الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جامعة أبي بكر بلقايد – تلمسان –
Université Aboubakr Belkaïd – Tlemcen –
Faculté de TECHNOLOGIE

THESE

Présentée pour l'obtention du **grade** de **DOCTORAT 3ème Cycle**

Filière : **Génie industriel**

Spécialité : **Génie Industriel et Productique**

Par : **SEKKAL Norelhouda**

Sujet

Investigation sur l'ordonnancement des systèmes de production sous des contraintes non-conventionnelles

Soutenue publiquement, le 21 / 04 / 2021, devant le jury composé de :

M. MELIANI Sidi Mohammed	Professeur	Université de Tlemcen	Président
M. BELKAID Fayçal	MCA	Université de Tlemcen	Directeur de thèse
Mme SARI-TRIQUI Lamia	MCA	Université de Tlemcen	Co-Directeur de thèse
Mme BOUTIFOUR Zohra	MCA	Ecole Nationale Polytechnique d'Oran	Examinatrice
Mme KOULOUGHLI Sihem	MCA	Université de Tlemcen	Examinatrice

Dédicace

À la mémoire de mon père « Le soufi mon amour ». À tous les Soufis de la Tarika Tijania

À ma mère, la femme guerrière qui ne cesse de combattre pour nous.

À mes chères sœurs, celles qui me soutiennent sans cesse et sans conditions Fatima, Batoul, Aouicha, Kamila et Hafsa.

À mon unique et cher frère, la prunelle de nos yeux, Amine.

À mes grands amours, cher mari et mon fils, sans lesquels j'aurai fini cette thèse des mois avant.

À mes adorables, charmants, fascinants, aimables neveux Yassine, Djalil, MoulayAhmed, Abdelhak, Sarouri, Youcef, Anes, Ali, Rayan et mon fils avec eux, Aman.

Remerciements

Après avoir rendu grâce à Dieu, c'est un devoir agréable d'exprimer en quelques lignes la reconnaissance que je dois à tous ceux qui ont contribué de près ou de loin à l'élaboration de ce travail, qu'ils trouvent ici mes vifs respects et ma profonde gratitude.

Au directeur de thèse Monsieur BELKAID Fayçal, maître de conférences A à l'université de Tlemcen, de m'avoir inspiré le sujet, pour son encadrement fructueux, sa bienveillance et ses encouragements. À la co-directrice Madame SARI TRIQUI Lamia, maître de conférences A à l'université de Tlemcen, pour son suivi rigoureux et ses précieux conseils.

À Monsieur MELIANI Sidi Mohammed le directeur du laboratoire MELT, de m'avoir fait le privilège d'assurer la présidence de cette thèse. Aux examinateurs qui m'ont fait honneur de participer à mon jury de thèse ; Madame BOUTIFOUR Zohra maître de conférences A et chef de département du Génie industriel à l'Ecole Nationale Polytechnique d'Oran et Madame KOULOUGHLI Sihem maître de conférences A à l'université de Tlemcen.

À Madame MENADJLIA Nardjes pour son incontournable aide dans cette thèse. À tous ceux qui ont croisé mon chemin professionnel ; enseignants et encadrant. À la personne qui nous a fournit, par son intelligence et ses compétences, une très bonne formation, un remerciement spécial et sincère au Professeur SARI Zaki. À cette Kazakhstanaise qui a sacrifié sa liberté pour permettre aux chercheurs un accès gratuit et rapide à toute la documentation nécessaire, Alexandra Elbakyan Merci.

À tous mes proches ; familles et amis, il est finalement plus facile d'écrire la thèse entière que de témoigner en quelques lignes de toute l'affection, l'amour et l'amitié que j'ai envers vous, merci d'avoir été présents et de toutes les choses que des simples mots n'expriment pas :

À mes parents, ceux qui ont planté en moi les principes de la vie, du respect d'autrui, ceux qui m'ont transmis tant de choses et m'ont appris les valeurs qui donnent de la richesse à l'existence.

À mes sœurs Fatima, Batoul, Aouicha, Kamila et Hafsa, celles que je respire et qui me tiennent debout. Toutes les métaphores linguistiques ne sauront exprimer mon amour et ma gratitude envers vous. À mon très cher et unique frère, Amine mon soutien et repère dans cette vie.

À ma belle mère Assia pour tous ses efforts et son inlassable dévouement à sa famille, à mon beau frère Malek et ma belle-sœur Meriem, et à toi Ismahane pour toutes les fois où tu m'a gardé Aman, pour ta patience et ta bonté avec lui, pour ton soutien et ta sagesse ; ta présence parmi nous ne fait qu'embellir notre vie, Merci.

Le meilleur reste pour la fin, à mon autre moitié, mon ange gardien, je garde ma plus profonde gratitude, à mon Mari, pour sa générosité infinie, pour son soutien et sa compréhension, pour tous les efforts qu'il fournit à être là, à nos côtés, Merci Hamid, infiniment.

Sommaire

Liste des figures

Liste des tableaux

Introduction générale

L'évolution technologique exige aux entreprises de plus en plus de réactivité pour qu'elles puissent s'adapter aux fluctuations du marché. Dans ces conditions, l'optimisation des systèmes de production devient un enjeu important pour les industriels. Parmi les problèmes d'optimisation couramment abordés, il y a les problèmes d'ordonnancement qui sont au cœur de nombreux sujets de recherche. Ces derniers visent à prédéfinir l'affectation des tâches aux machines et l'allocation des ressources aux tâches, en prenant en considération les contraintes du système.

L'ordonnancement est un processus d'aide à la prise de décision important dans les systèmes de production. Il s'agit d'un ensemble de techniques qui visent à modéliser mathématiquement et résoudre les problèmes de priorité des tâches et d'indisponibilité des ressources, afin de maitriser les coûts et les délais de production.

La notion d'ordonnancement à été introduite pour la première fois vers les années 50 par (Johnson, 1954), elle consiste à prédéfinir l'affectation et la séquence des opérations (tâches) aux différents operateurs (machines), l'allocation des ressources aux tâches ou aux machines, avec l'élaboration des dates de début et de fin de traitement de chaque tâche et les dates d'utilisation de chaque machine et ressource, en minimisant un critère donné, (Pinedo, 2008). Cela en respectant les différentes contraintes du système tel que les contraintes de ressources, et aussi en prenant en considération les fluctuations qui parviennent dans le système tel que les effets de détérioration ou d'apprentissage.

La recherche en ordonnancement est un domaine très vaste, qui a incité plusieurs chercheurs à développer et à étudier des modèles d'ordonnancement, (Cheng, 1990), (Torabi et al., 2013). Au début ces modèles d'ordonnancement étaient trop basiques, ils négligeaient les effets de détérioration et/ou d'apprentissage, autrement dit, le temps opératoire d'une tâche était considéré constant et indépendant de tout autre paramètre. Alors qu'en réalité le temps opératoire d'une tâche peut dépendre des propriétés de la tâche et de l'état de la machine, c'est à dire si une tâche perd ses propriétés et se détériore, elle devient plus difficile à exécutée et donc nécessite plus de temps pour son traitement, la même chose se passe lorsqu'une machine se détériore, avec le temps et le nombre d'utilisation la machine perd en performances et devient plus lente dans l'exécution des tâches. Dans l'étude de (Browne and Yechiali, 1990) l'effet de la détérioration a

été modélisé par une augmentation dans le temps opératoire de la tâche, et l'effet d'apprentissage par une diminution dans le temps opératoire, et depuis la plupart des chercheurs en ordonnancement intègrent l'effet de détérioration et/ou d'apprentissage dans leurs études. De nombreuses études ont combiné entre effet de détérioration et contraintes de ressources, (Yin et al., 2014), (Yin et al., 2015).

La plupart des études menées dans le domaine l'ordonnancement classique considèrent un système composé de machines et d'un seul type de ressource, (Celano et al., 2008), (Edis and Oguz, 2012), (Belkaid et al., 2013), or qu'en réalité un système de production comprend des machines et plusieurs types de ressources ; comme les lubrifiants et le carburant qui sont deux types de ressources consommables très souvent nécessaires dans les ateliers de tournage, ou encore comme les ressources humaines et les outils qui sont des ressources renouvelables discrètes interdépendantes, où l'outil ne peut être exploité que par une ressource humaine. Ces ressources, quelque soit leurs catégories, (consommables, renouvelable, discrète ou continue), elles sont aussi importantes que les machines pour le processus de production. À savoir que le type d'une ressource dépend de sa fonction, il existe des ressources qui contrôlent le temps opératoire d'une tâche, ces ressources sont dites flexibles si l'augmentation de leur consommation diminue le temps opératoire, et elles sont dites qualifiées si le temps opératoire dépend discrètement et non-uniformément d'elles. Ces deux types de ressources représentent un atout majeur en brisant les goulots d'étranglement et en réduisant les délais,(Daniels et al., 1999).

Souvent dans les systèmes manufacturiers interviennent les ressources de type flexible, qui peuvent être des adjuvants, des accélérateurs ou des opérateurs, elles sont souvent considérées dans les problèmes d'ordonnancement avec des temps opératoires contrôlables. Les ressources qualifiées, tel que les ressources humaines qualifiées, quant à elles, sont rarement considérées. Très peu d'étude traite un problème d'ordonnancement avec ressources qualifiées, et encore moins d'étude traite un problème d'ordonnancement avec ressources humaines qualifiées.

L'intégration des ressources dans un modèle d'ordonnancement est d'une grande importance vue leurs rôles qui affectent de manière considérable les systemes manufacturiers, ainsi que l'importance que nous constatons dans la considération de plusieurs types de ressources ; parmi eux les ressources flexibles et qualifiées, dans un même modèle, nous ont motivé à mener une investigation sur les problèmes d'ordonnancement combinant effet de détérioration et deux types de ressources. Dans un premier temps, nous étudions le problème d'ordonnancement de machines parallèles identiques avec effet de détérioration et deux types de ressources consommables, dont l'une est une ressource flexible, l'autre dépend du temps opératoire. Dans un deuxième temps, nous étudions le problème d'ordonnancement de machines parallèles identiques avec effet de détérioration et deux types de ressources renouvelables, à savoir ressources humaines qualifiées et ressources partagées « outils ».

L'étude de ces problèmes d'ordonnancement passe principalement par deux étapes : la première est la transformation de l'objectif en un ou plusieurs critères à optimiser, la deuxième est la modélisation mathématique de ces critères et des contraintes imposées par le système. Toutefois, même si la combinaison entre l'effet de détérioration et des contraintes de plusieurs types de ressources dans un problème multi-objectif augmente la fiabilité de l'étude réalisée et l'efficacité de son application dans les cas réels, mais elle accroit la complexité du problème d'ordonnancement, et le classe parmi les problèmes d'optimisation NP-difficiles. Par conséquent, il devient difficile de résoudre, efficacement, un tel problème d'ordonnancement par une méthode de modélisation mathématique. D'où la nécessité de développer des méthodes, dites approchées, qui permettent l'obtention de solutions approximatives mais de qualité satisfaisante, en un temps de calcul raisonnable. Les métaheuristiques sont des méthodes approchées qui ont fait preuve d'efficacité dans la résolution des problèmes d'optimisation en général, et des problèmes d'ordonnancement en particulier. Par ailleurs, elles restent limitées face aux problèmes d'ordonnancement qui contiennent plusieurs sous problèmes, comme les problèmes d'affectation de tâches, de séquençage de tâches et d'allocation des ressources. Pour cette raison, une approche appelée approche de décomposition a été développée, elle vise à décomposer le problème en plusieurs sous-problèmes, et les ordonner pour les résoudre l'un après l'autre en prenant en compte à chaque fois la solution du sous-problème précédent.

Dans ce contexte, nous proposons des modèles mathématique de programmation linéaire pour les deux problèmes d'ordonnancement considérés ; problèmes d'ordonnancement avec effet de détérioration et deux types de ressources consommables, et problèmes d'ordonnancement avec effet de détérioration et ressources renouvelables. Nous proposons une nouvelle formulation pour l'allocation dynamique des ressources renouvelables, qui est moins complexe que les formulations traditionnelles. Par la suite, nous développons des algorithmes approximatifs, basés sur le recuit simulé, le recuit simulé multi-objectif et sur l'approche de décomposition.

Les deux principaux objectifs de cette thèse se résument en deux points :

- l'introduction de nouveaux modèles d'ordonnancement des machines parallèles identiques, qui n'ont pas été traités auparavant, tel que le premier modèle prend en considération l'effet de détérioration avec deux types de ressources consommables, et le deuxième modèle combine l'effet de détérioration avec l'allocation dynamique des ressources renouvelables ; à savoir ressources humaines qualifiées et ressources partagées.
- Le développement de nouvelles approches de résolution, tel qu'un modèle mathématique de programmation linéaire moins complexe que les modèles existants, et de nouveaux algorithmes approximatifs

L'étude menée dans cette thèse est divisée en quatre chapitres, chacun est structuré comme le suivant :

Le premier chapitre présente un aperçu général sur les modèles d'ordonnancement dans les systèmes de production. Ce chapitre est réservé à la présentation des éléments de base de l'ordonnancement dans une entreprise. à savoir que les modèles peuvent être classées en deux grandes parties ;la partie des modèles de l'ordonnancement classique et la partie des modèles de l'ordonnancement actuel où est positionné notre travail. Et pour finaliser le chapitre, nous présentons un aperçu général sur les travaux menés dans le domaine de l'ordonnancement des machines parallèles, ensuite, nous présentons une revue de la littérature détaillée sur les problèmes d'ordonnancement à machines parallèles avec effet de détérioration et avec contraintes de ressources renouvelables et consommables.

Le deuxième chapitre s'articule autour des méthodes de résolution destinées à aux problèmes d'ordonnancement. La résolution des ces problèmes peut se faire de deux manières : par les méthodes exactes et méthodes approximatives, nous définissons chaque méthode et détaillons ses avantages et inconvénients, un intérêt particulier est porté aux méthodes utilisées dans notre étude, à savoir la programmation linéaire, les métaheuristiques à base d'une seule solution, les méthodes de l'optimisation multi objectif, et les approches de décompositions. À la fin de ce chapitre, nous présentons un état de l'art sur les méthodes de résolution utilisées dans les problèmes d'ordonnancement abordés dans le premier chapitre.

Le troisième chapitre présente une étude sur l'optimisation multi-objectif d'un problème d'ordonnancement de machines parallèles avec effet de détérioration et deux différents types de ressources consommables, dont l'une est une ressource consommable flexible, et l'autre est une ressource consommable dont la consommation dépend du temps opératoire. Dans un premier temps, nous développons un modèle mathématique à partir de la programmation linéaire pour résoudre le problème. Par ailleurs, la stratégie de résolution s'avère très complexe du fait que les décideurs doivent faire face des fois à des situations difficiles, qu'un modèle mathématique de programmation linéaire ne peut les résoudre d'une manière efficace. Donc, pour remédier à cela, nous développons une méthode approximative multi-objectif, qui est le recuit simulé multi objectif « MOSA ». Dans le but d'améliorer la qualité des solutions nous développons un algorithme de décomposition « 2-steps algorithm ».

Le quatrième et dernier chapitre est réservé à l'étude et la résolution d'un problème d'ordonnancement multi-objectif où la consommation totale en énergie et le makespan sont minimisés dans un environnement de machines parallèles combinant effet de détérioration et contraintes de disponibilité de deux types de ressources renouvelables, à savoir outils et ressources humaines qualifiées, qui travaillent dans un environnement où les tâches sont accentuées par un temps opératoire dépendent d'elles. L'étude de ce problème passe, tout d'abord, par une analyse minutieuse du fonctionnement du système interprétée par une modélisation mathématique. Le modele proposé considére de manière orginale les contraintes de l'allocoation dynamique des deux ressources renouvelables. Étant donné la complexité de ce problème et du nombre de contraintes et de paramètres à prendre

en considération, sa résolution exacte par le biais du modèle mathématique s'avère difficile contenu la capacité de calcul nécessaire. Pour cette raison, nous développons un algorithme de décomposition multi-objectif, basé sur la métaheuristique du recuit simulé.

Finalement, nous présentons une conclusion pour clôturer la thèse, ainsi que des perspectives qui serviront comme orientations et repères pour de futures recherches.

Chapitre 1

Cadre conceptuel de la recherche

Résumé : La première partie de ce chapitre présente les problèmes d'ordonnancement. Nous rappelons d'abord les différentes notions de base de l'ordonnancement classique qui présentent des contraintes par rapport à la disponibilité des tâches et la capacité des machines. Puis nous expliquons d'autres contraintes qui sont non-conventionnelles et dont la considération rapproche le modèle d'ordonnancement théorique à des cas réels. Ceci est suivi d'un état de l'art sur le domaine étudié et la problématique abordé afin de positionner notre travail de recherche par rapport aux travaux existants.

Sommaire :

1.1 Introduction

L'évolution industriel actuelle exige des entreprises plus réactives, qui répondent aux besoins du marché, et s'adaptent à la variation des exigences de leurs clients. Ces changements nécessitent une gestion particulière des sites de production riche en contraintes, où les entités de calcul ne cessent de se développer. De plus, des imprévus qui interviennent de manière plus fréquente procurent diverses problématiques au niveau de ces sites. Et pour assurer le bon fonctionnement des unités de production, qui représentent le maillon conducteur indispensable de l'entreprise industrielle pour la réalisation de ses objectifs, la résolution des problématiques relatives à la fonction de l'ordonnancement est nécessaire. Cela se fait par le développement de méthodes utiles qui se traduisent par une intégration et une application des solutions théoriques au monde pratique, afin d'améliorer la productivité et la gestion quotidienne.

Dans un problème d'ordonnancement interviennent deux notions fondamentales : les tâches et les ressources. Une ressource peut être représentée par un moyen, technique ou humain, avec une capacité connue à priori. Une tâche est un travail élémentaire dont la réalisation nécessite un certain nombre d'unités de temps et d'unités de ressources. Ordonnancer un ensemble de tâches, c'est programmer leur exécution en leur allouant les ressources requises et en fixant leurs dates de début et dates de fin pour un plan de production optimal.

L'étude d'un problème d'ordonnancement passe tout d'abord par l'identification de tous les paramètres du système considéré : disposition des machines, priorité des tâches, besoins de ressources…etc. Ces paramètres diffèrent d'un atelier à un autre, selon le type des produits, leurs structures et leur circulation dans l'atelier.

Cependant, l'ordonnancement, dans les environnements de production complexes, peut devenir une tâche extrêmement difficile. Par conséquent, plusieurs chercheurs et praticiens se sont consacrés pour développer des techniques et des approches pour faire face aux problèmes d'ordonnancement, aussi complexes qu'ils soient. La première étape du développement de ces approches était la définition des notions de l'ordonnancement et la classification de ses problèmes, ensuite, l'introduction de certains modèles de bases qui modélisent des problèmes purement théoriques ; et ce n'est qu'à partir de la classification des problèmes d'ordonnancement et de ces modèles de base qu'on a pu développer des modèles plus complexes qui représentent des problèmes des cas réels des environnements manufacturiers. Ces derniers, qu'on appelle les modèles d'ordonnancement actuels sont représentés par des modèles classiques auxquels on a intégré des contraintes, appelées des contraintes non-conventionnelles.

Ce chapitre sera organisé comme suit, la prochaine section définira la fonction de l'ordonnancement dans une entreprise, celle qui la suit nous décrira les modèles de l'ordonnancement classique et leurs notions fondamentales. Ensuite, la quatrième

section introduira les modèles de l'ordonnancement actuel sous deux contraintes non-conventionnelles, qui sont l'effet détérioration et la contraintes de ressources additionnelles. Finalement les deux sections 5 et 6 présenterons un aperçu sur les études relatifs aux problèmes d'ordonnancement à machines parallèles avec effet de détérioration, et ceux avec contraintes de ressources, respectivement.

1.2 Fonction de l'ordonnancement dans une entreprise

Une entreprise est représentée par un ensemble de maillons concaténés, où chaque maillon est doté d'une fonction bien précise, pour un objectif commun. La gestion de l'entreprise nécessite une optimisation de toutes les opérations depuis l'approvisionnement de la matière première jusqu'à l'obtention du produit fini, elle nécessite aussi un système d'information performant qui assure la communication de ces différents maillons.

La fonction de l'ordonnancement apparait dans l'établissement d'un plan d'approvisionnement efficace, dans la programmation des arrivages de la matière première et leur stockage, elle apparait dans l'établissement aussi d'un plan de production qui consiste en la définition des gammes d'usinage et du passage et de l'ordre des tâches dans les machines, et en l'organisation des différents moyens de transport. En administration la fonction de l'ordonnancement apparait dans l'organisation et le classement des différents documents et papiers de l'entreprise et dans la gestion des différentes ressources humaines en précisant le rôle de chacune.

L'ordonnancement étant un outil de prise de décision à court terme, il intervient non seulement sur le niveau opérationnel, mais aussi tactique et stratégique, comme illustré dans la figure 1.1. Au niveau opérationnel, et dans chaque département de l'entreprise, l'ordonnancement organise le fonctionnement du département, optimise l'exploitation des ressources et minimise les coûts. Et grâce à un système informationnel performant, où chaque département est doté d'un ordinateur connecté à un ordinateur central, les départements supérieurs se permettent un accès à toutes les informations concernant les départements locaux. Cela facilite la prise de décision tactique comme la planification de la production, ainsi la prise de décision stratégique comme les décision d'installation de nouveaux espaces, (Pinedo, 2008).

