

[32] A. CHARNES, W. W. COOPER, and R. O. FERGUSON. Optimal Estimation of Executive Compensation by Linear Programming. *Management Science*, 1(2) :138–151, 1955.

[33] C. L. CHEN and R. L. BULFIN. Scheduling Unit Processing Time Jobs on a Single Machine with Multiple Criteria. *Computers and Operational Research*, 17(1) :1–7, 1990.

[34] C. L. CHEN and R. L. BULFIN. Complexity of Single Machine, Multi-criteria Scheduling Problems. *European Journal of Operational Research*, 70 :115–125, 1993.

[35] C. L. CHEN and R. L. BULFIN. Complexity of Multiple Machines, Multi-criteria Scheduling Problems. In *In 3rd Industrial Engineering Research Conference (IERC'94)*, pages 662–665, Atlanta, U.S.A, 1994.

[36] T. C. E. CHENG and N. SHAKHLEVICH. Proportionate Flow Shop with Controllable Processing Times. *Journal of Scheduling*, 2 :253–265, 1999.

[37] A. CHEVALIER. *Programmation Dynamique*. Dunod, 1977.

[38] F. D. CHOU and C. Y. LEE. Two-Machine Flowshop Scheduling with Bicriteria Problem. *Computers and Industrial Engineering*, 36(3) :549–564, 1999.

[39] C. A. COELLO COELLO. A Comprehensive Survey of Evolutionary Based Multiobjective Optimization Techniques. *Knowledge and Information Systems*, 1(3) :269–308, 1999.

[40] R. CONWAY, W. MAXWELL, and L. MILLER. *Theory of Scheduling*. Addison Wesley, Massachussets, 1967.

[41] S. COOK. The Complexity of the Theorem Prooving Procedures. *In Third ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[42] S. T. MC CORMICK and M. L. PINEDO. Scheduling n Independant Jobs on m Uniform Machines with Both Flow-time and Makespan Objectives : A Parametric Analysis. *ORSA Journal on Computing*, 7(1) :63–77, 1995.

[43] P. CZYZAK and A. JASZKIEWICZ. Pareto Simulated Annealing - A Metaheuristic for Multiple-Objective Combinatorial Optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1) :34–47, 1997.

[44] R. L. DANIELS and R. J. CHAMBERS. Multiobjective Flow-shop Scheduling. *Naval Research Logistics Quarterly*, 37 :981–995, 1990.

[45] P. DE, J. B. GHOSH, and C. E. WELLS. Scheduling to Minimize Weighted Earliness and Tardiness about a Commun Due Date. *Computers and Operational Research*, 18(5) :465–475, 1991.

[46] K. DEB. *Multi-Objective Optimization Using Evolutionary Algorithms*. Edition John Wiley & Sons, 2001.

[47] R. F. DECKRO, J. E. HEBERT, and E. P. WINKOFSKY. Multiple Criteria Job-Shop Scheduling. *Computers and Operations Research*, 9(4) :279–285, 1982.

[48] J. L. DENEUBOURG and S. GOSS. Collective Patterns and decision Making. *Ethology & Evolution*, 1 :295–311, 1989.

[49] J. L. DENEUBOURG, J. M. PASTEELS, and J. C. VERHAEGHE. Probabilistic Behaviour in Ants : A Strategy of errors ? *Journal of Theorical biology*, 105 :259–271, 1983.

[50] M. DORIGO. *Optimization, Learning and Natural Algorithms*. Thèse de Doctorat, Politecnico di Milano, Italy, 1992.

[51] M. DORIGO and DI CARO. *The Ant colony Optimization Meta-Heuristic*. New Ideas in Optimization, McGraw-Hill, 1999.

[52] M. DORIGO and L.M. GAMBARDELLA. Ant Colonies for the Travelling Salesman Problem. *BioSystems*, 43 :73–81, 1997.

[53] M. DORIGO, V. MANIEZZO, and A. COLORNI. Positive Feedback as a Search Strategy. Technical Report N° 91-016, Politecnico di Milano, Italy, 2O pages, 1991.

[54] M. DORIGO, V. MANIEZZO, and A. COLORNI. Ant System : Optimization by a Colony of Cooperating Agents. *IEEE Transactions on System, Man & Cybernitic*, 26(1) :29–41, 1996.

[55] EDITORIAL. Multicriteria Scheduling. *European Journal of Operational Research*, 167 :589–591, 2005.

[56] M. EHRGOTT and X. GANDIBLEUX. A Survey and Annotated Bibliography of Multiobjective Combinatorial Optimization. *OR Spektrum*, 22 :425–460, 2000.