1.3 Les modèles de l'ordonnancement classique

« Ordonnancer, c'est programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution. […]. Enfin, il faut programmer les tâches de façon à optimiser un certain objectif… »

(Carlier and Chrétienne, 1988)

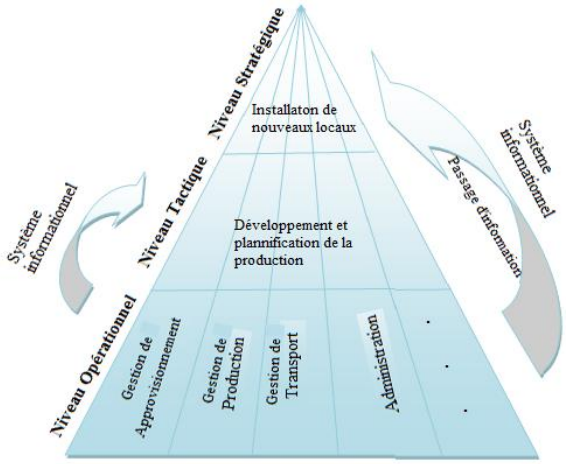


Figure 2.1 : La fonction de l'ordonnancement dans les différents niveaux décisionnels d'une entreprise

La fonction de l'ordonnancement est une fonction à court terme, même si elle a un impact sur les décisions tactiques et stratégiques. Elle définit des intervalles de temps durant lesquels les tâches sont exécutées. Autrement dit, l'ordonnancement détermine les dates de début et de fin de l'exécution de chaque tâche, et les dates d'utilisation de chaque machine/ressource. Ces fonctions revêtent deux aspects :

- Statique : quand l'exécution de toutes les tâches est définie tout au début de l'ordonnancement, sans tenir compte des changements qui se produisent durant le temps de l'exécution.
- Dynamique : Un ordonnancement dynamique se fait en temps réel. Les tâches sont affectées compte tenu de l'état de ressources et de l'avancement des travaux dans le temps.

Ainsi dans un problème d'ordonnancement classique, cinq notions fondamentales interviennent : les tâches, les machines, les contraintes, les objectifs et les critères.

1.3.1 Définition des éléments fondamentaux de l'ordonnancement classique

De nombreux modèles d'ordonnancement avec des différentes caractéristiques existent dans la littérature. Ces modèles sont construits à partir de quelques notions de base, à commencer par une (ou plusieurs) tâche qui doit passer par une (ou plusieurs) machine où elle doit subir une valeur ajoutée, en vérifiant certaines conditions représentées par des contraintes à respecter, à fin d'obtenir l'objectif visé, représenté par la notion du critère.

1.3.1.1 *Tâche*

Une tâche ou une activité est une entité élémentaire de travail, comme représenté dans la figure 1.2, elle est définie par une date de début, une date de fin et un temps opératoire, nécessitant un ensemble de ressources pour son exécution, (Esquirol and Lopez, 1999).

1.3.1.2 ***Machine***

La machine est la ressource fondamentale à l'exécution des tâches. Elle peut être définie par un certain nombre de caractéristiques; la capacité de production, degré de flexibilité, gamme d'usinage à réaliser...etc, (Esquirol and Lopez, 1999).

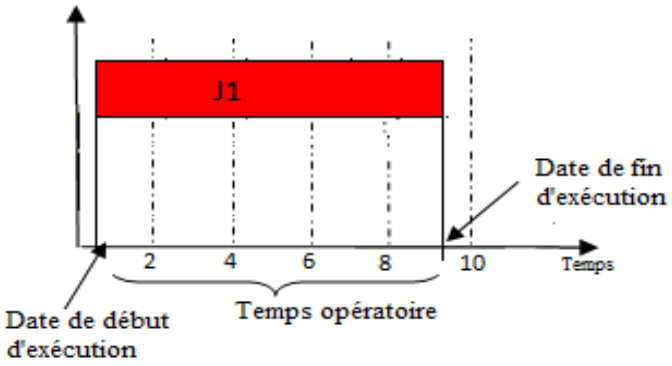


Figure 2.2 : les éléments définissant une tâche

1.3.1.3 ***Contraintes***

C'est une interaction entre les variables de décisions d'un problème qui peuvent être expliquées par: le nombre de produits, la taille du lot… De manière générale, une contrainte est une condition qui doit être respectée dans un environnement donné.

La théorie de l'ordonnancement distingue deux catégories de contraintes : les contraintes temporelles et les contraintes de ressource.

- Contraintes temporelle : les contraintes temporelles représentent les conditions relatives à la disponibilité des tâches en termes de priorité.
- Contrainte de précédence : cette contrainte définit les liens de précédences entre les tâches, lorsqu'une tâche doit s'exécuter avant une autre, comme dans la figure 1.3 où la tâche J doit impérativement s'exécuter avant la tâche I.

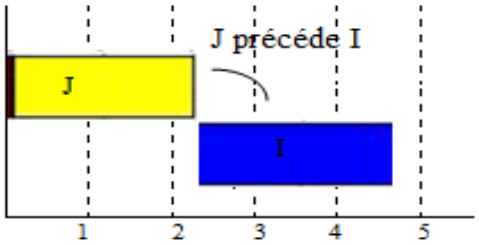


Figure 2.3 : Lien de précédence entre la tâche J et I

- Contrainte de disponibilité : ou la contrainte des dates limites d'une tâche, elle assure que la date de début d'exécution d'une tâche ne soit pas programmée avant sa disponibilité.
- Contrainte de disjonction : quand deux ou plusieurs tâches ne doivent pas être superposées dans le temps, cela est exprimé par une contrainte de disjonction.

- Contraintes de ressources : On peut distinguer plusieurs types de ressources selon leurs capacités d'exécution de tâches; des ressources qui peuvent effectuer plusieurs tâches simultanées, ou des ressources qui effectuent une seule tâche à la fois. Cette restriction dépend de la capacité de la ressource, qui peut être interprétée par des contraintes cumulatives, c'est-à-dire les tâches exécutées à un instant donné ne dépasse pas la capacité de la machine.

1.3.1.4 ***Critère***

Un critère est l'élément dont la valeur doit être prise en compte et optimisée lors d'un ordonnancement. Il définit la qualité d'une solution d'un problème d'ordonnancement. Les critères à optimiser dans les problèmes ordonnancement sont divisés par (Esquirol and Lopez, 1999) en trois classes; critères de temps, de ressources et de coût.

- Critères relatif au temps : un critère de temps est un critère qui prend en considération l'aspect temporel d'un ordonnancement. Cet aspect peut être défini par le temps total de production, la somme des temps d'achèvement des machines (ou temps de cycle), la différence entre les temps d'achèvement de chaque machine, la somme des retards ou le nombre de tâches en retard. Ces critères permettent d'optimiser l'utilisation des machines et des installations de l'usine, ainsi permettent à l'atelier de répondre aux commandes juste à temps.
- Critère relatif aux ressources : l'emploi de certaines ressources rares ou coûteuses, nécessite à l'atelier de production l'optimisation de ces mêmes ressources pour éviter d'éventuelle augmentation dans les coûts de production.
- Critère relatif aux coûts : prend en considération la notion des différentes dépenses relatives à la réalisation et la transformation d'une tâche, qui sont exprimées par la minimisation des coûts de la main d'œuvre requise, des coûts du lancement de production, les coûts de configuration des machines...etc. Ces critères servent à minimiser les coûts de production tout en gardant la même qualité des produits. Bien évidement la minimisation des différentes dépenses engendre la maximisation du profit et l'augmentation du gain espéré. Par conséquent, ils aident l'entreprise à rester compétitive en termes de coût des produits, et à garder sa place sur le marché.
- Critères environnementaux: depuis quelques années, et avec la thématique du développement durable, d'autres objectifs ont également émergés. Parmi ces objectifs considérés récemment on site : la minimisation du « TEC » l'énergie consommée totale dans les problèmes de l'efficacité énergétique (ou bien du coût du TEC) (Tigane et al., 2018) et (Lu et al., 2017). Ainsi que la minimisation du taux d'émission du CO2 considérée par (Liu, 2014). L'ajout de ces critères dans les modèles d'ordonnancement à augmenter considérablement la complexité des problèmes à traiter. Ce qui incite

plusieurs chercheurs à proposer des mécanismes en développant des approches innovantes pour fournir un plan d'ordonnancement optimal d'exécution de tâches et d'utilisation de ressources à la fois.

1.3.1.5 *Objectif*

L'objectif est un but à atteindre, établi par une action décisionnelle fourni par un plan préalablement étudié. Pour un problème d'ordonnancement, la fonction objectif renvoie à l'optimisation un ou plusieurs critères. Lorsqu'il s'agit d'un seul critère, on parle d'un problème d'optimisation mono-objectif, et lorsqu'il s'agit de plusieurs critères, on parle d'un problème d'optimisation multi-objectif, (Collette and Siarry, 2002).

L'optimisation mono-objectif oblige la modélisation du problème par une équation unique, ces restrictions risquent de biaiser la modélisation et donc renvoient vers des solutions non-satisfaisantes. L'optimisation multi-objectif, quant à elle, autorise ces degrés de libertés dans la modélisation, qui permettent la considération de plusieurs critères (Collette and Siarry, 2002). Cette souplesse a fait que la plupart des problèmes étudiés dans la littérature visent à optimiser plusieurs critères. Ces derniers se trouvent, plus souvent, contradictoires et dont l'importance relative est difficile à apprécier. De même ces critères visent à améliorer les performances de l'atelier, en jouant sur le triptyque (coût, qualité, délai). Figure 1.4 modélise l'aspect contradictoire des critères à optimiser; à travers cette figure, il est remarqué qu'un emploi maximal des ressources génère des coûts, mais il mène vers une minimisation des encours et du temps de production. Tandis que l'inverse ; l'emploi minimal des ressources, minimise les coûts, mais au même temps génère une augmentation des encours et du temps de production.

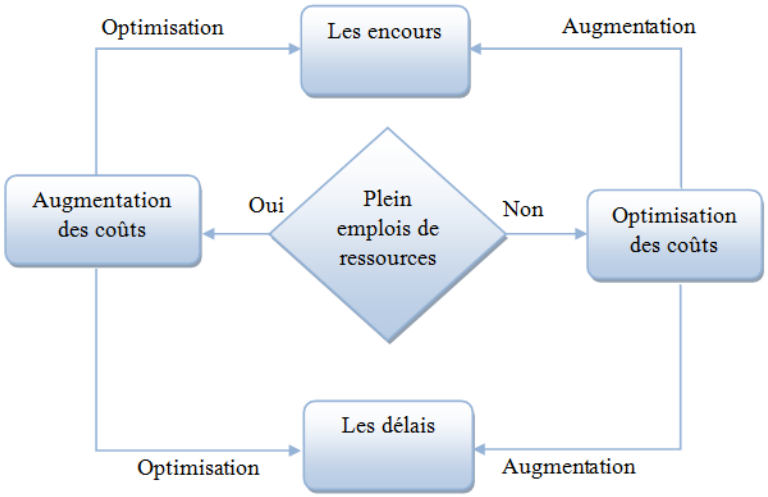


Figure 2.4 : L'aspect contradictoire des différents critères, (Collette and Siarry, 2002)

1.3.2 L'environnement des machines

Le positionnement des machines au sein des entreprises peut avoir plusieurs configurations avec des structures aussi complexes que différentes. La disposition de

ces machines peut être soit parallèle ou en série, et pour chaque configuration plusieurs ateliers peuvent avoir lieu, selon le nombre de machines qui la composent et l'ordre du passage des tâches. Pour la configuration parallèle on peut distinguer des ateliers à machines identiques, uniformes, ou non uniformes. Et pour la configuration série, on peut distinguer les ateliers flow-shop et job-shop. C'est ce qui a mené vers une classification des ateliers à ordonnancer comme illustrer par la figure 1.5:

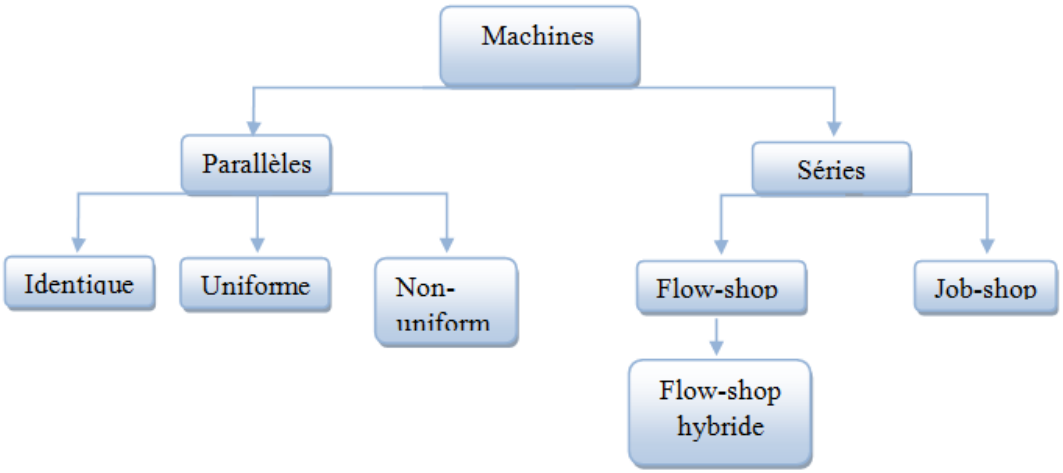


Figure 2.5 : Classification des différents types d'ateliers

1.3.2.1 *Une seule machine*

Les problèmes d'ordonnancement à une seule machine consistent à trouver la séquence adéquate des tâches dans la machine. Figure 1.6.

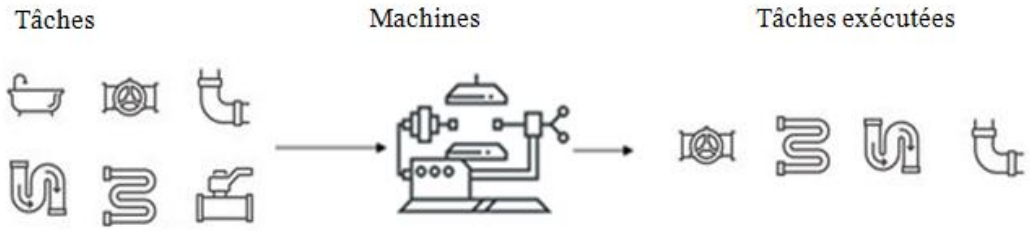


Figure 2.6 : Représentation graphique d'un atelier à machine unique

1.3.2.2 *Machines parallèles*

Les problèmes d'ordonnancement à une seule machine représentent un cas spécifique des problèmes d'ordonnancement à machines parallèles. Ces derniers consistent à trouver la séquence après avoir trouver l'affectation des tâches, de façon à ce qu'un critère soit optimisé, Figure 1.7. Parmi les problèmes d'ordonnancement à machines parallèles nous distinguons :

- Les problèmes d'ordonnancement à machines parallèles identiques (P) : le temps opératoire d'une tâche est le même pour toutes les machines.

- Les problèmes d'ordonnancement à machines parallèles uniformes (Q) : la vitesse d'exécution varie d'une machine à une autre, mais pas pour les tâches
- Les problèmes d'ordonnancement à machines parallèles non-uniformes (R) : la vitesse d'exécution varie selon les machines et selon les tâches.

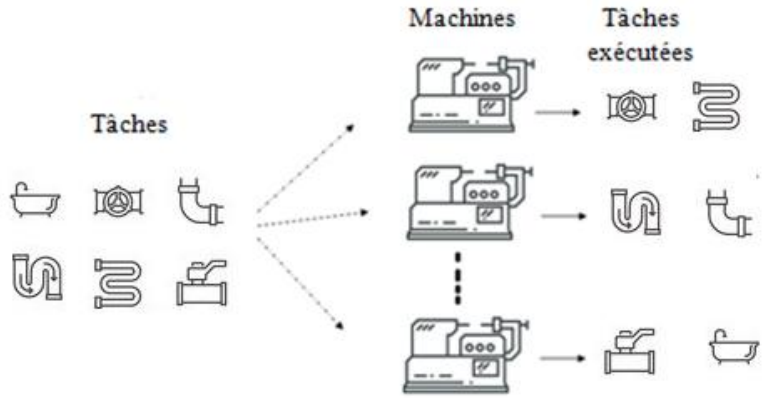


Figure 2.7 : Représentation graphique d'un atelier à machines parallèles

Les environnements des machines parallèles sont les environnements les plus considérés dans la littérature, en raison de leur large utilisation. Dans cette thèse nous traitons des problèmes d'ordonnancement dans un environnement de machines parallèles identiques. Toutefois, pour des raisons d'exhaustivité nous donnons de brèves descriptions sur les autres environnements.

1.3.2.3 ***Flow-shop***

Les problèmes d'ordonnancement de flow-shop sont des problèmes où les tâches exécutées passent par un cheminement à un seul sens. L'objectif consiste à trouver la séquence des tâches dans chaque machine. Le cas le plus étudié est lorsque les tâches suivent la même séquence sur toutes les machines. Figure 1.8.

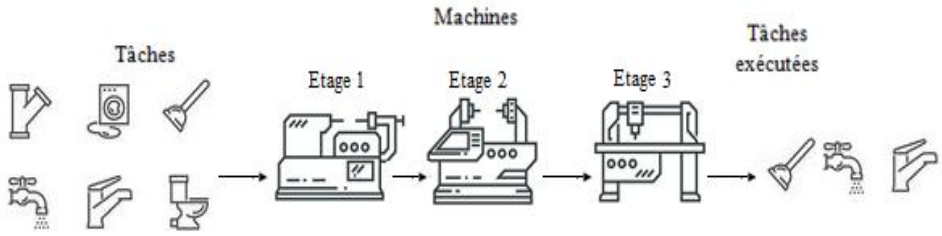


Figure 2.8 : Représentation graphique d'un atelier flow-shop

1.3.2.4 ***Flow-shop hybride***

C'est une génération de flow-shop où, sur un étage, il existe au moins deux machines parallèles pour l'exécution des tâches. Ces machines parallèles sont installées sur les étages qui représentent des goulots d'étranglement pour la chaine, Figure 1.9.

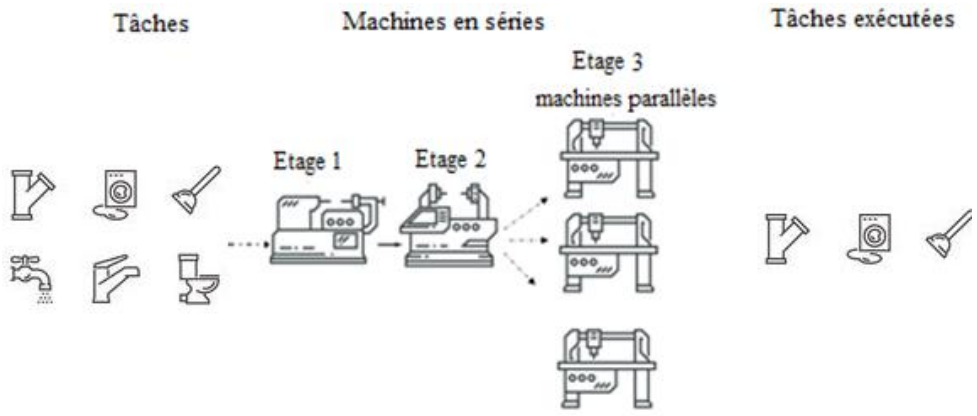


Figure 2.9 : Représentation graphique d'un atelier flow-shop hybride

1.3.2.5 *Job-shop*

Ou les ACM (ateliers à cheminement multiples), ce sont des unités manufacturières où les tâches, pour être exécutées, passent par plusieurs machines, pas forcément toute les machines, et elles ont chacune un passage différent, comme illustré dans la Figure 1.10. Ce type d'ateliers se trouve généralement dans la production artisanale.

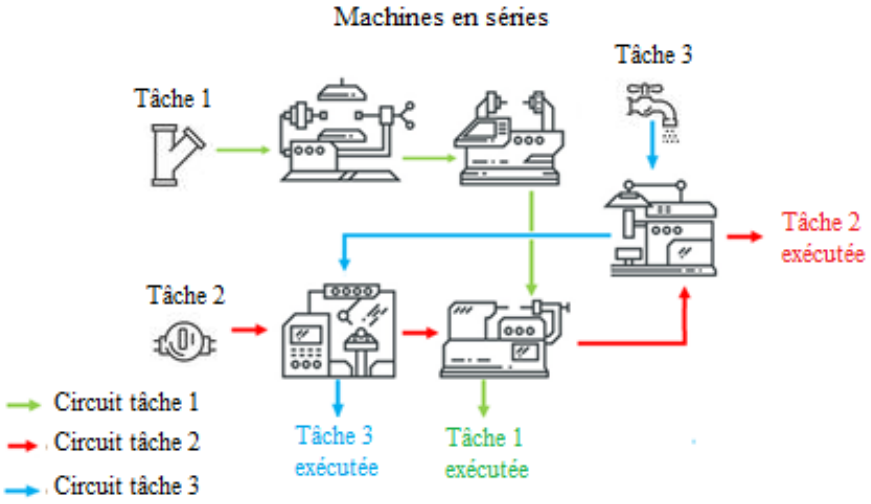


Figure 2.10 : Représentation graphique d'un atelier job-shop

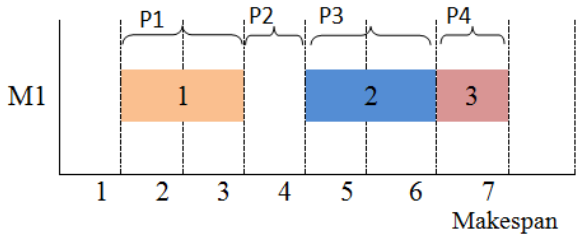
1.3.3 Représentation des problèmes d'ordonnancement

Pour représenter un problème d'ordonnancement, des études antérieures ont utilisé des outils et des techniques mathématiques. Parmi ces outils, il existe un schéma, proposé par (Graham et al., 1979), qui représente les problèmes d'ordonnancement par trois champs α|β|γ à suivre :

- α représente l'environnement machine
- β représente les caractéristiques des tâches.
- γ définit la fonction objectif (critères à optimiser)

1.3.3.1 **Représentations mathématiques**

Par ailleurs, la modélisation mathématique des problèmes d'ordonnancement permet à les étudier techniquement, à fin d'y proposer des solutions exactes ou des méthodes de résolution approximatives. Parmi les modélisations les plus abordées est la modélisation du problème en un problème d'optimisation combinatoire, où la résolution du problème consiste à trouver les combinaisons (tâche, machine, ressource…etc.) adéquates et optimales. La complexité du modèle dépend du nombre de ces combinaisons ainsi que des variables de décisions à définir par le modèle. Pour définir une variable de décision, on peut dire que c'est une variable contrôlable, à travers laquelle les éléments importants d'un ordonnancement sont définis, tel que la position d'une tâche dans la machine ou sa date de début d'exécution. Figure 1.11 illustre les variables de décision qui définissent ces deux éléments; l'une est une variable de position et l'autre est indexée par le temps.



$X_{j1,p1} = 1$ i.e. la tâche 1 est traitée dans la position 1.
$X_{j1,p2} = 0$ i.e. la tâche 1 n'est pas traitée dans la position 2.

$Y_{p3,t4} = 1$ i.e. la position 3 débute à l'instant 4.
$Y_{p2,t2} = 0$ i.e. la position 2 ne débute pas à l'instant 2.

Figure 2.11 : Les variables de décision définissant la position et la date de début d'une tâche

1.3.3.2 ***Représentation graphiques***

Une solution d'ordonnancement peut être représentée par un diagramme de Gantt. Ce dernier est constitué de plusieurs lignes ; représentant les machines. Chaque ligne comprend plusieurs tâches sous forme de rectangle, la longueur du rectangle représente le temps opératoire d'une tâche, la largeur représente son besoin en ressource s'il est considéré. Ce diagramme développé par H.-L Gantt 1910, donne un aperçu sur l'exécution des tâches dans les machines aux cours du temps. Figure 1.12 représente un diagramme de Gantt de deux machines exécutant 5 tâches.

Les modèles d'ordonnancements classiques, cités ci-dessus, ne peuvent représenter les structures des systèmes de production récents, à cause de leur complexité. Néanmoins ces modèles ont servi, et servent encore, au développement théorique des problèmes d'ordonnancement. Ce n'est qu'à partir de ces modèles de base qu'on peut modéliser des problèmes réels plus complexes. Ces modèles classiques sont représentés par les contraintes définies dans les sections d'avant, que nous appelons contraintes conventionnelles. La modélisation des d'ordonnancement

plus réels se fait par la considération d'autre exigences et d'autres contraintes, appelées contraintes non-conventionnelles, et qui sont interdépendantes aux contraintes conventionnelles.

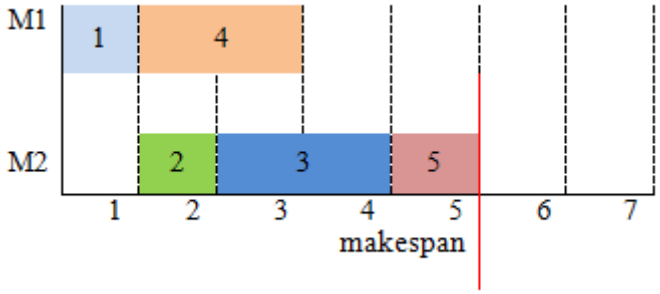


Figure 2.12 : Représentation d'un ordonnancement par diagramme de Gantt

Dans les problèmes d'ordonnancement classique, la machine est considérée comme unique ressource dont les performances restent inchangées durant tout l'horizon de l'ordonnancement. Cependant dans les cas réels, des ressources additionnelles à la machines sont très souvent nécessaires à l'exécution des tâches, ainsi la machine perd en performances au fur et à mesure qu'elle est en fonctionnement. Par cela, un ordonnancement n'est faisable et efficace à un système manufacturier que lorsqu'il considère toutes ses contraintes et les respecte. Pour cette raison, que les modèles d'ordonnancement ne cessent de se développer avec le développement technologique des systèmes manufacturiers.

1.4 Les modèles de l'ordonnancement actuel

L'évolution technologique et démographique procure des impactes significatifs sur le mode de vie, augmente le taux de la consommation pour une clientèle plus exigeante, de plus, face à une concurrence féroce sur un marché rude, conduit à une expansion massive de la production industrielle qui entraine plus de contraintes pour les systèmes productifs. Dans ce cadre, la proposition de nouvelles approches est sollicitée pour remédier efficacement à ces problèmes, en considérant les différentes contraintes additives (contraintes non-conventionnelles) dans les modèles d'ordonnancement. Dans cette thèse, nous traitons des problèmes d'ordonnancement avec la considération de deux contraintes non-conventionnelles : contrainte de détérioration et de ressources additionnelles. Pour cela, nous allons expliquer dans un premier temps l'ordonnancement avec détérioration et l'ordonnancement avec ressources additionnelles

1.4.1 L'ordonnancement avec détérioration

L'exécution d'une tâche nécessite un temps opératoire plus au moins constant. Cependant, dans plusieurs situations dans la vie réelle, les conditions de l'exécution des tâches et l'état de la machine peuvent provoquer une variation du temps

opératoire, cette variation est exprimée par la notion de détérioration, tel que donner par (Browne and Yechiali, 1990) qui ont été les premiers à l'introduire dans le domaine de l'ordonnancement.

L'effet de détérioration peut être visualisé aussi bien pour une tâche que pour une machine, la détérioration d'une tâche peut être définie par une augmentation dans son temps opératoire en fonction de sa date de début d'exécution, quant à la détérioration de la machine, elle peut être définie par une augmentation dans le temps opératoire des tâches en fonction de leur position dans la machine.

- **Exemple de détérioration d'une machine**

On suppose qu'un ensemble de pièces ont besoin d'une opération de perçage dans une perceuse. Étant donné l'état neuf de l'outil utilisé pour le perçage, le temps opératoire de la première pièce ne sera pas affecté par la détérioration, mais à cause de l'usure de l'outil, le temps opératoire augmentent en fonction du nombre des pièces suivantes.

- **Exemple de détérioration d'une tâche**

On suppose une machine en fonctionnement et qui a besoin d'une activité de maintenance, plus l'activité de maintenance est reportée dans le temps, plus la machine se détériore et donc son temps d'exécution augmente.

A partir de là, on peut distinguer deux types de fonction de détérioration :

1.4.1.1 *Fonctions de détérioration par position*

Le temps opératoire d'une tâche est une fonction croissante de sa position. Plus la position de la tâche est éloignée, plus son temps opératoire est grand. Les fonctions peuvent être linéaire ou non-linéaire: dans un tableau

Linéaire: Comme la fonction proposée par (Yang and Yang, 2010)

$$P_{jr} = p_j + r.b_j \quad (1.1)$$

Non-linéaire: Comme la fonction proposée par (Mosheiov, 2005), équation(1.2). Et celle proposée par (Yang and Yang, 2013), équation (1.3).

$$P_{jr} = p_j.r^{a_j} \quad (1.2)$$

$$P_{jr} = p_j.\sigma_j^{r-1} \quad (1.3)$$

Tel que P_{jr} est le temps opératoire actuel de la tâche j, p_j est son temps opératoire normal, r est sa position, $b_j, a_j \; ou \; \sigma_j$ sont les taux de détérioration

1.4.1.2 *Fonctions de détérioration par date de début d'exécution*

Le temps opératoire d'une tâche est une fonction croissante dépendante de la date de début de l'exécution de cette tâche. Aussi les fonctions peuvent être :

Linéaire: Comme la fonction proposée par (Gawiejnowicz, 2007), équation (1.4). Et celle proposée par (Wang et al., 2007), équation (1.5).

$$P_{jt} = b_j . t_j \quad (1.4)$$

$$P_{jt} = p_j + b_j . t \quad (1.5)$$

Non-linéaire: comme la fonction proposée par (Wang et al., 2008), équation (1.6).

$$P_{jt} = p_j . (a_j + b . t_j) \quad (1.6)$$

Tel que P_{jt} est le temps opératoire actuel de la tâche j, p_j est son temps opératoire normal, t_j est le temps de début d'exécution de la tâche j, $b_j \; ou \; b$ sont les taux de détérioration

Partiellement linéaire: comme la fonction proposée par (Chung and Kim, 2016), équation (1.7).

$$P_{jt} = \begin{cases} p_j & Si \;\; x_j \leq d_j \\ p_j + r_j . p_j & Six_j > d_j \end{cases} \quad (1.7)$$

Tel que r_j est la taux de détérioration, x_j est le temps de début d'exécution de la tâche j, d_j est la date due.

d'autres fonctions peuvent être retrouvées dans (Alidaee and Womer, 1999).

1.4.2 L'ordonnancement avec contraintes de ressources

Une ressource est un moyen technique ou humain requis pour la réalisation d'une tâche. Les problèmes d'ordonnancement avec consommation de ressources sont souvent plus compliqués, parce que l'exécution de chaque tâche nécessite en plus d'une machine disponible, une quantité de ressources. Dans la figure 1.13 nous avons essayé de modéliser l'effet d'une contrainte de ressources sur l'ordonnancement. Les tâches deux et quatre sont décalées dans le temps parce qu'elles ne peuvent pas être traitées en même temps que les autres. Cette figure illustre aussi la complexité qu'ajoutent les ressources à un problème d'ordonnancement. Ces dernières sont, soit disponibles en quantité limitée, ce qui fait d'elles une contrainte supplémentaire dans la modélisation, soit leur utilisation/consommation est un objectif à optimiser. Slowinski (1982) ont classifié les ressources par types et par catégories.

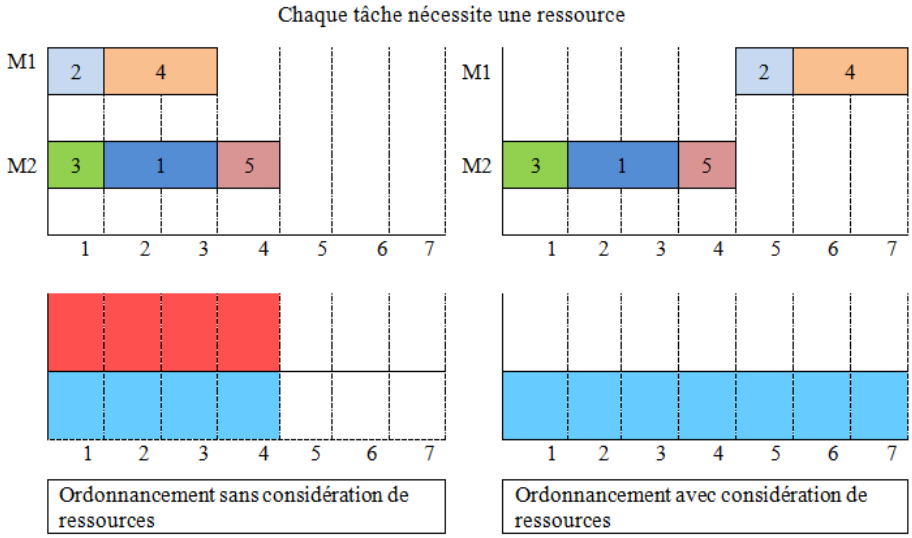


Figure 2.13 : L'impact des contraintes de ressources sur l'ordonnancement

Le classement des ressources par catégories prend en compte deux aspects: les contraintes sur les ressources et leur divisibilité. Quant au classement par type, il se fait selon la fonction de la ressource. Dans la fin de cette section nous discutons trois types de ressources qui sont : les ressources flexibles, humaines et qualifiées.

Les contraintes sur les ressources: Peuvent être interprétées par la nature de la ressource utilisée dans un système donné. Nous distinguons les ressources consommables, les ressources renouvelables, et les ressources doublement contraintes. Les ressources consommables contraignent le système de production par leur disponibilité, quant aux ressources renouvelables, elles le contraignent par leur capacité. La considération de ces ressources dans un problème d'ordonnancement affecte sa complexité de calcul en augmentant le nombre de combinaisons qui forment une solution ; (Garey and Johnson, 1975) ont examiné la complexité des problèmes d'ordonnancement de machines parallèles identiques avec ressources additionnelles, et (Blazewics et al., 1983) ont classifié les problèmes d'ordonnancement avec contraintes de ressources selon leur complexité.

1.4.2.1 *Ressources consommables*

C'est les ressources qui ne peuvent être utilisées qu'une seule fois; leur consommation est contrainte. Exemple de la matière première, pièces de rechanges…etc. Différentes assomptions, sur ces ressources, sont considérées et traitées. (Belkaid et al., 2016a) a supposé que les ressources consommables nécessaires à l'exécution des tâches ne sont pas toutes disponibles à l'instant zéro, mais elles sont livrées par des fournisseurs à des instants différents.

1.4.2.2 *Ressources renouvelables*

Ces ressources peuvent être réutilisées une fois libérée d'une tâche, mais elles ont une capacité qui, généralement, ne dépasse pas une seule tâche à la fois; leur

utilisation est contrainte. Exemple d'outils, opérateurs…etc. (Fanjul-Peyro et al., 2017) a traité un problème de machines parallèles non-uniformes avec des ressources partagées.

1.4.2.3 ***Ressources doublement contraintes***

Ce type de ressources combine les contraintes des ressources consommables et ceux des ressources renouvelables, c'est-à-dire et l'utilisation et la consommation de cette ressource sont contraintes. Exemple d'énergie, lubrifiant…etc. Dans les problèmes d'ordonnancement avec ce type de ressources, une contrainte sur leur disponibilité peut biaiser la modélisation et/ou même mener vers un modèle irrésoluble. Par ailleurs, la plupart des études des problèmes d'ordonnancement avec ressources doublement contraintes considèrent ces ressources comme un critère à optimiser.

La divisibilité des ressources: Dépend du dénombrement de celles-ci; s'il est échantillonné ou fluide. Par cela, nous distinguons deux catégories de ressources; les ressources discrètes et les ressources continues.

1.4.2.4 ***Discrète***

Une ressource est dite discrète lorsqu'elle est discrètement dénombrable, ne peut être allouée qu'en quantité discrète. Dans les installations manufacturières récentes, la présence de ressources additionnelles, comme les AGV (véhicules autoguidées), les palettes de transport…etc. est une nécessité. Ces ressources sont classées comme ressources discrètes renouvelables. Ainsi, dans les systèmes d'assemblages, les vices et les boulons utilisés dans la préhension des éléments sont considérés comme ressources discrètes consommables.

1.4.2.5 ***Continue***

Une ressource est dite continue lors qu'elle peut être allouée en quantité arbitraire. Pour des machines comme les tours, perceuses ou découpeuse de verre les lubrifiants sont nécessaires et sont consommées en quantité importante, ces lubrifiants, dans certain cas, peuvent être récupérés pour être réutilisés, on peut les classer dans la classe de ressources renouvelables continues. Les adjuvants et les carburants, quant à eux, ne peuvent être réutilisés plus qu'une seule fois, ce sont des ressources consommables continues.

Très souvent, dans les environnements manufacturiers, l'exécution d'une tâche nécessite, en plus de la machines, des ressources supplémentaires. Pour cette raison, de nombreux chercheurs dans l'ordonnancement se sont intéressés à l'étude des différents types et catégories de ressources, définis ci-dessus. Plusieurs modèles représentant les contraintes de ressources dans les environnements manufacturiers ont été développés. Vers les années 1996, la notion des ressources flexibles a été introduite par (Daniels Richard L. et al., 1996), ces ressources flexibles, qu'elles

soient consommables, renouvelables ou doublement contraintes, elles peuvent contrôler le temps opératoire d'une tâche.

1.4.2.6 *Les ressources flexibles*

Une ressource flexible est un type de ressources qui permet de contrôler le temps opératoire des tâches. Dans les formulations standards, le temps opératoire d'une tâche est considéré indépendant de la consommation de ressources. Or qu'en réalité, le temps opératoire d'une tâche peut varier en fonction de la quantité de certaines ressources qui lui sont allouées. Par exemples : dans l'industrie du béton préfabriqué, la quantité d'adjuvant utilisée dans le béton influence sur le temps de répartition et de séchage du béton. Un autre exemple se voit dans le temps de chargement et/ou du déchargement d'une marchandise, qui dépend du nombre d'opérateurs qu'y sont alloués.

En raison de l'importance des ressources flexibles qui représentent un outil stratégique dans l'amélioration des performances, un grand nombre d'étude traitant les problèmes d'ordonnancement avec ressources flexibles ont été publiés ces deux dernières décennies, parmi ces études nous citons (Richard L. et al., 1999), (Chen, 2004) and (Pei et al., 2018). De plus en plus de manufacturier s'intéressent à l'intégration des ressources flexibles, consommables ou renouvelables, dans leurs systèmes de production. Ce type de ressources pourvoit un retour immédiat sur l'investissement, en brisant les goulots d'étranglement, améliorant le taux de production, et réduisant les encours. Cependant, parmi les ressources les plus flexibles, et qui sont d'une importance cruciale dans un atelier sont les ressources humaines.

1.4.2.7 *Les ressources humaines*

Nous consacrons une sous-section à ces ressources du fait de leurs importances dans les systèmes manufacturiers, et parce qu'elles sont rarement considérées dans les problèmes d'ordonnancement étudiés. Les ressources humaines représentent toujours une nécessité pour le bon fonctionnement du système. Cependant, le besoin en ces ressources dépend du type de la tâche à effectuer (produit ou service) et du taux d'automatisation du système manufacturier. Cela dit, dans les systèmes manufacturiers automatisés, une ressource humaine intervient pour la configuration d'une machine ou pour le changement d'une pièce. Dans les systèmes semi-automatisés, une ressource humaine intervient pour l'accomplissement d'une opération dont les machines du système sont incapables d'accomplir, comme la vérification de la qualité du produit. Or que dans les systèmes non-automatisés, une ressource humaine est la ressource principale pour la réalisation de la tâche. Le meilleur exemple des systèmes manufacturiers dont la ressource humaine est la ressource principale est les systèmes de production artisanale, dont un exemple sera étudié dans le quatrième chapitre.

1.4.2.8 *Les ressources qualifiées*

Ceux sont les ressources dont les performances dépendent du type de la tâche exécutée. Cela dit qu'une ressource qualifiée est destinée à l'exécution d'un type de tâches précis, en étant incapable ou moins efficace dans l'exécution des autres types de tâches. Le meilleur exemple à donner sur les ressources qualifiées est l'exemple des ressources humaines, qui même en étant polyvalents ne peuvent partager toutes les mêmes compétences. L'exploitation de ces ressources est un atout majeur pour le développement et la flexibilité du système manufacturier, en permettant une exécution rapide et efficace des tâches. Ainsi elles représentent un avantage concurrentiel durable pour toute l'entreprise.

Dans les sections précédentes, nous avons présenté de manière générale les notions de bases de l'ordonnancement à fin d'orienter les lecteurs sur le domaine abordé dans cette thèse. Et pour positionner notre travailler dans la littérature, un tour d'horizon sur les travaux existants sera effectué et présenté dans la section suivante. Notre travaille qui s'articule autour des problèmes d'ordonnancement de machines parallèles identiques avec effet de détérioration et considération de ressources consommables continues et renouvelables discrètes.

1.5 Revue de la litérature

1.5.1 Travaux relatifs aux problèmes d'ordonnancement à machines parallèles

Un grand nombre de cas réel d'ordonnancement est représenté par les modèles d'ordonnancement à machines parallèles, nous donnons l'exemple des blocs opératoires, des guichets d'enregistrements, de perceuses ou de tours fonctionnant en parallèles. Même dans les modèles d'ordonnancement à machines en séries flexibles, on peut trouver des machines parallèles au niveau de certains étages. De ce fait, les problèmes d'ordonnancement de machines parallèles sont les plus étudiés dans la littérature.