[57] S. EILON and I.E. CHOWDHURY. Minimising Waiting Time Variance in the Single Machine Problem. *Management Science*, 23 :567–675, 1977.

[58] H. EMMONS. A Note on a Scheduling Problem with Dual Criteria. *Naval Research Logistics Quarterly*, 22(4) :615–616, 1975.

[59] H. EMMONS. One Machine Sequencing to Minimise Mean Flow Time with Minimum Number Tardy. *Naval Research Logistics Quarterly*, 22(4) :585–592, 1975.

[60] T. EREN and E. GÜNER. The Tricriteria Flowshop Scheduling Problem. *Int. J. Manufacturing Technology*, 2007.

[61] M. ERICKSON, A. MAYER, and J. HORN. The Niched Pareto Genetic Algorithms 2 Applied to the Design of Groundwater Remediation Systems. Dans : E. ZITZLER, K.

DEB, L. THIELE, C. A. COELLO COELLO et D. CORNE, éditeurs, First Conference on Evolutionary Multi-Criterion Optimisation, 681-695. Springer-Verlag. *Lecture Notes in Computer Science*, 1993 :681–695, 2001.

[62] P. ESQUIROL and P. LOPEZ. *L'Ordonnancement*. Economica, 1999.

[63] C. ESSWEIN, V. T'KINDT, and J. C. BILLAUT. *A Polynomial Time Algorithm for Solving a Single Machine Bicriterion Scheduling Problem*. Technical Report, Laboratory of Computer Science, University of Tours (France), 2001.

[64] G. W. EVANS. An Overview of Techniques for Solving Multiobjectif Mathematical Programs. *Management Science*, 30 (11) :1268–1282, 1984.

[65] M. L. FISHER. A Dual Algorithm for the One-Machine Scheduling Problem. *Mathematical Programming*, 11 :458–481, 1976.

[66] C. M. FONSECA and P. J. FLEMING. Genetic Algorithms for Multiobjective Optimization : Formulation, Discussion and Generalization. *Procedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo USA, 1993.

[67] P. FORTEMPS, C. OST, M. PIRLOT, J. TEGHEM, and D. TUYTTENS. Using Metaheuristics for Solving Production Scheduling Problem in a Chimical Firm : A Case Study. *International Journal of Production Economics*, 46-47 :13–26, 1996.

[68] T. D. FRY, R. D. ARMSTRONG, and R. H. BLACKSTONE. Minimising Weighted Absolute Deviation in Single Machine Scheduling. *IIE Transactions*, 19(4) :445–450, 1987.

[69] T. D. FRY, R. D. ARMSTRONG, K. DARBY-DOWMAN, and P. R. PHILIPOOM. A Branch and Bound Procedure to Minimise Mean Absolute Lateness on a Single Processor. *Computers an Operations Research*, 23(2) :171–182, 1996.

[70] T. D. FRY and R. H. BLACKSTONE. Planning for Idle Time : A Rationale for Underutilisation for Capacity. *International Journal of Production Rsearch*, 26(12) :1853–1859, 1988.

[71] T. D. FRY and G. LEONG. A Bi-criterion Approach to Minimising Inventory Costs on a Single Machine when Early Shipments are Forbidden. *Computers and Operations Research*, 14(5) :363–368, 1987.

[72] T. D. FRY and G. K. LEONG. A Bi-criterion Single Machine Scheduling with Forbidden Early Shipments. *Computers and Operations Research*, 10(2) :133–137, 1986.

[73] C. GAGNÉ, W. L. PRICE, and M. GRAVEL. Scheduling a Single Machine with Sequence Dependent Setup Times Using Ant Colony Optimization. *Soumis à IEEE Transactions on Evolutionary Computation*, Décembre 2000.

[74] R. GANGADHARAN and C. RAJENDRAN. A Simulated Annealing Heuristic for Scheduling in a Flowshop with Bicriteria. *Computers and Industrial Engineering*, 27(1-4) :473–476, 1994.

[75] M. R. GAREY and D. S. JOHNSON. *Computers and Intractability : A Guide to the Theory of $\mathcal{NP}$-Completness*. Freeman and Company, New York, 1979.

[76] M. R. GAREY, R. E. TARJAN, and G. T. WILFONG. One-Processor Scheduling with Symmetric Earliness and Tardiness Penalties. *Mathematics of Operational Research*, 13(2) :330–348, 1988.

[77] F. GEMBICKI. *Vector Maximisation of Control with Performance and Parameter Sensitivity Indices*. Thèse de Doctorat, Case Western Reserve University, Cleveland, U.S.A, 1973.

[78] A. M. GEOFFRION. Proper Efficiency and the Theory of Vector Maximisation. *Journal of Mathematical Analysis and Applications*, 22 :618–630, 1968.