1.5.1.1 *Les problèmes d'ordonnancement à une seule machine*

Les modèles d'ordonnancement à machine unique représentent un cas spécifique des modèles à machines parallèles. ils sont plus faciles à manipuler et permettent d'analyser le problème d'ordonnancement sous différents contraintes et objectifs, ils permettent aussi de développer des approches de résolutions plus facilement, tel que ces approches pourront être étendues pour le cas général de machines parallèles Sidney, (1977), Lakshminarayan et al., (1978) et França et al., (2001) ont tous traité un problème d'ordonnancement sur une seule machine avec comme objectif de minimiser les retards et les précocités des tâches. Chacun a proposé un algorithme différent pour la résolution du problème. à partir de ces algorithmes Valente and Schaller, (2010) ont développer une heuristique plus performante pour les problèmes d'ordonnancement de machine unique avec temps d'inactivité des machines.

Bansal, (1980) a développé un algorithme de séparations et évaluations (branch and boundalgorithm) pour la minimisation de la somme pondérée des temps d'exécution de chaque tâche. Laguna et al., (1991) ont développé un algorithme de recherche Tabou pour minimiser les coûts de configuration des machines et des retards pénalisés dans un environnement de machine unique. Lawler and Sivazlian, (1978) ont traité un problème de minimisation des coûts, tel que ces coûts varient selon le temps d'exécution d'une tâche.

1.5.1.2 *Les problèmes d'ordonnancement à machines parallèles*

Les modèles d'ordonnancement des machines parallèles sont les plus étudiés, pour la raison qu'ils représentent un grand nombre de cas réel, et en plus, ils peuvent servir à l'étude d'autres types de problèmes, comme le problème de flow-shop flexible considéré par (Rashidi et al., 2010). Une revue de la littérature des problèmes d'ordonnancement à machines parallèles a été présentée par (Cheng, 1990). Min and Cheng, (1999), Plateau and Rios-Solis, (2010) et Fanjul-Peyro and Ruiz, (2011) ont proposé des algorithmes pour résoudre les problèmes d'ordonnancement à machines parallèles non-uniformes. Min and Cheng, (1999) ont développé un algorithme génétique pour la minimisation du makespan. Plateau and Rios-Solis, (2010) ont utilisé la fameuse théorie de la programmation quadratique, ensuite ils ont soumis le modèle à une procédure de séparation et évaluation BAB. Fanjul-Peyro and Ruiz, (2011) quant à eux, ils ont développé des métaheuristiques basées sur la technique de réduction de la taille initiale du problème. Mokotoff, (2004) a proposé un algorithme exacte basé sur la résolution itérative des modèles de programmation linéaire relaxés, Dell'Amico et al., (2008) ont développé une métaheuristique et proposé un algorithme exacte basé sur la procédure de séparation et évaluation BAB. Les algorithmes développés dans les deux dernières études ont été destinés à la résolution des problèmes d'ordonnancement à machines parallèles identiques.

Cochran et al., (2003), Loukil et al., (2005) et Li et al., (2012) ont développé des algorithmes pour l'optimisation des problèmes d'ordonnancement multicritères ,dans un environnement de machines parallèle. Cochran et al., (2003) ont développé un algorithme génétique à plusieurs population, Loukil et al., (2005) ont adapté le recuit simulé multi-objectif, Li et al., (2012) ont proposé un modèle mathématique de programmation linéaire et ensuite développé un algorithme NSGA II.

Pour minimiser le retard et le TEC (consommation énergétique totale), Pan et al., (2018) ont développé une métaheuristique à population appelée ICA (imperial competitive algorithme), à laquelle ils ont intégré une méthode lexicographique pour comparé entre les solutions dans une population. Cet algorithme d'ICA amélioré à été comparé à plusieurs autres algorithmes dans la littérature; MOMA (multi-objectif memetic algorithm), algorithme de colonnie et le ICA classique, et il a prouvé son efficacité.

1.5.2 Travaux relatifs aux problèmes d'ordonnancement à machines parallèles avec effets de détérioration

L'objectif principal de la plupart des études sur l'ordonnancement, est la formulation de modèles mathématiques aussi proche que possible des problèmes réels de l'industrie. La détérioration est l'une des contraintes les plus répondues dans les milieux manufacturiers, étant donné que chaque machine, ressource ou main d'œuvre est sujette à la détérioration par usure ou par fatigue. Cependant, la plupart des études de l'ordonnancement classiques considèrent le temps opératoire d'une tâche constant, autrement dit, indépendant de cet effet. A cet égard, et surtout après l'étude de (Browne and Yechiali, 1990), dans laquelle la notion de détérioration a été introduite et modélisée, plusieurs autres études ont modélisé et traité les problèmes d'ordonnancement avec effet de détérioration. Alidaee and Womer, (1999) ont révisé les différentes fonctions de détérioration du temps opératoire.

La littérature indique l'existence de deux types de fonctions de détérioration, une fonction dépendante de la date de début d'exécution, et une autre de la position. Un troisième type de fonction de détérioration existe, rarement considéré, c'est le type des fonctions de détérioration par effet cumulative considéré dans (Rustogi and Strusevich, 2017). Dans cette section nous nous intéressons aux deux premiers types.

1.5.2.1 *Détérioration par date de début d'exécution*

Les fonctions de détérioration dépendantes du temps de début d'exécution sont considérées lorsque l'étude traite un problème d'ordonnancement avec détérioration des tâches, comme dans les ateliers du laminage d'acier, où les lingots à laminer se refroidissent s'ils ne passent pas directement du four au laminoir, et nécessiteront d'être réchauffés avant le laminage. Donc la détérioration des lingots est modélisée par la proportion du temps de réchauffement ajouté au temps opératoire normal. (X.-R. Wang and Wang, 2013).

plusieurs chercheurs ont considéré la détérioration dépendante de la date de début d'exécution de la tâche, parmi ceux :(Wu and Lee, 2003a), (Cheng et al., 2004) et (Jeng and Lin, 2005), Ces études ont traité le problème d'ordonnancement à une seule machine avec un temps opératoire des tâches dépendent de la date de début d'exécution. Chen, (1996) a étudié le même problème, mais sur un environnement de machines parallèles. Kang and Ng, (2007) ont proposé un algorithme approximative pour minimiser le makespan dans un environnement de machines parallèles avec détérioration des tâches, dans un temps polynomial. Dans le même contexte, Cheng et al., (2007) ont développé une heuristique, mais avec l'objectif de déterminer les dates dues adéquates tout en minimisant les pénalités des retards et des latences. Ji and Cheng, (2009) ont étudié des problèmes d'ordonnancement à machines parallèles avec une fonction de détérioration linéairement dépendante du temps de début d'exécution L'étude a été faite sous différents critères et a prouvé que ces problèmes sont NP-difficiles.

Un autre type de fonction de détérioration existe dans la littérature ; c'est les fonctions partiellement linéaires, ces fonctions sont utilisées lorsque l'effet de la détérioration commence à un seuil donné, et s'arrête à un autre seuil. Ces fonctions peuvent se trouver dans l'exemple d'un échantillon de sol (carotte) à analyser dans un laboratoire de travaux publics, où la carotte se dégrade après quelques heures de son enlèvement du sol. Ji and Cheng, (2007) ont considéré un problème d'ordonnancement à machines parallèles avec une fonction de détérioration partiellement linéaire, où le temps opératoire commence à croitre à un instant donné « d », et s'arrête à croitre à un autre instant donné « D ». Ainsi Chung and Kim, (2016) ont considéré une fonction de détérioration partiellement linéaire, où, si le date de début $x_j$est inférieur ou égale au seuil de détérioration d_j, $x_j \leq d_j$, le temps opératoire actuel est égale au temps opératoire normal. Si la date de début dépasse le seuil d_j,$x_j > d_j$, le temps opératoire actuel est défini par la somme du temps opératoire normal et le rapport du taux de détérioration $r_j$et le seuil de détérioration d_j,$p_j + r_j . d_j$.

1.5.2.2 ***Détérioration par position***

Les fonctions de détérioration dépendantes de la position, ou dépendantes de la séquence de la tâche, sont considérées lorsque, dans l'étude, nous cherchons à modéliser le vieillissement ou l'usure d'une machine après son utilisation, ou la fatigue des opérateurs humains. Cela a été premièrement introduit par (Gawiejnowicz, 1996). Ensuite Toksarı and Güner, (2009) ont traité un problème d'ordonnancement avec temps opératoire dépendent de la position de la tâche, l'objectif est de minimiser les retards et les latences des tâches. Zhao and Tang, (2010) ont considéré un problème d'ordonnancement à machines parallèles avec détérioration et activités de maintenance. La fonction de détérioration est dépendante de la position de la tâche :$P_j^r = p_j . r^{a_j}$ tel que : a_j est le taux de détérioration, r est la position et p_j est le temps opératoire normal. L'objectif est de minimiser le makespan. Considérant le même problème Yang and Yang, (2010) ont considéré le même problème, tel que la fréquence des activités de maintenances et leurs positions sont à déterminer. Ruiz-Torres et al., (2013), quant à eux, ils ont modélisé la détérioration en une position h d'une machine k par le ratio$q_{hk} = (1 - d_{x[h-1,k]k}) . q_{k(h-1)}$ tel que k définit la machine, $x[h-1,k]$ définit la tâche affectée à la machine k dans la position précédente (h-1), $d_{x[h-1,k]k}$ définit le taux de détérioration relatif à la tâche précédemment exécutée par la machine k, $q_{k(h-1)}$ est le ratio de détérioration dans la position précédente. Finalement, ils ont modélisé le temps opératoire actuel P ' par le rapport du temps opératoire normal p sur le ratio de détériorationq_{hk}, $P'_{x[h,k]k} = p_{x[h,k]k}/q_{hk}$. Rustogi and Strusevich, (2012) ont modélisé le temps opératoire avec détérioration par la fonction $P_j(r) = p_j . g(r)$, tel que : r représente la position de la tâche et g(r) est un facteur de détérioration par position. Zhang et al., (2018) ont proposé un nouveau modèle de fonctions de temps

opératoire dépendant de la position. Ding et al., (2019) aussi a considéré l'effet de la détérioration par position.

1.5.3 Travaux relatifs aux problèmes d'ordonnancement avec contraintes de ressources

L'optimisation des coûts dans les entreprises manufacturières consiste en l'exploitation optimale de toutes les ressources de l'entreprise (machines, employés, main-œuvre, outils…etc.). Pour cette raison, la machine ne doit pas être considérée comme l'unique ressource dans la modélisation des problèmes des environnements manufacturiers. Dans ce cas, des ressources additionnelles interviennent dans l'ordonnancement des tâches. Ces ressources peuvent être renouvelables, consommables, discrètes ou continues. Cette notion de ressources renouvelables a été introduite par (Blazewicz and Lenstra, 1983). Une partie des problèmes d'ordonnancement avec contraintes de ressources est placée dans un contexte où ces ressources additionnelles sont toujours disponibles en quantité suffisante, mais elles ont un coût à optimiser. Comme les problèmes d'efficacité énergétique considérés par (Lu et al., 2017) ou par (Tigane et al., 2018). Or que dans une autre partie de problèmes, les ressources ne sont disponibles qu'en quantité limitée, par conséquent, l'ordonnancement doit respecter la disponibilité des ressources. Un nombre considérable d'études existantes dans la littérature traitent les problèmes d'ordonnancement avec contraintes de disponibilité des ressources : Belkaid et al., (2013) ont traité le problème d'ordonnancement à machines parallèles identiques avec des ressources consommables dont la quantité nécessaire n'est pas toute disponible dès l'instant 0. Özpeynirci et al., (2016) ont considéré, dans un environnement de machines parallèles, la contraintes de plusieurs types de ressources renouvelables, indispensables à l'exécution des tâches. Ces ressources sont disponibles quantité limitée. Dans l'étude de (Edis et al., 2013) les problèmes d'ordonnancement avec ressources additionnelles ont été révisés et discutés.

Étant donné que l'intérêt de cette thèse est porté aux problèmes d'ordonnancement à machines parallèles, dans ce qui suit, nous réviserons les travaux relatifs aux problèmes d'ordonnancement à machines parallèles avec contraintes de ressources. Ces contraintes, dans la littérature, sont divisées en plusieurs classes ; les plus abordées sont les contraintes de ressources renouvelables, consommables et flexibles.

1.5.3.1 *Ressources renouvelables*

Les problèmes d'ordonnancement à machines parallèles avec ressources renouvelables représentent un grand nombre de problèmes réels, Généralement, dans toutes les industries, et particulièrement, dans les installations industrielles modernes dont la présence d'AGV ou de robots est nécessaire. Etant donné l'intérêt porté à ces problèmes les manufacturiers, l'implication des chercheurs à développer des modèles réalistes ne cesse d'augmenter. Blazewics and Kubiak, (1987) sont parmi les premiers qui ont étudié le problème d'ordonnancement de machines parallèles

identiques, avec contraintes de ressources additionnelles (discrètes et renouvelables), pour minimiser le temps de cycle. Ventura and Kim, (2003) ont étudié le problème d'optimisation des déviations totales absolues entre les temps de cycle de chaque tâche. Kellerer and Strusevich, (2003) ont étudié la complexité du problème d'ordonnancement des machines parallèles dédiées avec contraintes de ressources renouvelables. Par machines dédiées on veut dire que les taches sont préalablement affectées aux machines. Huang et al., (2010) ont traité le problème d'ordonnancement des machines parallèles dédiées avec un seul serveur, pour comme objectif de minimiser le makespan. Ce même problème a été considéré par (Kim and Lee, 2012), mais avec une affectation de tâches non-spécifiée. Récemment, Fanjul-Peyro et al., (2017) et Villa et al., (2018) ont considéré le problème de minimisation de makespan dans un environnement de machines parallèles non-uniformes avec contraintes de ressources additionnelles. Le problème d'ordonnancement avec machines parallèles et contraintes de ressources a été traité par (Özpeynirci et al., 2016) et (Gökgür et al., 2018), où les ressources sont présentes sous formes d'ensembles d'outils nécessaires à l'exécution des tâches, et sont disponibles en quantités limitées.

1.5.3.2 ***Ressources consommables***

Les contraintes de ressources consommables se trouvent dans la littérature en deux façons ; La première est lorsque les ressources ne sont pas disponibles en quantité suffisante dès l'instant 0 : Ces contraintes se trouvent dans les problèmes d'ordonnancement des industries du textile, cosmétiques ou l'industrie du meuble, elles se trouvent aussi dans les chaines d'assemblages pour l'attache des différentes pièces. Toker et al., (1991) ont traité un problème d'ordonnancement à machine unique avec des ressources consommables. Xie, (1996) a lui aussi traité un problème d'ordonnancement à machine avec plusieurs ressources consommables, ils les ont nommé : des ressources financières. Ainsi Györgyi and Kis, (2015) ont considéré que les ressources consommables nécessaire à l'exécution des tâches sont livrées à des dates préalablement connues en quantité limité. Belkaid et al., (2016a) et Györgyi and Kis, (2017) ont, tous les deux, considéré la contrainte de ressources consommables indisponibles dans un problème d'ordonnancement à machines parallèles. La deuxième façon est lorsque la ressource est disponible en quantité suffisante, mais elle a un coût considéré comme objectif à minimisé. Vickson, (1980) sans avoir à mentionné la nécessité de ressources, ils ont considéré que l'exécution d'une tâche génère des coûts, donc ils ont traité le problème de minimisation des coûts d'exécution des tâches dans un environnement de machine unique. Avec les préoccupations croissantes concernant le changement climatique et dans le contexte de la diminution des ressources énergétiques fossiles, les ressources considérées comme fonction à optimiser, sont généralement les ressources énergétiques. Wang et al., (2018a) ont traité un problème multi-objectifs de minimisation d'énergie totale consommée « TEC » et du makespan, dans un environnement de machines parallèles identiques. Ce même problème a été traité

par (Tigane et al., 2018) dans un environnement de machines parallèles non-uniformes et par (Zhang et al., 2019) dans un environnement de machines parallèles non-uniformes avec changement d'outil.

1.5.3.3 *Ressources flexibles*

Les problèmes d'ordonnancement avec ressources flexibles reflètent des problèmes de plusieurs cas réel de l'industrie, comme mentionner précédemment. Pour cette raison que, après l'introduction de la notion des ressources flexibles par (Daniels Richard L. et al., 1996), les chercheurs s'intéressent de plus en plus aux problèmes d'ordonnancement avec temps opératoire dépendant des ressources. Daniels et al., (1999) ont étudié le problème d'ordonnancement à machines parallèles dans lequel l'effet opérationnel des ressources flexibles a été considéré. Shabtay and Kaspi, (2006) ont considéré le problème d'affectation et de séquençage des tâches aux machines et d'allocation de ressources, tel que le makespan est minimiser. Ces ressources sont consommables et contrôlent le temps opératoire. Ruiz-Torres et al., (2007) ont étudié le problème d'ordonnancement à machines parallèles avec ressources, tel que la vitesse des machines dépend de la quantité des ressources allouées, avec comme objectif de minimiser le nombre de tâches en retard. Su and Lien, (2009) et Low and Wu, (2016) ont étudié le problème d'ordonnancement à machine parallèle, avec une fonction de temps opératoire linéairement dépendante aux ressources allouées aux tâches$p_{ij}(r_{ij}) = b_{ij} - a_{ij} * r_{ij}$.d'autre études sur l'ordonnancement avec ressources flexibles comme : (Chuang et al., 2009) (Yang et al., 2014) et (Hsieh et al., 2015).

1.5.3.4 *Ressources humaines*

La contrainte des ressources humaines est très peu considérée dans les problèmes d'ordonnancement étudiés, mis à part quelque papiers ; (Celano et al., 2008) et (Garcia et al., 2018).

Agnetis et al., (2014) a considéré un système de production artisanal de type job-shop avec des ressources humaines spécifiquement qualifiées. Abdeljaoued et al., (2018) a considéré un problème d'ordonnancement à machines parallèles avec des ressources qualifiées qui peuvent être des employés qualifiés.

1.5.4 Travaux relatifs aux problèmes d'ordonnancement avec détérioration et ressources flexibles

Wang et al., (2012) ont considéré un problème d'ordonnancement à machine unique, avec un temps opératoire de tâche dépendant de la position de la tâche, de la date de début d'exécution et de la quantité de ressources lui ont allouées. Wei et al., (2012) ont étudié le problème d'ordonnancement à machine unique, avec un temps opératoire dépendant du début d'exécution de la tâche et des ressources allouées. Deux objectifs ont été considérés séparément à savoir : la minimisation d'une fonction de coût du makespan, temps de cycle, différences absolues totales entres les

temps de cycles et coût des ressources ; et la minimisation d'une fonction de coût du makespan, temps d'attente, différences absolues totales des temps d'attente et le coût des ressources. X.-Y. Wang and Wang, (2013) et X.-R. Wang and Wang, (2013) ont tout les deux considéré détérioration des tâches avec un temps opératoire dépendant des ressources allouées, dans un environnement de machines parallèles afin d'optimiser une fonction à plusieurs critères.

Hsu and Yang, (2014) ont analysé le problème d'ordonnancement de machines parallèles non-uniformes, avec un temps opératoire dépendant de la position de la tâche et des ressources allouées. Pei et al., (2018) ont considéré un temps opératoire dépendant des ressources et de l'effet d'apprentissage.

Nous remarquons que dans toutes ces études menées sur les problèmes d'ordonnancement avec contraintes de ressources, aucune étude n'a traité les problèmes d'ordonnancement avec plusieurs types de ressources ayant différents impact sur le système. En outre, les ressources humaines sont rarement considérées dans la littérature, encore moins les ressources humaines qualifiées.

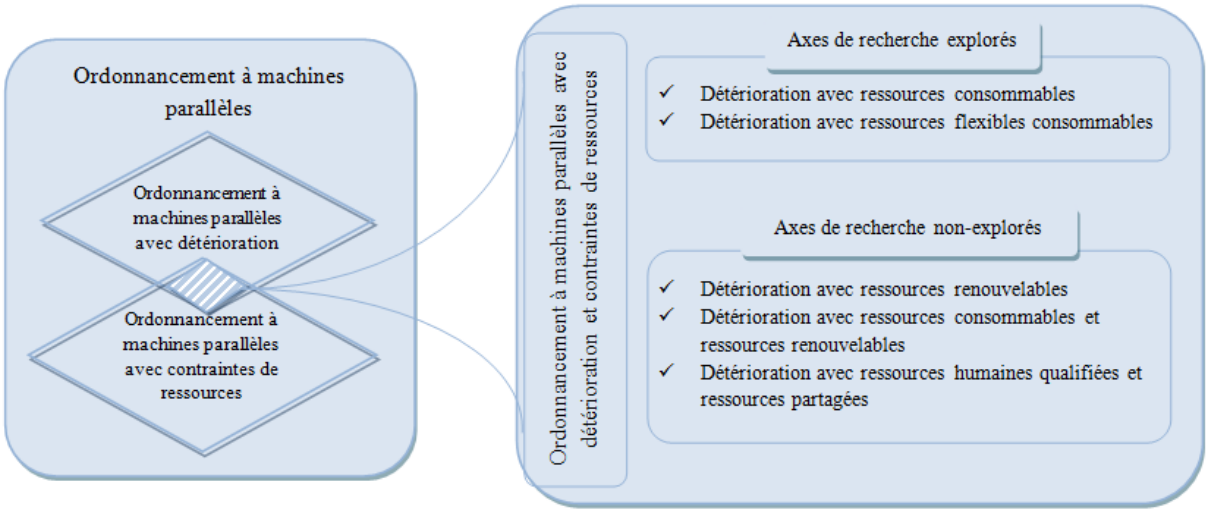


Figure 2.14 Les études manquantes dans le domaine d'ordonnancement à machines parallèles avec détérioration et considération de ressources.

1.6 Synthèse « Problématique »

A partir de cet état de l'art présenté, il a été remarqué que les fonctions de détérioration par positions, contrairement aux fonctions de détérioration par dates de début d'exécution, ne sont pas très considérées dans la littérature. Ainsi la plupart des fonctions de détérioration par position existantes dans la littérature sont des fonctions non-linéaires, alors que les fonctions par date de début d'exécution sont variées entre linéaire, non-linéaire et partiellement linéaire. Plusieurs manques aussi ont été aperçus en ce qui concerne les contraintes sur les ressources ; à commencer par la considération de plus qu'un seul type de ressource, qui n'a été abordée dans aucune étude de la littérature. Dans des cas de la vie réelle nous trouvons que

plusieurs ressources interactives interviennent dans l'exécution d'une tâche. Donc, l'investigation sur les problèmes d'ordonnancement avec différents types de ressources représente une étude contributive dans la littérature. Le deuxième manque aperçu est le manque de considération de l'effet de détérioration avec les ressources renouvelables dans le même problème, or que ce genre de problème est très répandu dans les systèmes manufacturiers. La figure 1.14 résume ces lacunes existantes dans la littérature, et présente les axes de recherches non-explorés, et sur lesquels notre étude est menée.

Après avoir encadré les lacunes constatées dans la littérature, nous présenterons dans les chapitres suivants ; chapitre III et IV, des études pour combler ces lacunes. La figure 1.15 positionne nos travaux par rapport aux travaux existants. Dans le chapitre III, nous mènerons une étude sur le problème de machines parallèles identique avec effet de détérioration et deux types de ressources consommables, dont l'une est une ressource flexible qui permet le contrôle du temps opératoire, l'autre est une ressource dont la consommation dépend du temps opératoire. L'objectif de l'étude étant de minimiser le makespan et les coûts des deux ressources. Dans le chapitre IV, nous introduirons la notion de l'efficacité énergétique, qui est devenue un sujet d'une importance cruciale étant donné de la consommation excessive de l'énergie sur l'environnement. Cependant, nous résoudrons le problème de minimisation du makespan et de la consommation totale de l'énergie dans un environnement de machines parallèles identiques sous contraintes de ressources humaines qualifiées et de ressource partagées.

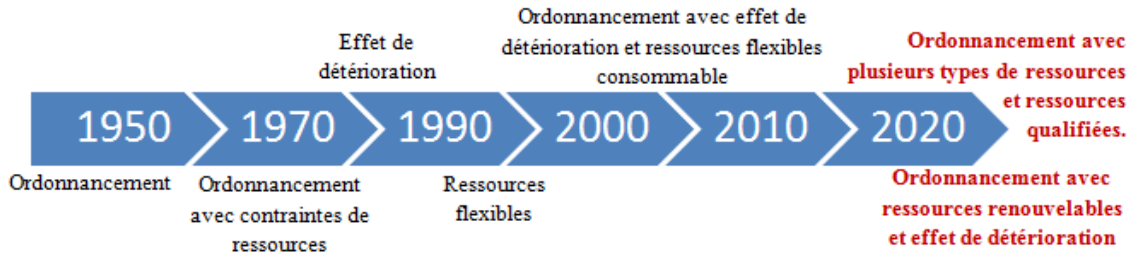


Figure 2.15 Les études de l'ordonnancement au fil des années

La résolution de problème classique d'ordonnancement à machines parallèles, consiste en l'affectation des tâches aux machines, tout en définissant les temps de début et de fin d'exécution. Cette simple procédure d'affectation a été prouvée d'être NP-difficile par (Lenstra, 1977), qui ont démontré l'appartenance de ce type de problèmes, même avec seulement deux machines, à la classe des problèmes d'optimisation NP-difficile. Il est à noter que dans ces problèmes la machine est la seule ressource, et la séquence des tâches importe peut, contrairement aux problèmes d'ordonnancement de machines parallèles avec considération de ressources qui se révèlent beaucoup plus complexes. Carlier and Rinnooy Kan, (1982) ont démontré qu'un problème d'ordonnancement à une seule machine avec ressources consommables sont NP-difficile au sens fort, lorsque les temps opératoire des tâches sont différents. Ainsi Blazewicz and Lenstra, (1983) ont prouvé que les problèmes

d'ordonnancement à machines parallèles avec ressources consommables sont NP-difficiles, lorsque le nombre de machines est supérieur à deux. À partir de là, nous pouvons conclure que les problèmes d'ordonnancement avec détérioration et plusieurs types de ressources qui vont être considérés et traités dans les prochains chapitres sont des problèmes NP-difficiles.

1.7 Conclusion

Dans ce chapitre, nous avons classé les problèmes d'ordonnancement en deux catégories :

- La catégorie des problèmes d'ordonnancement classique qui définissent les notions fondamentales de l'ordonnancement, ces problèmes servent au développement théorique de l'ordonnancement étant loin des vrais problèmes de l'industrie.
- La catégorie des problèmes d'ordonnancement actuels qui modélisent des problèmes plus réels. Ces derniers sont développés à partir des problèmes de l'ordonnancement classique, auxquels on a intégré des contraintes non-conventionnelles.

Dans ce contexte, nous avons défini les notions de bases de l'ordonnancement classique et ses contraintes, ensuite nous avons abordé deux contraintes non-conventionnelles qui sont la contrainte de ressources et celle de la détérioration.

Nous avons retracé l'état de l'art des problèmes d'ordonnancement de machines parallèles, sans et avec les différentes contraintes précédemment abordées « contraintes de ressources et de détérioration », dans le but de positionner notre problématique par rapport aux travaux existants.

Le chapitre suivant sera réservé aux différents types d'approches de résolution dédiées aux problèmes d'ordonnancement.

Chapitre 2

Techniques de résolution des problèmes d'ordonnancement

Résumé : Ce chapitre est dévolu à définir les différentes méthodes de résolution destinées aux problèmes d'ordonnancement. Ces derniers peuvent être résolus de deux manières; exacte ou approximative. En effet deux classes on été étudiées, la classe des méthodes exactes et celle des méthodes approximatives. Ensuite, quelques méthodes de résolution des problèmes d'ordonnancement multi-objectif seront présentées. Un autre type de méthodes sera présenté; c'est les méthodes qui décomposent le problème et le résolvent en plusieurs sous-problèmes. Finalement un état de l'art est formulé sur le développement et l'adaptation de ces méthodes aux problèmes d'ordonnancement mentionnés dans le premier chapitre.

Sommaire :

2.1 Introduction

Les problèmes d'ordonnancement mobilisent un grand nombre de chercheurs de différents domaines. Cette émulation est due à l'importance de la fonction de l'ordonnancement dans les systèmes manufacturiers. Or, l'efficacité d'un ordonnancement consiste, premièrement, en la modélisation des problèmes avec la considération de toutes ses contraintes, notions que nous avons abordées dans le premier chapitre. Deuxièmement, en l'implémentation d'une méthode efficace de résolution de ces mêmes problèmes, et c'est ce que nous aborderons lors de ce deuxième chapitre

La nature combinatoire des problèmes de l'ordonnancement fait que les méthodes de résolutions soient généralement issues de la discipline de la recherche opérationnelle, exactement de l'optimisation combinatoire. Plusieurs de ces méthodes et leurs adaptation aux problèmes existent dans la littérature, et sont classées selon plusieurs critère; exactitude, complexité, ...etc.

L'utilisation, d'une méthode quelconque, nécessite son adaptation au problème. Autrement dit, l'implémentation d'une méthode pour un problème d'ordonnancement classique n'est pas la même pour un problème avec contraintes de ressources ou/et de détérioration.

Dans ce deuxième chapitre nous présenterons, premièrement, les différentes méthodes de résolution, leur classification et leurs modes d'adaptation Et deuxièmement, nous présenterons les travaux qui ont adapté l'une des méthodes présentées à un problème d'ordonnancement classique, avec détérioration ou avec contraintes de ressources.

2.2 Classification des méthodes de résolution

La résolution d'un problème d'ordonnancement consiste en la définition d'une combinaison de paramètres (job, machines, ressources...etc.). La combinaison de la solution optimale peut être définie de façon mathématique, qui dit mathématique dit exacte, par la résolution des quelques équations, ou par l'élaboration d'un algorithme défini par quelques étapes, ce dernier pourra mener vers une solution exacte ou approximative. Dans ce contexte, les méthodes de résolution sont classées, selon la solution générée, en méthodes exactes ou approximatives, la figure 2.1 représente brièvement les différentes méthodes de résolution classiques; exactes et approximatives. Ces derniers ont prouvé leur efficacité dans la résolution de plusieurs types de problèmes d'ordonnancement. Cependant, et avec le développement technologique, les problèmes d'ordonnancement deviennent de plus en plus complexes, ce qui a fait que les méthodes de résolution classiques ont nécessité des améliorations pour plus d'efficacité. Dans la fin de cette section, nous présentons deux méthodes améliorées; la première est destinée aux problèmes multi-objectifs, la deuxième est destinée aux problèmes très complexes et qui peuvent être décomposés en plusieurs sous-problèmes.

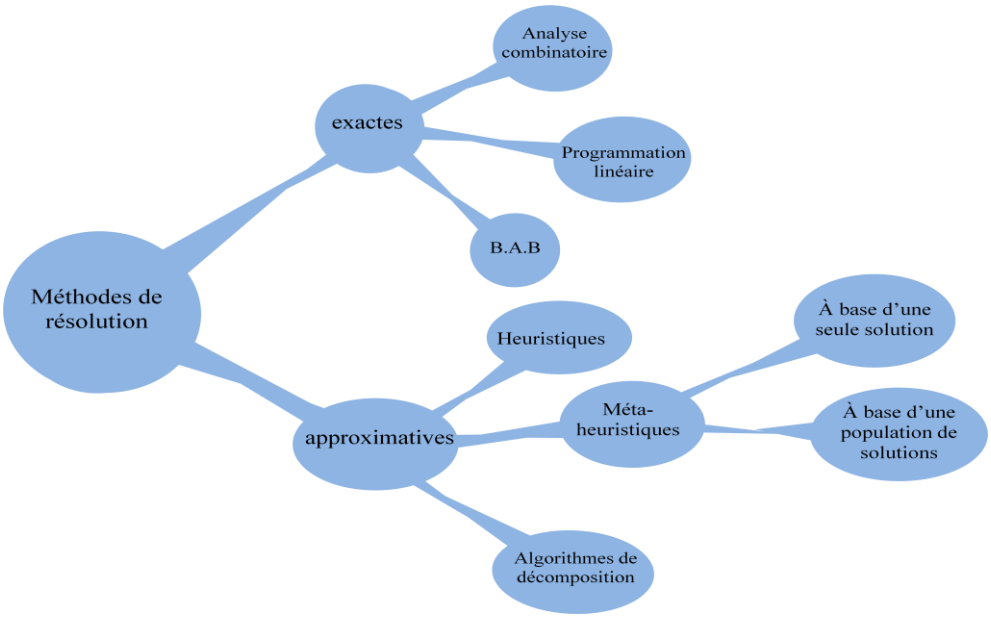


Figure 2.1 : Classification des différentes méthodes de résolution

2.2.1 Les méthodes de résolution exactes

2.2.1.1 *Analyse combinatoire*

L'analyse combinatoire, dans ce contexte, désigne toutes les méthodes analytiques utilisées dans l'élaboration d'algorithmes pour la résolution des problèmes combinatoires. L'analyse combinatoire étudie les différentes configurations des différents éléments d'un problème, basée sur une analyse profonde de celui-ci, de telle sorte que le comportement du modèle, qui modélise le problème doit être examiné pour le moindre changement. Cette méthode permet la connaissance de certaines propriétés du problème, qui peuvent être utilisées dans la conception d'un algorithme. Cette méthode sera utilisé dans le troisième chapitre, pour le développement d'un lemme, qui par son tour permet l'élaboration d'un algorithme de résolution.

2.2.1.2 *Programmation linéaire/non-linéaire*

C'est la méthode la plus évidente, décrite comme une méthode élégante et très flexible par (Kan, 1976). Elle consiste en la formulation de la fonction objectif et des contraintes par des équations d'égalité et/ou d'inégalité, dont la résolution se fait par un solveur. L'inconvénient de cette méthode réside dans le temps de calcul qui s'accroit de façon exponentielle en fonction des combinaisons.

2.2.1.3 *Séparation et évaluation*

Connu sous l'appellation de "branch and bound" BAB, c'est la méthode la plus utilisée dans la résolution des problèmes d'optimisation. La raison de sa popularité est la simplicité de son principe de fonctionnement basée sur une énumération

implicite et intelligente des certains sous-ensembles jugés plus pertinent que les autres. L'implémentation de cette méthode nécessite la connaissance des propriétés du problème pour pouvoir sélectionner les sous-ensembles à évaluer, ce qui n'est pas faisables dans tout les cas.

Les problèmes d'ordonnancement en générale et d'ordonnancement à machines parallèles en particulier, sont connus par leur difficulté, très souvent classés comme problèmes NP-difficile. Donc les méthodes exactes peuvent consommer d'énormes capacités de calcul, surtout lorsqu'il s'agit d'un problème d'optimisation de grande taille. Dans ce contexte, d'autres méthodes plus flexibles et moins complexes, ont été développées et adaptées aux problèmes d'ordonnancement, ce sont les méthodes approximatives, tel que les heuristiques et les métaheuristiques. La plupart des ces méthodes ont été développées pour la résolution de problème d'optimisation mono-objectif. Cependant, un grand nombre de problème d'ordonnancement sont modélisés par des modèles d'optimisation multi-objectif, ce qui a mené vers le développement de techniques qui permettent l'utilisation de certaines heuristiques ou métaheuristiques pour la résolution des problèmes d'ordonnancement multi-objectif.

2.2.2 Les méthodes approximatives

La résolution des problèmes d'ordonnancement par les méthodes mentionnées avant exige soit la connaissance des propriétés du problème ce qui est des fois très difficile ou même impossible à faire. Et Même si ses propriétés sont définies, la complexité du problème fait en sorte que trouver une solution, par une méthode exact, prendrait beaucoup de temps, à un point que la méthode devient inefficace. Sous ces circonstances, les méthodes approximatives, qui fournissent des solutions sous-optimales mais satisfaisantes, représentent une alternative très attirante. Parmi ces méthodes approximatives on cite :

2.2.2.1 *Les heuristiques*

Ce sont des méthodes développées pour résoudre des problèmes spécifiques, nécessitant une capacité de calcul raisonnable. En ordonnancement, une heuristique implique, généralement, la détermination de la ressource critique dont l'utilisation influence tout le système, et de la gérer de façon à minimiser son influence sur les autres ressources et sur la fonction objectif. En général, une heuristique est une suite d'étapes qui construit ou améliore une solution. Les règles de priorités sont des heuristiques constructives parmi les plus reconnues en ordonnancement, ce sont des prescriptions qui détermine l'ordre des tâches qui doivent être exécutées selon des priorités; comme par exemple:

- L'heuristique LPT (largest processing time): donne la priorité aux tâches ayant le temps opératoire le plus long, elle donne des solutions plus ou moins acceptable, souvent elle est utilisée dans les métaheuristique à une seule solution, pour définir la solution initiale lorsqu'il s'agit de minimiser le makespan.

- L'heuristique STP (shortest processing time): celle-ci, inversement à l'heuristique LPT, ordonne les tâches dans un ordre croissant des temps opératoire, elle a été proposée par (Smith, 1956) pour la résolution du problème de minimisation de la somme des temps de cycle (completion time) dans un environnement d'une seule machine $1||\sum C_j$. Elle peut être utilisée pour générer une solution initiale pour des métaheuristiques à une seule solution, lorsqu'il s agit d'un problème du type $1||\sum C_j$, avec des contraintes non-conventionnelles (Kaabi-harrath, 2004)
- L'heuristique EDD (earliest due date): cette heuristique, proposé par (Jackson, 1955), concerne les problèmes de minimisation d'un critère relatif au retard, elle prend en compte explicitement les dates de fin au plus tard, en séquençant les tâches dans un ordre croissant des dates de fin au plus tard (due date). Pour des problèmes 1||Lmax ou 1||Tmax, cette heuristique fournis des solutions optimales pour des problèmes du même type avec d'autre contraintes (Kaabi-harrath, 2004), elle peut servir de solution initiale à l'adaptation d'une métaheuristique.

Les heuristiques définies ci-dessus sont des heuristiques destinées à la résolution des problèmes classiques, d'autres heuristiques ont été développées pour faire face à des problèmes plus complexes comme les problèmes d'ordonnancement avec effet de détérioration, comme:

- L'heuristique LDR: une heuristique proposée par (Mosheiov, 1998). Inspirée de l'heuristique LPT, elle consiste à séquencer les tâches dans un ordre décroissant de leurs taux de détérioration.

Chacune des heuristiques existantes est spécifique à un problème précis, et ne peut pas être adaptée à un autre. En outre, la possibilité de développer une heuristique pour chaque problème d'ordonnancement n'est pas assurer, étant donné le nombre de paramètres nécessaire à identifier pour définir une solution, et la complexité des combinaisons formant une solution faisable. Pour ces raisons, des méthodes, quasiment indépendantes du problème et du degré de sa complexité, ont été développées, ce sont les métaheuristiques.

2.2.2.2 ***Les métaheuristiques***

Ce sont des méthodes de résolution destinées aux problèmes d'optimisation difficiles, qui ne peuvent être étudiés mathématiquement et résolus par des méthodes exactes. Étant donné la complexité des problèmes d'ordonnancement(Garey and Johnson, 1979), surtout ceux avec des contraintes non-conventionnelle, les méthodes exacte ne peuvent résoudre que les petites tailles de ces problèmes. Les heuristiques quant à elles, nécessitent une analyse combinatoire et des propriétés structurelles particulières qui ne peuvent toujours être définies. En conséquence, les métaheuristiques ont été développées; des méthodes qui ont souvent fait preuve de leur efficacité, et dont l'implémentation n'exige ni une analyse combinatoire ni

l'élaboration des propriétés structurelles du problème; Un exemple sur l'indépendance d'une des métaheuristiques est illustré dans la figure 2.1.

"Une métaheuristique est un ensemble de concepts utilisés pour définir une méthode de résolution qui peut être appliquée à un large éventail de différents problèmes. En d'autre terme, elle peut être vue comme un algorithme applicable à différents problèmes d'optimisation, avec relativement très peu de modification pour l'adapter à un problème spécifique."

G. Zâpfel et R. Braune, Meta-heuristic Search Concepts [], 2010

Les métaheuristiques sont des approches de résolution auxquelles on peut faire appel pour résoudre n'importe qu'elle problème d'optimisation NP-difficile. Elles sont devenues les techniques d'optimisation les plus étudies en raison de leurs capacités à fournir des solutions satisfaisantes en un temps de réponse abordables, ainsi que leurs aptitudes à gérer la complexité élevée des problèmes. Toutefois, la capacité de ces approches à faire face aux problèmes les plus complexes, sans trop se soucier du degré de complexité est contrebalancée par le fait qu'elles soient non-déterministes et ne donnent aucune garantie sur la qualité des solutions fournies.

Les techniques constituant une métaheuristique vont, pas toujours, mais très souvent de la simple procédure de recherche locale à des processus d'apprentissage plus complexes. Ces processus s'appuient sur un ensemble de principes, qui commence par la recherche d'une solution optimale ou presque optimale en explorant l'espace de toutes les solutions possibles, ensuite, assure une large exploration par des mécanismes qui permettent d'éviter d'être bloqué dans des régions de l'espace de recherche. Ainsi, parmi ces principes, il y a l'indépendance de l'algorithme de la métaheuristique de tout paramètre relatif au problème à résoudre, autrement dit, une métaheuristique peut être décrite de manière abstraite, sans faire appel au problème. Cependant, elle peut faire appel à des heuristiques qui tiennent compte de la spécificité du problème à résoudre.

De nombreuses métaheuristiques ont été développées au court des dernières décennies, chacune peut être meilleure que les autres dans la résolution d'un type de problème donné. Pour cette raison, nous donnons, dans la section suivante, une classification et une définition des métaheuristiques les plus utilisées dans la résolution des problèmes d'ordonnancement.

2.3 Classification des métaheuristiques

Plusieurs classifications des métaheuristiques existent ; une première classification est basée sur le fait que la métaheuristique soit inspirée d'un phénomène naturel ou non. Une deuxième classification, faite par (Zäpfel et al., 2010), divise les métaheuristiques en métaheuritsiques constructives et métaheuristiques amélioratrices.