[79] F. GLOVER. Heuristic for Integer Programming using Surrogate Constraints. *Decision Sciences*, 8 :156–166, 1977.

[80] F. GLOVER. Future Paths for Integer Programming and Links to Artificial Intellegence. *Decision Sciences*, 13 :533–549, 1986.

[81] F. GLOVER. Tabou Search-Part i. *ORSA Journal on Computing 1*, pages 190–206, 1989.

[82] F. GLOVER. Tabou Search-Part ii. *ORSA Journal on Computing 2*, pages 4–62, 1990.

[83] D. GOLDBERG. *Genetic Algorithms Search, Optimization, and Machine Learning*. Addition Wesley, Reading, USA, 1989.

[84] T. GONZALEZ and S. SAHNI. Open Shop Scheduling to Minimise Finish Time. *Journal of the Association for Computing Machinery*, 23 :665–679, 1976.

[85] T. GONZALEZ and S. SAHNI. Flow Shop and Job Shop Schedules : Complexity and Approximation. *Operations Research*, 26(1) :36–52, 1978.

[86] S. GOSS, R. BACKERS, J. L. DENEUBOURG, S. ARON, and J. M. PASTEELS. *How Trail Laying and Trail Following Can Solve Foraging Problems for Ant colonies*, volume G20 of *NATO-ASI Series*. Springer Verlag, Berlin, 1990.

[87] GOtha. Les Problèmes d'Ordonnancement. *R.A.I.R.O Recherche Opérationnelle / Operations Research*, 27(1) :77–150, 1993.

[88] R. L. GRAHAM, E. L. LAWLER, J. K. LENSTRA, and A. H. G. RINNOOY KAN. Optimization and Approximation in Deterministic Sequencing and Scheduling : a Survey. *Annals of Discrete Mathematics*, 5 :287–326, 1979.

[89] J. GUPTA, A. HARIRI, and C. POTTS. Single-Machine Scheduling to Minimize Maximum Tardiness with Minimum Number of Tardy Jobs. *Annals of Operations Research*, 92 :107–123, 1999.

[90] J. GUPTA and R. RAMNARAYANAN. Single Facility Scheduling with Dual Criteria : Minimizing Maximum Tardiness Subject to Minimum Number of Tardy Jobs. *Production Planning and Control*, 7 :190–196, 1996.

[91] J. N. D. GUPTA, K. HENNIG, and F. WERNER. Local Search Heuristic for the Two-Machine Bicriteria Flowshop Scheduling Problem with a Secondary Criterion. *Computers and Operations Research*, 29(2) :113–149, 2002.

[92] J. N. D. GUPTA, J. C. HO, and A. A. A. VANDERVEEN. Single Machine Hierarchical Scheduling with Customer Orders and Multiple job Classes. *Annals of Operations Research*, 70 :127–143, 1997.

[93] J. N. D. GUPTA, V. R. NEPPALLI, and F. WERNER. Minimising Total Flow Time in a Two-Machine Flowshop Problem with Minimum Makespan. *International Journal of Production Economics*, 69(3) :323–338, 2001.

[94] J. N. D. GUPTA and F. WERNER. On the Solution of 2-Machine Flow and Open Shop Problems with Secondary Criteria. *In 15th ISPE/IEE International Conference on CAD/CAM, Robotics, and Factories of the Future, Aguas de Lindoia, Sao Paulo, Brasil.*

[95] S. K. GUPTA and T. SEN. Minimising a Quadratic Function of Job Lateness on a Single Machine. *Engineering Costs of Production Economic*, 7(3) :187–194, 1983.

[96] S. K. GUPTA and T. SEN. Minimising the Range of Lateness on a Single Machine. *Journal of Operational Research Society*, 35 :853–857, 1984.

[97] M. P. HANSEN. Tabu Search for Multiobjective Optimization : MOTS. Technical report, Presented at 13$^{\text{th}}$ International Conference on MCDM, Technical University of Danmark, 1997.

[98] H. HECK and S. ROBERTS. A Note on the Extension of a Result on Scheduling with Secondary Criteria. *Naval Research Logistics Quarterly*, 19 :59–66, 1972.

[99] A. HERTZ and D. KLOBER. A Framework for the Description of Evolutionary Algorithms. *European Journal of Operational Research*, 126(1) :1–12, 2000.

[100] J. C. HO and Y. L. CHANG. A New Heuristic for the n-job, m-machine Flowshop Scheduling. *European Journal of Operational Research*, 52 :194–202, 1991.

[101] J. C. HO and Y. L. CHANG. Minimising the Number of Tardy Jobs for m Parallel Machines. *European Journal of Operational Research*, 84 :343–355, 1995.