- Métaheuristique constructives : sont des algorithmes qui construisent des solutions faisables en partant d'une solution initiale vide et en insérant, à

chaque étape, une composante dans la solution partielle courante, qui ne sera jamais améliorée par la suite. La majorité des méthodes constructives sont de type "glouton". A chaque étape, la solution courante est complétée de la meilleure façon possible. Les méthodes constructives se distinguent par leur rapidité et leur grande simplicité. On obtient en effet très rapidement une solution faisable pour un problème donné sans avoir recours à analyser ses propriétés. Cependant, le principal défaut de ces méthodes se trouve dans le fait que l'algorithme, à chaque étape, choisit la meilleure partie de la solution, en considérant seulement cette partie du problème, sans tenir compte des conséquences que cela engendre sur la solution finale ou complète, Dans ce sens, les méthodes de type glouton sont souvent considérées comme myopes. Il est donc judicieux, dans le cas général, de mettre au point des procédures anticipant les effets secondaires et les conséquences futures occasionnées par les décisions prises lors de la construction d'une solution faisable, (WIDMER, 2001). Parmi les métaheuristiques constructives, nous détaillerons, dans ce qui suivra, l'algorithme du glouton et l'algorithme des colonies de fourmis.

- Métaheuristiques amélioratrices : sont des algorithmes itératifs qui explorent l'espace des solutions possibles en déplaçant, à chaque itération, la solution courante vers une autre meilleure. Une métaheuristique de ce type débute à partir d'une solution s_0 choisie aléatoirement ou alors par le biais d'une méthode constructive. Le passage d'une solution faisable à une autre se fait sur la base d'un ensemble de modifications élémentaires. Une solution voisine s' obtenue à partir de s en appliquant une modification élémentaire. Le voisinage N(s) d'une solution s est défini comme l'ensemble des solutions faisables atteignables depuis s en effectuant une modification élémentaire. Un tel processus d'exploration est interrompu lorsqu'un ou plusieurs critères d'arrêt sont satisfaits, (WIDMER, 2001).

Une troisième classification est faite sur les métaheuristiques, elle les divise en métaheuristiques à une seule solution, qui sont basées sur la manipulation et l'amélioration d'une seule solution, tel que la recherche Tabou ou le recuit simulé et métaheuristiques à population, qui sont basées sur la manipulation de toute une population de solutions, tel que les algorithmes génétiques. Ces deux types de métaheuristiques sont complémentaire, celles à solution unique sont orientée exploitation, elles ont le pouvoir d'intensifier la recherche dans des régions locales, quant aux métaheuristiques à population, elles sont orientées plus exploration, elles permettent une meilleure diversification des solutions.

2.3.1 Métaheuristique constructive, à une seule solution « algorithme glouton »

Dans un problème d'optimisation tel que les problèmes d'ordonnancement, une solution est définie par un ensemble d'éléments E= $\{e_1, e_2, e_3, \ldots e_n\}$. Une solution partielle peut être vue comme l'ensemble E'= $\{e_1, e_2, e_3, \ldots e_k\}$ tel que $k<n$ et $E' \subset E$. Dans un algorithme glouton, l'ensemble définissant une solution initiale

doit être vide, et à chaque étape une heuristique locale est utilisée pour affectée un nouvel élément à la solution partielle. Une fois un élément e_i est affecté à une position de l'ensemble, il n'est jamais déplacé par la suite, autrement dit, une décision prise, quelle qu'elle soit n'est jamais remise en question,(Talbi, 2009). La conception d'un algorithme glouton passe par les deux étapes suivantes :

- La définition de l'ensemble des éléments : cette étape consiste en la formulation ou la conception de la forme de la solution du problème à résoudre et la définition de tous ses éléments.
- L'heuristique locale pour la sélection d'élément : selon le contexte du problème, une heuristique est nécessaire à chaque étape de la construction. Cette heuristique doit sélectionner le meilleur élément de la liste courante, en termes de ses implications dans la fonction objectif.

Les algorithmes gloutons sont des méthodes connues par leur simplicité ; ils sont d'un degré de complexité inférieur à celui des algorithmes itératifs. Toutefois, l'importance partielle que portent ces algorithmes aux problèmes d'optimisation, décroit leurs performances en termes de qualité de solution, (Talbi, 2009).

2.3.2 Métaheuristique constructive, à population « colonie de fourmis »

Le principe des algorithmes des colonies de fourmis (ACO) est d'imiter le comportement coopératif dans l'organisation des fourmis, pour résoudre de vrais problèmes d'optimisation. Ces métaheuristiques sont très efficaces dans la résolution de plusieurs problèmes d'optimisation combinatoire.

L'intérêt principal du comportement d'une fourmi apparait dans son habiliter à trouver le chemin le plus court pour transporter la nourriture de sa source à la fourmilière, en coopérant avec les autres individus de la colonie. Pour faire, la fourmi laisse une trace chimique « phéromone » sur le chemin d'où elle est passée, cette phéromone est une substance olfactive et volatile, son rôle est de guider les autre fourmis sur le point cible. Plus la quantité de phéromone est grande, plus la probabilité qu'une fourmi choisit le chemin est grande. Une fourmi choisit un chemin en fonction de la quantité de phéromone qui s'y trouve, (Talbi, 2009).

Chaque agent-fourmis possède les caractéristiques suivantes :

- La fourmi laisse derrière elle une quantité de phéromone lorsqu'elle se déplace d'un élément i à un autre élément j.
- Elle choisit l'élément de destination suivant une probabilité, qui est en fonction de la distance entre cet élément et sa position et la quantité de phéromone qui s'y retrouve.
- Elle est dotée d'une mémoire, qui lui permet d'éviter un passage répétitif sur le même élément, (Mommarché, 2000).

2.3.3 Métaheuristiques amélioratrices, à une seule solution

Le principe de ces méthodes consiste à définir une solution initiale, aléatoirement ou en utilisant une des heuristiques connues, et de procéder à la recherche d'une autre solution qui améliore la fonction objectif. La recherche d'une nouvelle solution se fait dans le voisinage de la solution initiale, on parle de méthodes de voisinage ou de recherche locale, (Gunther et al., n.d.).

Ces méthodes de voisinage, par un processus itératif, décident du remplacement de la solution initiale ou courante par une autre de son voisinage, si celle-ci a une fonction objectif meilleure, ou si elle répond à d'autre critère relatif à la métaheuristique. L'arrêt de l'algorithme peut dépendre du nombre des itérations, d'un paramètre interne, ou d'une valeur de la fonction objectif atteinte.

Ces métaheuristiques sont classées, généralement, par la façon dont elle parcourt le voisinage ou sélectionne la solution. Parmi les métaheuristiques à une seule solution les plus connues, nous détaillerons dans ce qui suit la recherche tabou et le recuit simulé.

2.3.3.1 *Recherche Tabou*

Formulée par (Glover, 1986), la recherche Tabou est une stratégie de recherche locale qui repose sur trois principes: i)- la recherche de la meilleure solution, à partir d'une seule solution, en effectuant un balayage sur l'ensemble des solutions possibles. ii)-l'enregistrement des meilleures solutions dans une mémoire, pour éviter de tourner dans le même cycle. iii)-la possibilité de choisir une solution moins bonne. (Dréo, 2006)

Les éléments essentiels de cette méthode sont:

- Une solution initiale.
- Une technique de génération de solutions voisines.
- Une liste Tabou dans laquelle une solution choisisse doit être enregistrée.
- Un critère d'arrêt.

La méthode consiste à générer, à partir de la solution initiale, une solution voisine qui ne sera acceptée que si elle améliore la fonction objectif. Une fois la solution acceptée, elle est enregistrée dans une liste appelée Liste Tabou, cette liste nous permet de détecter les optimums locaux. Ces étapes sont répétées jusqu'à ce que l'algorithme atteigne un optimum local, et donc le choix d'une solution moins bonne est permis, pour échapper aux impasses. Les étapes se répètent jusqu'à l'atteinte du critère d'arrêt.

2.3.3.2 *Recuit simulé*

Le recuit simulé est une métaheuristique très populaire, basées sur le principe de la recherche locale. Proposée par (S. Kirkpatrick et al., 1983), le recuit simulé a

été inspiré du principe de refroidissement des métaux. Son pseudo-code est représenté dans l'algorithme 1.

Tel que la recherche Tabou, le recuit simulé repose sur trois principe: i)-La recherche de la meilleure solution, à partir d'une seule, en balayant l'ensemble des solutions possibles. ii)- la possibilité d'accepter une solution moins bonne, à condition que sa probabilité de mener vers une solution meilleure soit acceptable. iii)-la diminution lente de la température à chaque itération. Ces principes sont démontrés dans la figure 2.2, illustrant l'avantage de l'indépendance de la majorité des étapes de cette métaheuristique du problème. (Dréo, 2006)

Les éléments essentiels à l'implémentation de cette métaheuristique sont:

- Une solution initiale
- Une technique de génération de solutions voisines (ou de recherche locale)
- Une température initiale T
- Un taux de refroidissement
- Une probabilité d'acceptation Pa
- Un critère d'arrêt

Par un mouvement bien défini, effectué sur la solution initiale ou courante, l'algorithme génère une autre solution « solution voisine », dont la qualité dépend de sa fonction objectif. Si la valeur de la fonction objectif E_{i+1} de la solution voisine S_{i+1} est meilleure que celle de la solution couranteE_i, la solution voisine S_{i+1} serait considérée comme solution pour la prochaine itération. Sinon, la probabilité que cette solution mène vers une solution meilleure est évaluée, la probabilité est calculée par l'équation (2.1).

$$P_i = e^{\frac{\Delta E_i}{T_i}} \tag{2.1}$$

Si cette probabilité calculée est plus grande que la probabilité d'acceptation définie au début, la solution voisine S_{i+1} serait considérée comme solution initiale de la prochaine itération. Sinon, S_{i+1}serait rejetée, et la solution courante serait reconsidérée pour la prochaine itération.

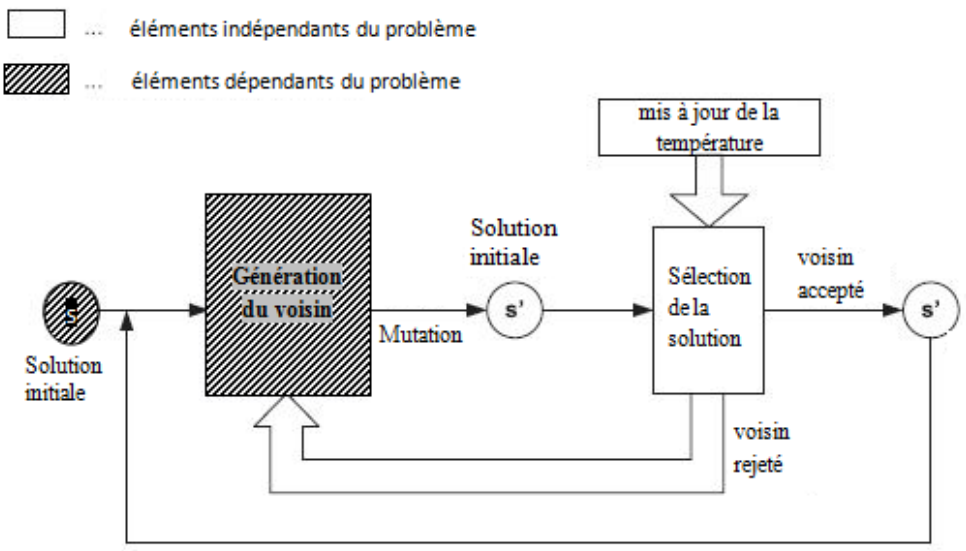


Figure 2.2 : Le principe de fonctionnement du recuit simulé avec ses différents éléments, (Zäpfel et al., 2010).

Algorithme 1: Pseudo-code du recuit simulé

Initialiser (solution initiale s, température initiale T_0, Probabilité d'acceptation Pa, taux de refroidissement α, critère d'arrêt)
Tant que: critère d'arrêt pas atteint, répéter:
    générer une solution voisine s'
    Comparer entre s et s'
    Si E (s')<E (s) ou P (s')>Pa:
        remplacer s par s' pour la prochaine itération
    Sinon:
        rejeter s' et garder s pour la prochaine itération
    Fin
    Fin
    Mettre à jour T et i.
Fin

2.3.4 Métaheuristique amélioratrice, à population « algorithme génétique »

Il existe un autre type de métaheuristiques améliorateurs, qu'on appelle les algorithmes génétiques. Ces algorithmes, au lieu de chercher à améliorer progressivement une seule solution (comme les algorithmes de la recherche locale), ils améliorent à la fois plusieurs solutions, appelées population. L'amélioration d'une population se fait globalement par les techniques de reproduction, basées sur la sélection et la mutation de quelques éléments.

Les algorithmes génétiques ont été inspirés de la théorie de l'évolution. Ils simulent l'évolution naturelle des gènes des individus, prenant en considération le phénomène de l'hérédité et la loi de survie. La loi de la sélection naturelle des gènes impose la survie des individus les mieux adaptés à l'environnement (les plus robustes). Cependant, la simulation de ce phénomène en un algorithme consiste à

considérer différentes solution possibles comme des individus, dont la robustesse dépend de la qualité, autrement dit, de la valeur de la fonction objectif. (Kaabi-harrath, 2004)

Le fonctionnement de ces algorithmes repose sur les principes suivants: i)-la sélection de certaines individus sur lesquels le processus de mutation est appliqué. ii)- l'évaluation de la fonction objectif pour chaque individu. iii)-la sélection des individus à prendre pour la prochaine itération. (Zäpfel et al., 2010)

Étant donné la flexibilité et l'efficacité des métaheuristiques, à solution unique ou à population, l'utilisation de ces méthodes est devenue essentielle pour l'étude et la résolution des problèmes d'ordonnancement. Par ailleurs, l'aspect mono-objectif des métaheuristiques a fait qu'elles ne conviennent pas à certains problèmes dont l'objectif est multicritère. Et donc, des versions de métaheuristiques multi-objectif ont été développées.

2.3.5 Les métaheuristiques multi-objectif

L'ordonnancement d'un atelier a pour but d'assurer la meilleure exploitation de toutes les ressources de l'atelier sans risque d'explosion des coûts de production. Cela dit, l'ordonnancement doit assurer un bon équilibre entre l'exploitation des ressources et l'augmentation des coûts. Lorsque l'objectif de l'ordonnancement doit se ramener à un seul critère, il y a un risque qu'un autre critère négligé ait des conséquences indésirables sur l'efficacité de l'ordonnancement. D'où la notion de l'ordonnancement multi-objectif, qui permet des degrés de liberté dans la modélisation du problème, excluant tout risques de déviation de la modélisation. Plusieurs méthodes d'optimisation multi-objectif existent (Collette and Siarry, 2002); parmi ces méthodes, on cite la méthode scalaire de pondération qui est utilisée dans certaines métaheuristiques multi-objectif, et la méthode exploitant le recuit simulé MOSA (multi objectif simulated annealing).

2.3.5.1 *La méthode scalaire de pondération*

C'est la méthode la plus évidente. Le but de cette méthode est de revenir à un problème d'optimisation mono-objectif, en appliquant un coefficient de pondération w_i, sur un ensemble de fonctions f.

Si on avait n fonctions à optimiser$f_1, f_2 \dots f_i$, la fonction finale F après pondération est présentée dans l'équation (2.2)

$$F = \sum_{i=1}^{n} w_i . f_i \tag{2.2}$$

Les coefficients de pondérations doivent respecter la relation suivante:

$$\sum_{i=1}^{n} w_i = 1$$

2.3.5.2 *La méthode du recuit simulé multi-objectif*

Proposé par (Ulungu et al., 1999), elle utilise les mêmes étapes du recuit simulé pour la recherche de la surface de compromis, la seule différence réside dans l'évaluation de la probabilité π d'une mauvaise solution.

On considère qu'il y n fonctions à optimiser, simultanément. La solution du problème E est donnée par un ensemble de solutions, E_i, i=1...n, chacune appartient à sa fonction, la probabilité$\pi_i$de chaque solution est une fonction deΔE_i qui est la différence entre la solution courant et la nouvelle solution, et de$T_k$qui est la température dans la k^{ieme} itération. Elle est représentée par l'équation (2.3)

$$\pi_i = \begin{cases} e^{\frac{\Delta E_i}{T_k}} \\ 1 \end{cases} \quad (2.3)$$

La probabilité de la solution finale E, peut être calculée de deux façons:

i)- Par le produit de toutes les probabilités π_i: la probabilité de la solution est définie par le produit de toutes les probabilités pondérées par le poids λ_i de chaque fonction i. Équation (2.4).

$$P(E) = \prod_{i=1}^{n} (\pi_i)^{\lambda_i} \quad (2.4)$$

ii)-Par la sélection de la plus petite probabilité. Équation (2.5).

$$P(E) = min(\pi_i)^{\lambda_i} \quad (2.5)$$

La sélection de la solution et les conditions d'acceptation sont les mêmes du recuit simulé classique.

De nombreuses autres métaheuristiques multi-objectif existent dans la litérature, comme par exemple la métaheuristique PASA (pareto archived simulated annealing), la recherche Tabou multi objectif et les algorithmes génétiques multi-objectifs: VEGA, MOGA et NSGA II. Plus de détails sur ces métaheuristiques peuvent être trouvés dans le livre (Collette and Siarry, 2002).

Toutes ces méthodes définies auparavant, ont été utilisées par les chercheurs pour la résolution des problèmes d'ordonnancement. Chaque méthode peut être plus efficace dans la résolution de certains types de problèmes plus que d'autres, selon les objectifs et les paramètres de celui-ci. Cependant, lorsqu'il s'agit d'un problème d'optimisation très complexe, comme les problèmes de l'ordonnancement actuels, l'implémentation de ces méthodes classiques peut devenir très compliquée et des fois même inefficace. Cela est en raison, premièrement, des solutions infaisables sur lesquelles on pourrait tomber lors de la recherche d'une solution meilleure.

Deuxièmement, du domaine des solutions qui est très large, et dans lequel l'algorithme peut s'égarer. L'exemple illustrant cela, est l'exemple du problème d'ordonnancement avec effet de détérioration et consommation de ressources ; pour résoudre ce problème, une métaheuristique classique doit trouver au même temps l'affectation adéquate des tâches aux machines, leurs séquences et l'allocation adéquate des ressources aux tâches, une combinaison de trois éléments, dont chacun appartient à un domaine assez large. Partant de cette analyse, et en vue de procurer des méthodes de résolutions efficaces et moins complexes, des approches de décomposition ont été développées et très souvent utilisées dans la résolution de problèmes d'optimisation décomposables.

2.3.6 Algorithmes de décomposition

Un algorithme de décomposition est une méthode de résolution visant avant tout à trouver les sous-problèmes qui composent le problème principale ensuite les séquencer et finalement les résoudre. Théoriquement, un algorithme de décomposition fournit des solutions (approximatives ou exactes) pour des problèmes complexes, en les décomposant en plusieurs petits sous-problèmes plus faciles à résoudre (Ovacik and Uzsoy, 1997a). Une solution donnée par un algorithme de décomposition est généralement construite de façon hiérarchique, autrement dit, le premier sous-problème est résolu indépendamment, mais la résolution du deuxième sous-problème dépend de la solution du premier, et ainsi de suite. Par conséquent, l'ordre des sous-problèmes impacte sur la qualité de la solution finale. Niu et al., (2012) ont comparé entre deux approches de décomposition d'un problème d'affectation et de séquençage, tel que la première débute par l'affectation de tâches et la deuxième débute par leurs séquençage.

Ces approches approximatives; heuristiques et métaheuristiques, représentent une excellente alternative aux méthodes exactes, mais elles fournissent des solutions approximatives, dont la qualité de la solution reste à vérifier. Donc, le développement d'une approche de résolution approximative doit impérativement être suivi d'une analyse de ses performances.

2.3.7 Les performances d'une métaheuristique

Les métaheuristiques, en tant que méthodes de résolution approximatives, sont basées principalement sur des techniques de recherches ou de sélection soit aléatoires soit intuitives, auxquelles dépend l'efficacité de l'algorithme, et donc la qualité de la solution obtenue. Par conséquent, le paramétrage des techniques de recherche et/ou de sélection est l'un des facteurs assurant l'efficacité de l'algorithme. Dans plusieurs études adaptant une métaheuristique à un problème d'ordonnancement, le paramétrage de l'algorithme se fait en exécutant l'algorithme sous différentes combinaisons des paramètres, et en comparant les solutions données par l'algorithme sous ces combinaisons avec la solution optimale. La meilleure combinaison est donc celle qui procure les solutions les plus proches de l'optimum.

Après le paramétrage de la métaheuristique, une analyse de ses performances est primordiale à son évaluation. Cette analyse consiste en trois étapes:

- Structuration des expérimentations: définir l'ensemble des expériences, les instances sur lesquelles les expériences sont faites.
- Évaluation des résultats obtenus par rapport à la qualité des solutions, le temps de calcul; le temps nécessaire à la métaheuristique pour résoudre le problème.
- Analyse statistique d'étudier l'effet des différents facteurs, du problème ou de la métaheuristique, sur son efficacité, (Belkaid, 2014).