[102] J. A. HOOGEVEEN. *Single-Machine Bicriteria Scheduling*. Thèse de Doctorat, CWI, Amsterdam, The Netherlands, 1992.

[103] J. A. HOOGEVEEN. Multicriteria Scheduling. *European Journal of Operational Research*, 167 :592–623, 2005.

[104] J. A. HOOGEVEEN and S. L. VAN DE VELDE. Minimising Total Completion Time and Maximum Cost Simultaneously is Solvable in Polynomial Time. *Operational Research Letters*, 17 :205–208, 1995.

[105] J. HORN, N. NAFPLIOTIS, and D. E. GOLDBERG. A Niched Pareto Genetic Algorithms for Multiobjective Optimization. *Procedings of the First IEEE Conference on Evolutionary Computation IEEE World Congresson Computational Intelligence*, pages 82–87, Piscataway USA, 1994.

[106] K. HUCKERT, R. RHODE, O. ROGLIN, and R. WEBER. On the Interactive Solution to a Multicriteria Scheduling Problem. *Zeitchrift fur Operations Research*, 24 :47–60, 1980.

[107] H. ISHIBUCHI and T. MURATA. Multi-Objective Genetic Local Search Algorithm and its Applications to Flowshop Scheduling. *IEEE Transactions on Systems, Man and Cybernetics - Part C : Applications and Reviews*, 28(3) :392–403, 1998.

[108] J. R. JACKSON. Scheduling a Production Line to Minimise Maximum Tardiness. *Research report 43, Management Science Research Project, UCLA*, 1955.

[109] R. J. W. JAMES and J. T. BUCHANAN. A Neighbourhood Scheme with a Compressed Solution Space for the Early/Tardy Scheduling Problem. *European Journal of Operational Research*, 102 :513–527, 1997.

[110] R. J. W. JAMES and J. T. BUCHANAN. Performance Enhancements to Tabu Search for the Early/Tardy Scheduling Problem. *European Journal of Operational Research*, 106 :254–265, 1998.

[111] A. JASZKIEWICZ. Comparison of Local Search-Based Metaheuristics on the Multiple Objective Knapsack Problem. *Fondations of Computing and Decision Sciences*, 26(1) :99–120, 2001.

[112] A. JASZKIEWICZ. Genetic Local Search for Multi-Objective Combinatorial Optimization. *European Journal of Operational Research*, 137(1) :50–71, 2002.

[113] T. C. JOHN. Tradeoff Solutions in Single Machine Production Scheduling for Minimizing Flow Time and Maximum penalty. *Computers and Operations Research*, 16(5) :471–479, 1984.

[114] S. M. JOHNSON. Optimal Tow and Tree stage Production Schedules with Setup Time Included. *Naval Research Quarterly*, 1 :61–68, 1954.

[115] A. H. RINNOOY KAN. *Machine Scheduling Problems : Classification, Complexity and Computations*. Nijhoff, The Hague, 1976.

[116] J. KANET. Minimising the Average Deviation of Job Completion Times about a Commun Due Date. *Naval Research Logistics Quarterly*, 28(4) :643–651, 1981.

[117] J. KANET. Minimising Variation of Flow Time in Single Machine Systems. *Management Science*, 27(12) :1453–1459, 1981.

[118] R. M. KARP. *(Reductibility Among Combinatorial Problems) in R. E. MILLER and J. W. THARCHER (eds) (Complexity of Computer Computations)*. Number 85-94. Plenum Press, New York, 1972.

[119] Y. D. KIM and C. A. YANO. Minimising Mean Tardiness and Earliness in Single Machine Scheduling Problems with Unequal Due Dates. *Naval Research Logistics*, 41 :913–933, 1994.

[120] A. KIRAN and A. UNAL. A Single Machine Problem with Multiple Criteria. *Naval Research Logistics Quarterly*, 38 :721–727, 1991.

[121] S. KIRKPATRICK, C. GELLAT, and M. VECCHI. Optimization by Simulated Annealing. *Science 220*, pages 671–680, 1983.

[122] S. K. KONDAKCI, E. EMRE, and M. KOKSALAN. Scheduling of Unit Processing Time Jobs on a Single Machine. *In [G. FANDEL and T. GAL]*, pages 654–660, 1997.

[123] C. KOULAMAS. Single-Machine Scheduling with Times Windows and Earliness/Tardiness Penalties. *European Journal of Operational Research*, 91 :190–202, 1996.

[124] H. W. KUHN. The Hungarian Method for the Assignement Problem. *Naval Research Logistics Quarterly*, 2 :83–98, 1955.

[125] R. KUMAR and P. ROCKETT. Improved Sampling of the Pareto-Front in Multiobjective Genetic Optimizations by Steady-State Evolution : A Pareto Converging Genetic Algorithm. *Evolutionary Computation*, 10(3) :283–314, 2002.

[126] G. J. KYPARISIS and C. KOULAMAS. Openshop Scheduling with Makespan and Total Completion Time Criteria. *Computers and Operations Research*, 27 :15–27, 2000.

[127] E. L. LAWLER. Optimal Sequencing of a Single Machine Subject to Precedence Constraints. *Management Science*, 19 :544–546, 1973.

[128] E. L. LAWLER, A. H. G. RINNOOY KAN, and B. LAGEWEG. Minimising Total Costs in One-Machine Scheduling. *Operations Research*, 23 :908–927, 1975.

[129] J. K. LENSTRA, A. H. G. RINNOOY KAN, and P. BRUCKER. Complexity of Machine Scheduling Problems. *Annals of Discrete Mathematics*, 1 :343–362, 1977.

[130] J. Y. T. LEUNG and G. H. YOUNG. Minimising Schedule Length Subject to Minimum Flow Time. *SIAM Journal on Computing*, 18(2) :314–326, 1989.

[131] C. J. LIAO and R. H. HUANG. A Algorithm for Minimising the Range of Lateness on a Single Machine. *Journal of Operational Research Society*, 42(2) :183–186, 1991.

[132] C. J. LIAO, W. C. YU, and C. B. JOE. Bicriteria Scheduling in Two-Machine Flowshop. *Journal of Operational Research Society*, 42(2) :183–186, 1997.

[133] K. S. LIN. Hybrid Algorithm for Sequencing with Bicriteria. *Journal of Optimization Theory and Applications*, 39(1) :105–124, 1983.

[134] S. LIN. Computer Solutions of the Travelling Salesman Problem. *Bell System Technical Journal*, 44 :2245–2269, 1965.

[135] T. LOUKIL. *L'Ordonnancement Multicritère de la Production : Fondemens, Nouvelles Approches de Résolution et Applications*. Thèse de Doctorat d'État non publié, Université de Sfax, Avril 2001.

[136] P. LUCIC and D. TEODOROVIC. Simulated annealing for the Multi-objective Aircrew Rostring Problem. *Transportation Research Part A : Policy and Practice*, 33 :(1) :19–45, 1999.

[137] B. L. MACCATHY and J. LIU. Addressing the Gap in Scheduling Research : a Review of Optimization and Heuristic Methods in Production Scheduling. *International Journal of Production Research*, 31(1) :59–79, 1993.

[138] R. MAZZINI and V. A. ARMENTANO. A Heuristic for Single Machine with Early and Tardy Costs. *European Journal of Operational Research*, 128 :129–146, 2001.

[139] R. MCNAUGHTON. Scheduling with Deadlines and Loss Functions. *Management Science*, 6 :1–12, 1959.

[140] F. MENCZER, M. TEGERATU, W. N STREET, and D. TUYTTENS. Efficient and Scalable Pareto Optimization by Evolutionary Local Selection Algorithms. *Evolutionary Computation*, 8(2) :223–247, 2000.

[141] K. MIETTINEN. *On the Methodology of Multiobjective Optimization with Applications*. Thèse de Doctorat, University of Jyvaskyla, Department of Mathematics, Jyvaskyla, Finland., 1994.

[142] S. MIYAZAKI. One Machine Scheduling Problem with Dual Criteria. *European Journal of Operational Research Society of Japan*, 24(1) :37–50, 1981.

[143] S. MOHRI, T. MASUDA, and H. ISHII. Bi-criteria Scheduling Problem on Three Identical Parallel machines. *International Journal of Production Economics*, 60-61 :529–536, 1999.

[144] Y. MONDEN. *Toyota Production System*. Engineering and Management Press, Norcross, GA, 1968.

[145] J. M. MOORE. An n Jobs, One Machine Sequencing Algorithm for Minimising the Number of Late Jobs. *Management Science*, 15(1) :102–109, 1968.

[146] T. MURATA, H. ISHIBUCHI, and H. TANAKA. Multi-Objective Genetic Algorithm and its Applications to Flowshop Scheduling. *Computers and Industrial Engineering*, 30(4) :957–968, 1996.

[147] A. NAGAR, S. S. HERAGU, and J. HADDOCK. A Branch-and-Bound Approach for a Two-Machine Flowshop Scheduling Problem. *Journal of Operational Research Society*, 46 :721–734, 1995.

[148] M. NAWAZ, E. ENSCORE, and I. HAM. A Heuristic Algorithm for the m-machine, n-job flow-shop Sequencing Problem. *Management Science*, 11 :91–95, 1983.