2.4 Revue de la littérature sur les approches de résolution utilisées en ordonnancement

La plupart des études des problèmes d'ordonnancement consistent en la formulation mathématique du problème en un modèle de programmation linéaire pour une résolution exacte, ensuite l'adaptation d'une méthode de résolution approximative. Certains papiers se contentent d'une analyse combinatoire du problème, et définissent ses propriétés, et des fois des lemmes, qui permettent le développement d'une méthode BAB ou un autre heuristique. Rocha et al., (2008) a proposé un modèle mathématique en programmation linéaire, et ensuite il a développé un algorithme BAB pour la minimisation du makespan dans un problème d'ordonnancement à machines parallèles. Gharehgozli et al., (2009) ont proposé un modèle de programmation en nombre entier mixte pour un problème d'ordonnancement à machines parallèles.

2.4.1 Approches de résolution pour les problèmes d'ordonnancement

Les problèmes d'ordonnancement à machines parallèles classiques sont des problèmes NP-difficiles, de telles sortes qu'aucune méthode de résolution exacte ne peut les résoudre de façon efficace. Plusieurs méthodes approximatives ont été développées dans la littérature; Chudak, (1999) a proposé un algorithme rapide basé sur l'algorithme LIST. Min and Cheng, (1999) aussi ont proposé un algorithme génétique pour la minimisation du makespan des machines parallèles identiques. Kim et al., (2002) ont adapté le recuit simulé pour résoudre le problème de minimisation de la somme des retards dans un environnement de machines parallèles non-uniformes. il a utilisé l'heuristique EDD pour générer la solution initiale. Lee et al., (2006) ont adapté le recuit simulé à un problème de minimisation du makespan dans un environnement de machines parallèles, il a utilisé l'heuristique LPT pour générer la solution initiale. Low, (2005) a développé un recuit simulé pour un problème d'ordonnancement de flow shop hybride avec machines parallèles non uniformes. Wang and Alidaee, (2019) ont proposé un algorithme pour résoudre les grandes instances du problème de la minimisation de la somme des temps de cycles pondérés dans un environnement de machines parallèles.

Lorsqu'il s'agit de problèmes d'ordonnancement multi-objectifs, NSGA II est la métaheuristique la plus populaire; (Wang et al., 2018b), (Souier et al., 2019) et (Boufellouh and Belkaid, 2020). D'autres métaheuristiques aussi ont été développées pour les problèmes multi-objectifs ; Varadharajan and Rajendran, (2005) et Mokhtari et al., (2011) ont développé le recuit simulé multi objectif (MOSA) pour résoudre un problème d'ordonnancement multi-objectif de flow shop.

2.4.2 Approches de résolution pour les problèmes d'ordonnancement à machines parallèles avec détérioration

Après l'étude réalisée par (Browne and Yechiali, 1990), de nombreux chercheurs se sont intéressés aux problèmes d'ordonnancement avec effet de détérioration, ou plus généralement avec temps opératoire variable. En effet, divers théorèmes et algorithmes ont été proposés, selon le problème. Wu and Lee, (2003b) ont proposé et prouvé, à travers une analyse combinatoire, deux théorèmes pour la résolution d'un problème de minimisation de makespan dans un environnement de machine unique avec détérioration et contraintes d'indisponibilité. Wang, (2007) a proposé plusieurs théorèmes pour différents problème d'ordonnancement de machine unique détérioration linéaire des tâches (problème de minimisation de makespan et de minimisation des retards), à travers ces théorèmes il a développé des algorithmes pour chaque problème. Toksarı and Güner, (2009) ont proposé un modèle mathématique pour minimisation des retards et latences de machines parallèles avec effet d'apprentissage et détérioration (linéaire et non-linéaire), il a prouvé la propriété V-shape du problème; cette propriété assure l'optimalité si les tâches qui se terminent avant la date due sont ordonnancées en ordre décroissant des temps opératoires Pj, et les tâches qui se termine après la date due sont ordonnancées en ordre croissant des Pj. à travers cette propriété un algorithme V-shape exacte a été développé. Huang and Wang, (2011) en faisant appel à un lemme mathématique, ils ont proposé deux autres lemmes, le premier pour minimiser le TADC (la somme des différences absolues en temps de cycle), et le deuxième pour minimiser le TADW (la somme des différences absolues des temps d'attente). Pour la minimisation du makespan dans un environnement de machines parallèles avec détérioration des machines. Liu et al., (2013) ont proposé et prouvé des théorèmes sur la complexité du problème de minimisation du makespan dans un environnement de machines parallèles, il a proposé la fameuse heuristique LDR (ordre décroissant des tâches selon leurs taux de détérioration). Ouazene and Yalaoui, (2017), en s'inspirant des travaux de (Liu et al., 2013), ont proposé un algorithme exacte TMDO (Two Machines Deteriorating jobs Optimal algorithm) pour résoudre le problème de deux machines parallèles avec détérioration. Ensuite, il l'a étendu à un algorithme (extended-TMDO) pour résoudre un problème plus général de plusieurs machines parallèles avec détérioration.

Tous ces théorèmes et ces algorithmes cités ci-dessus, sont destinés à des problèmes spécifiques. Ils deviennent inefficaces au moindre changement dans les paramètres du problème. Pour cela, plusieurs papiers ont choisi de développer des

métaheuristiques, qui sont plus flexibles, pour la résolution des problèmes considérés. Mazdeh et al., (2010) ont proposé un modèle mathématique de programmation linéaire et ensuite a développé une recherche Tabou pour résoudre le problème de minimisation du makespan et des coûts de détérioration dans un environnement de machines parallèles. Ruiz-Torres et al., (2013) ont proposé un modèle mathématique de programmation linéaire. Ensuite, il a formulé quelques lemmes pour un cas spécifique. Et finalement, il a conçu un algorithme de résolution à base de recuit simulé. Chung and Kim, (2016) ont proposé un modèle de programmation linéaire en nombre entier mixte pour minimiser la somme des temps de cycles et des temps des activités de maintenance dans une machine unique avec détérioration et maintenance. Il a proposé un algorithme génétique avec une heuristique qui s'occupe de la répartition des tâches dans le chromosome. Tigane et al., (2018) ont proposé un algorithme NSGA II pour résoudre le problème multi-objectif de minimisation du makespan et du TEC (énergie consommée totale) dans un environnement de machines parallèles non-uniformes avec détérioration de tâches. Soleimani et al., (2020) ont proposé un modèle mathématique de programmation linéaire pour formuler le problème de minimisation des retards pondérés et de l'énergie consommée dans un environnement de machines parallèles non-uniformes avec détérioration et effet d'apprentissage. Ensuite, il a développé un algorithme génétique, un CSO (cat swarm optimizer) et IABC (interactive artificial bee colony) et a comparé entre eux en termes d'efficacité.

2.4.3 Aperçu sur les approches de résolution pour les problèmes d'ordonnancement à machines parallèles avec contraintes de ressources

Les problèmes d'ordonnancement avec contraintes de ressources sont plus proches de la réalité, et représentent un grand nombre de cas réels. Cependant, le taux de complexité élevé de ces problèmes fait que leur résolution nécessite le développement d'algorithmes approximatifs. Etant donnée leur popularité, plusieurs algorithmes ont été développés spécifiquement pour ces problèmes. Edis et al., (2013) ont révisé la complexité des problèmes d'ordonnancement avec contraintes de ressources ainsi les méthodes de résolution développées pour la résolution de ces problèmes.

Özpeynirci et al., (2016) ont proposé un modèle mathématique pour un problème de minimisation de makespan dans un environnement de machines parallèles avec contraintes de disponibilité d'outils. Il a aussi développé une recherche Tabou pour la résolution des grandes instances du problème. Fanjul-Peyro et al., (2017) ont proposé un modèle mathématique originale pour formuler le problème de minimisation du makespan de machines parallèles non-uniformes avec contraintes de ressources additionnelles. Le modèle considère les tâches comme des rectangles dont la longueur est le temps opératoire, la largeur est la consommation de ressources. Ces rectangles doivent se contenir dans un emballage rectangle dont la largeur est limitée par la contrainte de disponibilité de ressources, et la longueur est à minimiser. Bien que l'efficacité de ce modèle dépasse celle des autres modèles

existants, mais la taille du problème qu'il puisse résoudre reste limité. Ils ont donc proposé des matheuristique; une combinaison entre les techniques de métaheuristiques et du modèle mathématique. Dans le même contexte de la résolution de problème d'ordonnancement à machines parallèles non-uniformes, Villa et al., (2018) ont proposé quelques heuristiques constructives et a développé une recherche locale pour l'amélioration des solutions construites. Abdeljaoued et al., (2018) ont proposé une heuristique et a développé un recuit simulé pour améliorer les résultats de l'heuristique, cela pour résoudre un problème d'ordonnancement de machines parallèles avec ressources renouvelables qualifiées.

Györgyi and Kis, (2017) ont proposé plusieurs théorèmes pour les problèmes d'ordonnancement de machines parallèles avec ressources non-renouvelables. Belkaid et al., (2016b) ont proposé un modèle mathématique de programmation linéaire pour formuler le problème de minimisation du makespan de machines parallèles avec allocation dynamique de ressources consommables. Ensuite, il a développé un algorithme génétique pour résoudre le problème.

pour les problèmes de ressources flexibles Daniels Richard L. et al., (1996) ont proposé une heuristique SBA (static-based algorithm) cette heuristique a été améliorée par (Daniels et al., 1997) en lui intégrant une recherche Tabou. Edis and Oguz, (2012) ont révisé les modèles mathématiques de programmation linéaire des problèmes de minimisation du makepsan, spécifiés et non-spécifié, dans les machines parallèles avec ressources flexibles. Ensuite, il a proposé un modèle de relaxation de contraintes IP pour résoudre les problèmes de grandes instances. Low and Wu, (2016) ont proposé un modèle mathématique de programmation linéaire pour formuler le problème d'ordonnancement à machines parallèles avec temps opératoires contrôlables. Ensuite, il a développé deux algorithmes basés sur la métaheuristique de colonie de fourmis. Le premier consiste à résoudre simultanément le problème d'affectation et d'allocation. Le deuxième consiste en l'affectation des tâches, tout en définissant le makespan, ensuite l'allocation de façon à réduire le makespan. Sekkal and Belkaid, (2020) ont développé un recuit simulé pour la résolution d'un problème multi-objectif de minimisation du makespan et du coût des ressources, avec détérioration et temps opératoire de tâches dépendent de la quantité de ressource y allouée.

2.4.4 Aperçu sur les approches de décomposition dédiées aux problèmes d'ordonnancement

Du fait qu'elles simplifient la résolution du problème, les approches de décomposition sont très efficaces lorsqu'il s'agit de résoudre un problème d'ordonnancement avec contraintes de ressources. Daniels et al., (1999) ont développé une heuristique de décomposition basée sur l'heuristique SBH de (Daniels Richard L. et al., 1996) (static-based-heuristic) pour résoudre le problème d'ordonnancement de machines parallèles non-spécifié avec contrainte de ressources flexibles. Su and Lien, (2009) ont proposé deux heuristiques de décomposition pour le même problème de minimisation du makespan dans un environnement de

machines parallèles avec ressources flexibles. La première heuristique consiste en l'affectation des tâches ensuite l'allocation des ressources. La deuxième consiste en l'allocation des ressources ensuite l'affectation des tâches. Mokhtari et al., (2011) ont développé une approche de décomposition pour résoudre un problème multi objectif dans un flow shop avec ressources flexibles. L'approche consiste en deux sous-problème: i)-sous-problème de séquençage des tâches dans les machines. ii)-sous-problème d'allocation de ressources aux tâches. Aussi l'heuristique proposé par (Villa et al., 2018) peut être classée avec les heuristique de décomposition, vue qu'elle construit la solution en deux étapes: i)-L'ordonnancement des tâches sans considération de ressources. ii)-La construction d'une solution avec considération des ressources, à partir de l'ordre des tâches défini lors de la première étape.

À travers cette revue de la littérature, nous avons remarqué un manque d'originalité dans les modèles mathématique de programmation linéaire, la plus part des études utilise le même principe pour la modélisation des contraintes de disponibilité de ressources. Ensuite, nous avons constaté que, malgré sa simplicité et son efficacité, le recuit simulé multi-objectif est rarement utilisé dans la résolution des problèmes d'ordonnancement, nous avons aussi constaté que peu d'études ont utilisé des lemmes pour la résolution d'un sous-problème dans un algorithme de décomposition.

Pour pallier à ce manque, et dans le but de résoudre les problèmes considérés, nous allons proposer, dans les chapitres suivants, des modèles mathématiques de programmation linéaire, dont un est original. Nous allons, ensuite, développer des algorithmes de recuit simulé multi-objectif et des algorithmes de décomposition.

2.5 Conclusion

Dans ce chapitre, nous avons exposé les différentes méthodes de résolution d'un problème d'optimisation. En se focalisant sur les méthodes destinées à la résolution des problèmes d'ordonnancement, nous avons classées ces méthodes de résolution en des méthodes exactes qui permettent d'étudier et d'analyser le problème, et des méthodes approximatives qui résolvent le problème de façon efficace.

Un intérêt particulier a été porté aux méthodes, utilisées dans cette thèse, dans la résolution des problèmes mentionnés dans le premier chapitre. Ces méthodes sont: La programmation linéaire, l'analyse combinatoire, le recuit simulé, méthode multi-objectif et les approches de décomposition.

Ensuite, une revue de la littérature sur l'adaptation de ces différentes méthodes aux problèmes d'ordonnancement introduits dans le premier chapitre a été présentée. et les méthodes de résolution utilisées dans cette étude ont été définies.

Dans le chapitre suivant, nous allons étudier un problème d'ordonnancement à machines parallèles avec plusieurs types de ressources et effets de détérioration. Pour ce problème, nous allons proposer un modèle mathématique, développer un

lemme, un algorithme de recuit simulé multi-objectif "MOSA" et un autre algorithme de décomposition basé sur le MOSA et le lemme proposé.

Chapitre 3

Problème d'ordonnancement à machines parallèles sous des contraintes de ressources

Résumé : Dans ce chapitre nous étudions un problème d'ordonnancement à machines parallèles identiques avec effet de détérioration et deux différents types de ressources. Dans une première partie de l'étude de ce problème, nous commençons par sa modélisation mathématique en un modèle de programmation linéaire, suivi d'une analyse combinatoire. Dans une deuxième partie, nous développons de deux méthodes de résolution approximatives, pour résoudre efficacement le problème d'ordonnancement considéré. Les performances de ces méthodes approximatives sont analysées et comparés aux performances du modèle mathématique.

Sommaire :

Introduction

Étant un outil de prise de décision, l'ordonnancement est appelé dans tous les secteurs industriels ; des entreprises fournissant un produit tangible ou des services. Le modèle d'ordonnancement des ateliers à machines parallèles est le plus répandu dans le monde industriel. Ces ateliers se trouvent dans les ateliers de menuiserie, de peinture, dans les hôpitaux, les aéroports, les flowshop flexibles comme les limonadières flexibles….etc.

Dans le but d'améliorer le rendement d'un atelier et de faire face aux exigences des clients, l'ordonnancement à pour objectif d'optimiser l'exploitation des ressources en prenant en considération tous les paramètres et les contraintes du système de production, tel que la disponibilité des machines, le temps opératoire des tâches…etc. Dans les études des problèmes de l'ordonnancement classique, le temps opératoire d'une tâche a été considéré comme constant et indépendant de tout autre paramètre, or qu'en réalité il peut être sujet de plusieurs variations tel que l'effet de détérioration.

L'effet de détérioration apparait dans les activités de maintenance, qui prennent plus de temps si elles ne sont pas exécutées à leurs dates prévues. Il apparait lors de l'utilisation d'une quelconque ressource (machine, outil, ressource humaine) qui prenne de l'âge et perd en performances à chaque utilisation. La détérioration est un effet indésirable pour les industriels, du fait qu'elle augmente le temps opératoire d'une tâche, créant une hausse dans le temps de production total et dans la consommation de certains ressources comme l'électricité, le fuel, les lubrifiants…etc. Dans certain cas, il existe des ressources qui peuvent remédier à cette indésirable hausse. Ces ressources, appelées les ressources flexibles, détiennent un contrôle sur le temps opératoire et peuvent le réduire à un certain pourcentage. La consommation de ces dernières ressources, comme la consommation de toutes autres ressources, génère des coûts. Dans ce contexte, nous traitons dans ce chapitre un problème de minimisation du makespan et des coûts de ressources dans un environnement de machines parallèles identiques avec effet de détérioration et deux types de ressources, dont la consommation de l'une contrôle le temps opératoire, et la consommation de l'autre est contrôlée par le temps opératoire. Ces ressources ont des impacts inverses ; la première est une ressource consommable flexible, dont l'augmentation de la quantité consommée diminue le temps opératoire, la deuxième est une ressource consommable dont la quantité consommée est une fonction du temps opératoire. Cette relation contradictoire entre ces deux types de ressources n'a pas été considérée dans la littérature. Cependant, pour mieux illustrer la relation entre ces deux types de ressources et l'effet de détérioration, nous projetons le problème considéré sur des cas réels fournissons des exemples de cas réel, parmi ceux deux exemples d'ateliers de production de biens, et un autre atelier de production de services. Les deux ateliers produisant des biens sont : ateliers de fabrication de béton préfabriqué, et atelier de ponçage de carrelage. L'atelier

produisant des services et un cabinet dentaire, plus exactement sur le processus de détartrage dentaire.

- **Le premier exemple** : Ateliers de fabrication du béton préfabriqué

Dans un atelier de béton préfabriqué, plusieurs chaines de production fonctionnent en parallèle, ces chaines sont considérées comme des machines parallèles identiques. Après la fabrication d'un panneau de béton, la machine se détériore ralentissant le temps opératoire de la tâche suivante. Dans ce cas, le temps opératoire d'une tâche dépend de sa séquence dans la machine. D'autre part, le temps pour fabriquer un panneau de béton peut être réduit par l'utilisation d'adjuvant (accélérateur, super-plastifiant) ; Les accélérateurs réduit le temps de séchage du panneau et donc l'énergie consommée pour le séchage, et les super-plastifiants augmentent la maniabilité du béton et le rendent plus facile à étaler, et donc ils réduisent l'énergie consommée par la table vibrante qui s'occupe de l'étalage du béton. Il est important d'équilibrer entre la consommation des adjuvants et celle de l'énergie pour une meilleure exploitation de l'atelier.

- **Le deuxième exemple** : Atelier de ponçage du carrelage

Dans une usine de production de carrelage, la dernière étape dans la chaine de production, qui est l'étape du ponçage, comprend plusieurs machines à poncer identiques et qui fonctionnent en parallèle. La vitesse du processus de ponçage dépend du flux de l'eau fournie à la machine ponceuse ; de plus grandes quantités d'eau par unités de temps réduisent le temps de ponçage. D'autre part, la machine ponceuse consomme de l'énergie durant l'exécution d'une tâche. En outre, le disque de ponçage de la machine ponceuse se détériore après l'exécution de chaque tâche, ralentissant l'exécution de la tâche suivante. Il est remarqué que la séquence de la tâche dans la machine et la quantité de la ressource d'eau qui lui est allouée contrôle le temps opératoire d'une tâche, et le temps opératoire contrôle la consommation en énergie, ces deux ressources (eau et énergie) peuvent coûter chère à l'entreprise, il est donc important d'équilibrer entre leurs consommations.

- **Le troisième exemple** : Processus de détartrage dentaire

Dans une clinique dentaire, où plusieurs chaises dentaires fonctionnent en parallèle, le temps pour effectuer un détartrage à un patient dépend de la quantité de l'agent détartrant consommée. Et à cause de la fatigue du dentiste, ce temps dépend aussi du nombre de patients traités avant. (Li and Wang, 2016) a traité un problème d'ordonnancement des traitements d'urgence avec effet de détérioration et temps opératoire contrôlable.

L'objectif de ce chapitre est d'étudier le problème considéré et de le résoudre efficacement. L'étude et la résolution de ce problème font appel aux outils introduits dans le premier et deuxième chapitres, et sont constituées de trois étapes: i)-étape de la modélisation mathématique par un modèle de programmation linéaire, ii)-ensuite l'étape de l'analyse combinatoire du modèle mathématique proposé, iii)-et finalement l'étape de la résolution du problème par deux méthodes approximatives différentes

qui sont le recuit simulé multi-objectif « MOSA » et un algorithme de décomposition qui utilise le MOSA et les résultats de l'analyse combinatoire.

Ce chapitre est divisé comme suit : la prochaine section s'occupe de la description du problème, la troisième section définie les détails du modèle mathématique formulé. La section 4 présente une analyse combinatoire du modèle proposé. La section 5 présente un algorithme de recuit simulé multi-objectif qui est adapté au problème pour le résoudre. La section 6 présente un autre algorithme, cette fois un algorithme de décomposition. La section 7 résume les résultats d'une analyse de sensibilité pour ces deux algorithmes, ainsi qu'une comparaison entre les performances du modèle mathématique, du premier et du deuxième algorithme.