[149] R. T. NELSON, R. K. SARIN, and R. L. DANIELS. Scheduling with Multiple Performance Measures : the One-Machine Case. *Management Science*, (32)4 :464–479, 1986.

[150] V. R. NEPPALLI, C. L. CHEN, and J. N. D. GUPTA. Genetic Algorithms for the Two-Machine Bicriteria Flowshop Problem. *European Journal of Operational Research*, 95 :356–373, 1996.

[151] E. NOWICKI. An Approximation Algorithm for the m-machine Permutation Flow Shop Scheduling Problem with Controllable Processing Times. *European Journal of Operational Research*, 70 :342–349, 1993.

[152] E. NOWICKI and S. ZDRZALKA. Two-machine Flow Shop Scheduling Problem with Controllable Processing Times. *European Journal of Operational Research*, 34 :208–220, 1988.

[153] P. S. OW and T. E. MORTON. The Single Machine Early/Tardy Problem. *Management Science*, 35(2) :177–190, 1989.

[154] S. S. PANWALKER and W. ISKANDER. A Survey of Scheduling Rules. *Operations Research*, 25 :45–61, 1977.

[155] P. PINEDO. *Scheduling Theory, Algorithms and Systems*. Prentice Hall, Englewood Cliffs, N.J, 1995.

[156] C. RAJENDRAN. Two-Stage Flowshop Scheduling Problem with Bicriteria. *Journal of Operational Research Society*, 43(9) :871–884, 1992.

[157] C. RAJENDRAN. A Heuristic for Scheduling in Flowshop and Flowline-based Manufacturing Cell with Multi-criteria. *International Journal of Production Research*, 32(11) :2541–2558, 1994.

[158] C. RAJENDRAN. Heuristics for Scheduling in Flowshop with Multiples Objectives. *European Journal of Operational Research*, 82 :540–555, 1995.

[159] R. RAMESH, S. ZIONTS, and M. H. KARWAN. A Class of Practical Interactive Branch-and-Bound Algorithms for Multicriteria Integer Programming. *European Journal of Operational Research*, 26 :161–172, Juin 1986.

[160] F. RIANE, N. MESKENS, and A. ARTIBA. Bicriteria Scheduling Hybrid Flowshop Problems. *In International Conference on Industrial Engineering and Production Management (IEPM'97), Fucam*, pages 34–43, Lyon, France, 1997.

[161] B. ROY. *Méthodologie Multicritère d'Aide à la Décision*. Economica, Paris, France, 1985.

[162] B. ROY and D. BOUYSSOU. *Aide Multicritère à la Décision : Méthodes et Cas*. Economica, Paris, France, 1993.

[163] A. J. RUIZ-TORRES, E. E. ENSCORE, and R. R. BARTON. Simulated Annealing Heuristics for the Average Flow-time and the Number of Tardy Jobs Bi-criteria Identical Parallel Machine Problem. *Computers and Industrial Engineering*, 33(1-2) :257–260, 1997.

[164] S. SAHNI. Preemptive Scheduling with Due Dates. *Operations Research*, 27 :925–934, 1979.

[165] E. SANLAVILLE. *Conception et Analyse d'Algorithmes de Liste en Ordonnancement Préemptif*. PhD thesis, Université de Paris VI, Paris, France, 1992.

[166] S. C. SARIN and R. HARIHARAN. A Two Machine Bicriteria Scheduling Problem. *International Journal of Production Economics*, 65 :125–139, 2000.

[167] S. SAYIN and S. KARABATI. A Bicriteria Approach to the Two Machine Flow Shop Scheduling Problem. *European Journal of Operational Research*, 113 :435–449, 1999.

[168] J. D. SCHAFFER. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. *Procedings of the First International Conference on Genetic Algorithms*, pages 93–100, 1985.

[169] L. SCHWARTZ. *Cours d'Analyse*. Hermann, 1967.

[170] A. SEIDMANN, S. S. PANWALKER, and M. L. SMITH. Optimal Assignement of Due-Dates for a Single Processor Scheduling Problem. *International Journal of Production Research*, 19(4) :393–399, 1981.

[171] W. J. SELEN and D. D. HOTT. A Mixed Integer Goal-Programming Formulation of a Flowshop Scheduling Problem. *Journal of Operational Research Society*, 37 :1121–1128, 1986.

[172] T. SEN and S. K. GUPTA. A Branch-and-Bound Procedure to Solve a Bicriterion Scheduling Problem. *IIE Transactions*, 15(1) :84–88, 1983.

[173] T. SEN, F. RAISZADEH, and P. DILEEPAN. A Branch-and-Bound Approach to the Bicriterion Scheduling Problem Involving Total Flow Time and Range of Lateness. *Management Science*, 34(2) :255–260, 1988.

[174] P. SERAFINI. Simulated Annealing for Multiobjective Optimization Problems. *Proceedings of the 10^{th} International Conference on Multiple Criteria Decision Making*, pages 87–96, Taipei, Taiwan, 1992.

[175] J. G. SHANTIKUMAR. Scheduling n jobs on One Machine to Minimize The Maximum Tardiness with Minimum Number Tardy. *Computers and Operations Research*, 10(3) :255–266, 1983.

[176] F. S. SIVRIKAYA-SERIFOGLU and G. ULUSOY. A Bicriteria Two Machine Permutation Flowshop Problem. *European Journal of Operational Research*, 107 :414–430, 1998.

[177] W. E. SMITH. Various Optimizers for Single-Stage Production. *Naval Research Logistics Quarterly*, 3(1) :59–66, 1956.

[178] R. M. SOLAND. Multicriteria Optimization : A General Characterization of Efficient Solutions. *Decision Sciences*, 10 :27–38, 1987.

[179] F. SOURD and S. KEDAD-SIDHOUM. An Efficient Algorithm for the Earliness-Tardiness Scheduling Problem. Technical report, LIP6, CNRS, Université de Pierre et Marie Curie, 8 rue du Capitaine Scott,75015 Paris, France, Septembre 2005.

[180] N. SRINIVAS and K. DEB. Multiobjective Optimization Using Non Dominated Sorting in Genetic Algorithm. *Evolutionary Computation*, 2(3) :221–248, 1995.

[181] R. STEUER. *Multiple Criteria Optimization : Theory, Computation and Application.* John Wiley & sons, 1986.

[182] P. S. SUNDARARAGHAVAN and M. U. AHMED. Minimising the Sum of Absolute Lateness in Single-Machine and Multi-Machine Scheduling. *Naval Research Logistics Quarterly*, 31(2) :325–333, 1984.

[183] A. SUPPAPITNARM, A. SEFFEN, G. T. PARKS, and P. J. CLARKSON. A Simulated Annealing Algorithm for Multiobjective Optimization Problems. *Engineering Optimization*, 33(1) :59–85, 2000.

[184] W. SZWARC. Single-Machine Scheduling to Minimise Absolute Deviation of Completion Times From a Common Due Date. *Naval Research Logistics*, 36 :663–673, 1989.

[185] W. SZWARC. Adjacent Ordrings in Single Machine Scheduling with Earliness and Tardiness Penalties. *Naval Research Logistics*, 40 :229–243, 1993.

[186] G. TALBI. Métaheuristiques pour l'Optimisation Combinatoire Multi-Objectifs : Etat de l'Art. *Rapport Interne*, Université des Sciences et Technologies de Lille, France, juin 1999.

[187] M. TEGZE and M. VLACH. Impoved Bounds for the Range of Lateness on a Single Machine. *Journal of Operational Research Society*, 39 :675–680, 1988.

[188] V. T'KINDT and J. C. BILLAUT. *Multicriteria Scheduling : Theory, Models and Algorithms.* Springer, Tours (France), deuxième edition, 2005.

[189] V. T'KINDT, J. C. BILLAUT, and C. PROUST. Solving a Bicriterion Scheduling Problem on Unrelated Parallel Machines Occurring in the Glass Bottle Industry. *European Journal of Operational Research*, 135 :(1) :42–49, 2001.

[190] V. T'KINDT, J. N. D. GUPTA, and J. C. BILLAUT. Two-Machine Bicriteria Flowshop Scheduling Problem with a Secondary Criterion. *Computers and Operations Research*, 30(4) :505–526, 2003.

[191] V. T'KINDT, N. MONMARCHÉ, F. TERCINET, and D. LAUGT. A Ant Colony Optimization Algorithm to Solve a 2-Machine Bicriteria Flowshop Scheduling Problem. *European Journal of Operational Research*, 142(2) :250–257, 2002.

[192] A. TUZIKOV, M. MAKHANIOK, and R. MANNER. Bicriterion Scheduling of Identical Processing Time Jobs by Uniform Processors. *Computers and Operations Research*, 25(1) :31–35, 1998.

[193] E. L. ULUNGU, J. TEGHEM, P. H. FORTEMPS, and D. TUYTTENS. MOSA Method : A Tool for Solving Multiobjective Combinatorial Optimization Problems. *Journal of Multicriteria Decision Analysis*, 8 :221–236, 1999.

[194] A. D. VAN VALDHUIZEN and G. B. LAMONT. Multiobjective Evolutionary Algorithms : Analyzing the state-of-the-Art. *Evolutionary Computation*, 8(2) :125–147, 2000.

[195] D. VANDERPOOTEN. *L'Approche Interactive dans l'Aide Multicritère à la Décision.* Thèse de Doctorat, Université de Paris IX, Dauphine, Paris, France., 1990.

[196] R. G. VICKSON. Choosing the Job Sequence and Processing Times to Minimise Total Processing plus Flow Cost on a Single Machine. *Operations Research*, 28(5) :115–167, 1980.

[197] R. G. VICKSON. Two single Machine Sequencing Problems Involving Controllable Job Processing Times. *IIE Transactions*, 12(3) :158–162, 1980.

[198] A. VIGNIER. *Contribution à la Résolution des Problèmes d'Ordonnancement de Type Monogamme Multimachine (Flow-shop Hybride).* PhD thesis, E3i, Université de Tours, Tours, France, 1997.

[199] P. VINCKE. *Aide Multicritère à la Décision.* Collection SMA, Ellipse, Paris, France, 1989.

[200] L. VAN WASSENHOVE and K. R. BACKER. A Bicriterion Approach to Time /Cost Trade-offs in Sequencing. *European Journal of Operational Research*, 11(1) :48–54, 1982.

[201] L. VAN WASSENHOVE and L. F. GELDERS. For Solution Techniques for a General One Machine Scheduling Problem : A Comparative Study. *European Journal of Operational Research*, 2(4) :281–290, 1978.

[202] L. VAN WASSENHOVE and L. F. GELDERS. Solving a Bicriterion Scheduling Problem. *European Journal of Operational Research*, 4 :42–48, 1980.

[203] A. WIERZBICKI. The Use of Reference Objectives in Multiobjectif Optimisation. *In [G. FANDEL and T. GAL]*, pages pages 468–486, 1990.

[204] J. M. WILSON. Alternative Formulations of a Flowshop Scheduling Problem. *Journal of Operational Research Society*, 40(4) :395–399, 1989.

[205] H. XUONG. *Mathématiques Discrères et Informatiques*. Edition Masson, 1992.

[206] C. A. YANO and Y. D. KIM. Algorithms for a Class of Single Machine Weigthed Tardiness and Earliness Problem. *European Journal of Operational Research*, 52 :167–178, 1991.

[207] W. C. YEH. A New Branch-and-Bound Approach for the $n/2/Flowshop/aF + bC_{max}$ Flowshop Scheduling Problem. *Computers and Operations Research*, 26 :1293–1310, 1999.

[208] P. L. YU. Dynamic Programming in Finite-stage Multicriteria Decision Problems. Technical report, Technical Report 118, School of Business, University of Kansas, U.S.A, 1978.

[209] P. L. YU and L. SEIFORD. *Multistage Decision Problems with Multicriteria*. 1981.

[210] E. ZITZLER, M. LAUMANNS, and L. THIELE. SPEA2 : Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. *Proceedings of the EUROGEN 2001 - Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems*, Barcelona Spain, 2001.

[211] E. ZITZLER and L. THIELE. Multiobjective Evolutionary Algorithms : A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4) :257–271, 1999.

L'ORDONNANCEMENT MULTICRITÈRE

Résumé

Ce manuscrit présente d'abord un état de l'art sur quelques types de problèmes d'ordonnancement multicritère les plus importants. Après avoir évoqué le mode de traitement de ce type de problèmes - modélisation, prise en compte des critères, résolution à l'aide de procédures spécifiques - nous présentons son application aux problèmes d'ordonnancement à une seule machine, sur machines parallèles et problèmes d'ateliers. Le constat de la réalisation de ce mémoire rapporte sur l'extension d'un algorithme basé sur la méthode par séparation et évaluation en utilisant l'algorithme de liste LPT-FAM et la règle de tri EDD pour la détermination d'un ordonnancement Pareto optimal au problème $P||Lex(C_{\max}, L_{\max})$.

Mots-clés : Ordonnancement, Optimisation multi-objectif, Optimalité de Pareto, État - de - l'Art.

MULTICRITERIA SCHEDULING

Abstract

This manuscript presents a state - of - the - Art on some most important multicriteria scheduling problems first. After having mentioned how to deal with this type of problems - modelisation, taking into account the criteria, resolution using specific procedures - we present its application on single machine, parallel machines and shop scheduling problems. In this thesis, we applied an extension of an algorithm based on the branch-and-bound method using the LPT-FAM and EDD scheduling rules to determine a Pareto optimal schedule to the problem $P||Lex(C_{\max}, L_{\max})$.

Keywords : Scheduling, Multicriteria Optimization , Pareto Optimality , state - of - the - Art.