3.1 Description du problème d'ordonnancement considéré

Le problème d'ordonnancement considéré est représenté par un système de N tâches indépendantes à exécuter dans l'une des M machines parallèles et identiques, chaque tâche pour son exécution nécessite une quantité de ressources, qu'on appelle quantité basique, du premier typeR_1, et consomme une quantité d'un autre type de ressourcesR_2. L'allocation du premier type de ressources à une tâche peut varier de la quantité basique à une quantité supérieure, cette variation influence sur le temps opératoire de la tâche. Autrement dit, plus la quantité de ressourcesR_1allouée à une tâche augmente plus le temps opératoire de celle-ci diminue. En considérant l'effet de la détérioration, qui fait que le temps opératoire augmente en dépendance de la séquence de la tâche dans la machine, le temps opératoire d'une tâche n'est plus constant, mais variable. Inspiré de (Chen, 2004), le temps opératoire d'une tâche, dans notre cas, est formulé par l'équation (3.1). Quant à la consommation du deuxième type de ressourcesrb_{2j}est représentée par l'équation (3.2). Le temps opératoire est une fonction linéaire dépendante de la séquence de la tâche, autrement dit, de sa position dans la machine, et de la quantité de ressourcesR_1qui lui est allouées, et la fonction de rb_{2j}est linéairement indépendante du temps opératoire.

$$Pr_{jp} = a_j + \alpha(p-1) - \beta.(r_{1j} - rb_{1j}) \tag{3.1}$$

Tel que :

a_j Le temps opératoire normal d'une tâche j

α et β Taux de détérioration, et les unités de temps réduites par unité de ressourcesR_1.

rb_{1j} Les quantités basique de R_1 ; nécessaire à l'exécution d'une tâche j.

r_{1j} Les quantités allouées à la tâche j.

La consommation du deuxième type de ressource rb_{2j}aussi est formulée par l'équation (3.2), qui est une fonction dépendante du temps opératoire :

$$rb_{2j} = \gamma.a_j + \gamma'.(Pr_{jp} - a_j) \tag{3.2}$$

Tel que

γ et γ' : Unités de ressourcesR_2 consommées par unité de temps.

3.1.1 Les hypothèses

Pour bien encadré le problème à résoudre, quelques hypothèses et notations importantes ont été posées:

- Les machines sont identiques ; les temps opératoire ne changent pas en fonction des machines.
- Une machine exécute une seule tâche à la fois
- Les tâches ne sont pas préemptives, chacune doit être exécutée par une seule position d'une machine.
- Le temps opératoire d'une tâche est une fonction linéairement dépendante de la position de la tâche et de la quantité deR_1qui lui est allouée.
- A cause de la détérioration la machine peut consommer plus de ressourcesR_2, et c'est pourquoi γ'>γ.
- Les taux α et β sont communs à toutes les tâches.
- Les quantités de ressources $R_1 \, et \, R_2$ sont disponibles en quantités suffisantes.

3.1.2 Les objectifs

L'objectif principal de ce chapitre est de résoudre le problème considéré de façon efficiente, c'est à dire de trouver des solutions satisfaisantes en un temps de calcul acceptable. Cependant, la résolution de ce problème consiste en l'affectation des tâches aux différentes machines et l'allocation de la quantité adéquate des ressources R_1 à chaque tâche, de façon à ce que deux critères doivent être optimisés; F1 qui est le temps de production total "makespan", et F2 qui est le coût des deux ressources R_1et R_2.

3.1.3 Les contraintes

Les seules contraintes à considérer c'est les contraintes de capacité de la machine. Quant aux ressources, elles sont toutes les deux disponibles en quantités suffisantes.

Cependant, il est important de noter que la plupart des études qui ont considéré un temps opératoire dépendant de la quantité de ressources allouée, ont posé une borne supérieure à la fonction. Sinon le temps opératoire d'une tâche convergera vers une valeur nulle ou négative, et le modèle sera faussé.

Dans notre cas, où la fonction objectif vise à minimiser le makespan et le coût des deux ressources, le solveur visera à équilibrer entre le temps opératoire et entre la quantité de ressources. Mais dans le cas où les décideurs favorisent la minimisation du makespan à celle des coûts, le solveur choisira probablement de maximiser la consommation des ressources pour réduire le makespan à 0. Pour éviter ce conflit, nous supposons que les ressources allouées, quelque soient leurs

quantités, ne peuvent réduire plus qu'un pourcentage donné A% du temps opératoire. Mathématiquement, cela est représenté par l'équation (3.3).

$$a_j * A\% > \beta * r_{1j} \rightarrow \frac{a_j}{\beta} * \mathrm{A}\% > r_{1j} \tag{3.3}$$

3.2 Modélisation mathématique "programmation linéaire du problème"

L'étude d'un problème d'ordonnancement consiste principalement sur sa modélisation mathématique, qui, à partir des données du système, formule les contraintes imposées et les objectifs à atteindre sous forme de paramètres que nos appelons « sorties ». La programmation linéaire est l'un des outils mathématiques les plus appelés pour faire face à un problème d'ordonnancement. Ainsi pour le développement d'un modèle de programmation linéaire, plusieurs paramètres doivent être définis, par ceux les données ; qui définissent le système où se trouve le problème étudié, les sorties ; qui eux sont les paramètres à définir par une solution donnée, et finalement les variables de décision ; qui sont contrôler par les contraintes et grâce auxquelles le programme linéaire converge vers les solutions optimales. Dans les trois sous-sections suivantes, les données, sorties et les variables de décision utilisées pour formuler notre modèle mathématique seront présentées.

3.2.1 Les données

Ce sont les paramètres introduits par les preneurs de décisions. Les données qui définissent notre système sont :

N	Le nombre de tâche j à traiter
M	Le nombre de machines k disponibles
a_j	Le temps opératoire normal de la tâche j
α	Le taux de détérioration
β	Le temps réduit par unité de ressource
γ et γ'	La quantité de ressources consommée par unité de temps, γ lorsqu'il s'agit du temps opératoire normal, et γ' lorsqu'il s'agit du temps opératoire ajouté par l'effet de détérioration
A%	Le pourcentage du temps opératoire normal que les ressources R_1peuvent réduire
rb_{1j}	La quantité basique des ressourcesR_1, la quantité nécessaire à l'exécution de la tâche
$Cu_1 et Cu_2$	Le coût unitaire des ressourcesR_1 et R_2, respectivement.

3.2.2 Les sorties

Représentent les paramètres ou les combinaisons définis par le modèle mathématique, et qui formulent une solution donnée. Les sorties à définir par notre modèle mathématique sont les suivants :

Pr_{pk}	Le temps opératoire normal dans la position p de la machine k, traitant une tâche quelconque.
r_{1pk}	La quantité de ressourcesR_1allouée à la position p de la machine k.
r_{2pk}	La quantité de ressourcesR_2consommée par la position p de la machine k.
$Cr_1 et Cr_2$	Le coût total des ressources consommées de$R_1\ et\ R_2$, respectivement.
C_k	Le temps de production total pour une machine k.
C_{max}	Le temps de production total de la dernière machine.

3.2.3 Les variables de décision

La modélisation de ce problème nécessite une variable de décision élémentaire, et une autre secondaire. La variable de décision élémentaire X_{jpk} est la variable de position de la tâche; celle-ci indique l'affectation de la tâche à la position adéquate de la machine adéquate. Cette variable permet aussi de définir la séquence de la tâche et donc de calculer la détérioration qu'elle subisse. La variable secondaire Y_{pk}nous indique si la position p de la machine k est occupée par une tâche ou pas.

$$X_{jpk} = \begin{cases} 1\ si\ la\ tâche\ j\ est\ affectée\ à\ la\ position\ p\ de\ la\ machine\ k \\ 0\ sinon \end{cases}$$

$$Y_{pk} = \begin{cases} 1\ si\ la\ position\ p\ de\ la\ machine\ k\ est\ occupée\ par\ une\ tâche \\ 0\ sinon \end{cases}$$

3.2.4 Les objectifs

Comme ça a été présenté dans la section 3.2.2, l'objectif à minimiser lors de la résolution du problème considéré est formulé par les deux critères qui sont le makespan et le coût des ressources. Ces deux critères sont modélisés par les équations (3.4a) et (3.5b), respectivement.

$$\text{Min F1= } C_{max} \tag{3.4a}$$

$$\text{Min F2=} Cr_1 + Cr_2 \tag{3.4b}$$

Tel que la première fonction F1 est représentée par les équations (3.14) et (3.15). Et la deuxième fonction F2, qui est définie par la somme de Cr_1 et Cr_2, est représentée par les équations (3.12) et (3.13), respectivement.

3.2.5 Les contraintes

La contrainte de la capacité des machines est représentée par l'équation (3.5). L'équation (3.6) assure que la tâche n'est pas préemptive, et donc elle doit être traitée dans une seule position par une seule machine. L'équation (3.7) assure qu'il n'y ait pas de position lacune dans la machine, autrement dit, une position n'est occupée que lorsque sa précédente est occupée. La contrainte sur la consommation des ressourcesR_1est représentée par les deux équations (3.8) et (3.9). L'équation (3.10) définit le temps opératoire actuel d'une tâche j affectée à une position p d'une machine k. l'équation (3.11) définit la quantité des ressourcesR_2 consommée.

$$\sum_{j=1}^{N} X_{jpk} \leq 1 \qquad \text{p=1…P, k=1…M} \qquad (3.5)$$

$$\sum_{k=1}^{M}\sum_{p=1}^{N} X_{jpk} = 1 \qquad \text{j=1…N} \qquad (3.6)$$

$$X_{jpk} \leq \sum_{j=1}^{N} X_{j(p-1)k} \qquad \text{p=1…P, k=1…M} \qquad (3.7)$$

$$r_{1pk} \geq \sum_{j=1}^{N} X_{jpk} \times rb_{1j} \qquad \text{p=1…P, k=1…M} \qquad (3.8)$$

$$\sum_{j=1}^{N} \frac{X_{jpk} a_j}{\beta} * A\% > r_{1pk} \qquad \text{p=1…P, k=1…M} \qquad (3.9)$$

$$Pr_{pk} = \sum_{j=1}^{N} X_{jpk} a_j - \beta(r_{1pk} - \sum_{j=1}^{N} X_{jpk} \times rb_{1j}) + \alpha(p-1) \qquad \text{p=1…P, k=1…M} \qquad (3.10)$$

$$rb_{2pk} = \gamma(\sum_{j=1}^{N} X_{jpk} a_j - \beta(r_{1pk} - \sum_{j=1}^{N} X_{jpk} \times rb_{1j})) + \gamma'(\alpha(p-1)) \qquad \text{p=1…P, k=1…M} \qquad (3.11)$$

$$Cr_1 = \sum_{k=1}^{M}\sum_{p=1}^{N} r_{1pk} * Cu_1 \qquad (3.12)$$

$$Cr_2 = \sum_{k=1}^{M}\sum_{p=1}^{N} rb_{2pk} * Cu_2 \qquad (3.13)$$

$$C_k = \sum_{p=1}^{P} Pr_{pk} \qquad \text{k=1…M} \qquad (3.14)$$

$$Cmax \geq C_k \qquad \text{k=1…M} \qquad (3.15)$$

3.2.6 Approche de l'agrégation des fonctions

La transformation du problème multi-objectif à un problème mono-objectif peut faire appel à plusieurs approches, tel que : Approche de l'agrégation, Approche des Métriques pondérés, Approches des ε-contraintes…etc. Ces approches sont utilisées pour générer les solutions optimales de Pareto (Talbi, 2009).

L'approche de l'agrégation est l'une des approches les plus utilisées. Elle combine, linéairement, entre les différentes fonctionsf_i pour former une seule fonction agrégée F, représentée par l'équation (3.16).

$$f(x) = \sum_{i=0}^{n} w_i . f_i(x) \qquad (3.16)$$

Tel que $w_i$représentent les poids de chaque fonction f_i.

Pour illustré l'importance de considérer une fonction multi-objectif (minimiser F1 et F2), le problème est résolu en minimisant seulement F1 « makespan », ensuite il est résolu en minimisant seulement F2 « les coûts », finalement, il est résolu en minimisant les deux fonctions F1 et F2 agrégées.

- Modèle 1 : minimisation du makespan seulement « F1 ».
- Modèle 2 : minimisation des coûts seulement « F2 ».
- Modèle 3 : minimisation du makespan et des coûts agrégés $F = w_1 . F1 + w_2 . F2$.

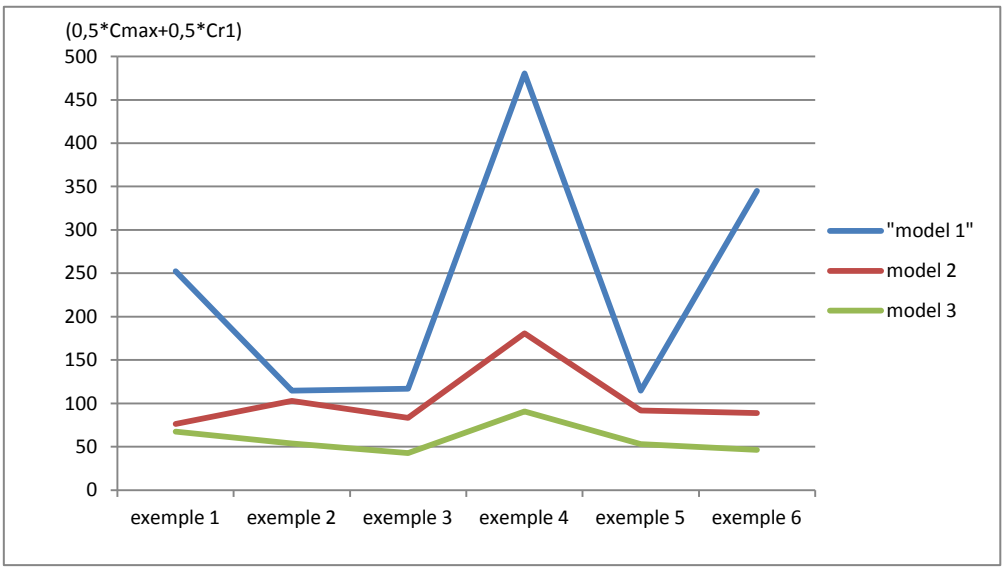


Figure 3.1 : Evaluation des performances de la fonction d'agrégation

Pour chaque modèle, la somme pondérée de F1 et F2 est calculée, les résultats sont représentés dans la figure 3.1. L'approche de l'agrégation nous permet de positionner chaque objectif par rapport aux autres, tenant compte de son importance et son influence sur le système. La figure 3.1 montre que lorsque le premier modèle est considéré, le makespan est minimisé tandis que le coût des

ressourcesR_1explose. Et lorsque le deuxième modèle est considéré, les coûts sont minimisés mais le makespan augmente inutilement. Ce n'est que lorsque le troisième modèle est considéré, que les deux fonctions F1 et F2 sont équilibrées.

Exemple 3.1: Pour illustrer l'impact des ressourcesR_1sur le modèle, un exemple de 7 tâches, 3 machines a été considéré, les paramètres sont définis comme suit : α=1, β=1, γ=1.3, γ'=1.8. Les temps opératoire et les quantités basiques deR_1sont définis dans le tableau 3.1.

Tableau 3.1 Instances de l'exemple 3.1

Job	**J_1**	**J_2**	**J_3**	**J_4**	**J_5**	**J_6**	**J_7**
a_j	3	20	3	3	10	2	3
rb_j	1	2	1	1	2	1	1

La figure 3.2(a) représente la solution du problème avec la consommation minimale des ressources R_1, la figure 3.2(b) représente la solution du problème avec deux unités additionnelles de R_1. Les deux solutions sont définies par les diagrammes de Gantt de chaque machine, le graphe de la consommation de R_1 et le graphe de la consommation de R_2, aussi pour chacune des trois machines. En comparant entre la figure 3.2(a) et 3.2(b), nous remarquons une diminution de deux unités de temps dans le makespan sans changer l'affectation ou la séquence des tâches, et par conséquent, une diminution de 2.6 unités dans le coût des ressources R_2. Ces diminutions sont dues à la consommation d'unités additionnelles de ressources R_1.

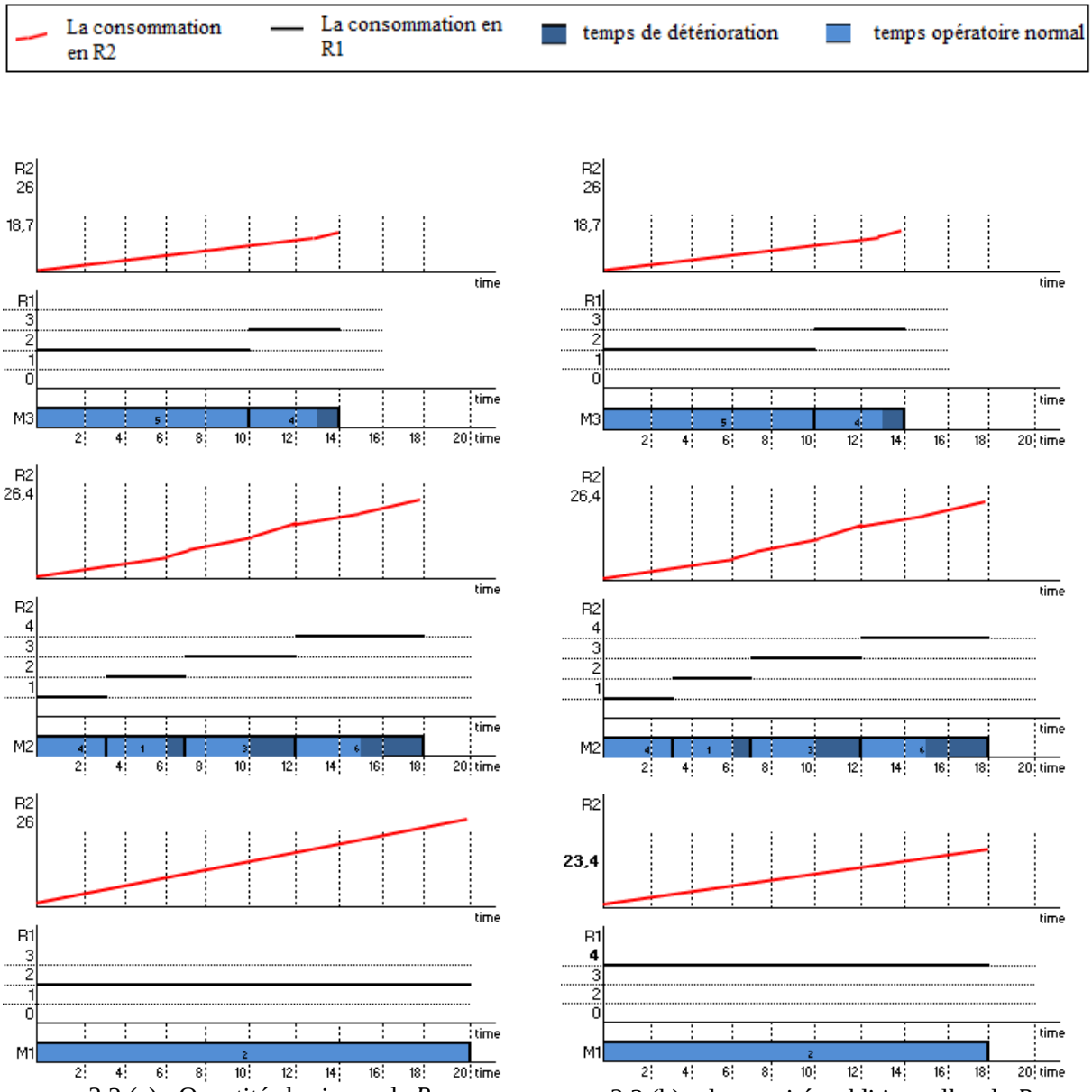
3.3 Analyse combinatoire du modèle

L'analyse combinatoire est classée parmi les méthodes de résolution exacte des problèmes d'ordonnancement. Elle permet une meilleure compréhension du comportement du model sous la variation de ses différents paramètres. Pour cela, nous allons dans cette section effectuer une analyse numérique et ensuite une analyse théorique du modèle.

3.3.1 Analyse Numérique

Par le mot numérique, nous voulons dire que l'analyse se fera à travers les résultats de plusieurs expériences. Ces expériences consistent en la résolution du problème sous différentes valeurs de ces paramètres. Donc nous prenons un exemple de petite taille, pour pouvoir le résoudre de façon optimale en utilisant le modèle mathématique et Lingo10 comme solveur. L'exemple est défini par 5 tâches à exécuter dans 3 machines. Les données fixes sont résumées dans le tableau 3.2. Les autres données sont variées d'une expérience à une autre. Pour mieux analyser notre problème, des expériences sont menées pour deux différents scénarios, les expériences pour le premier scénario ne prennent pas en considération les ressourcesR_2, ni leur minimisation. Les expériences pour le deuxième scénario

prennent en considération les ressources R_2, et leur coût comme critère à minimiser. Le but de traiter ces deux scénarios différemment est d'analyser l'influence de chaque ressource sur le comportement du modèle mathématique proposé.



3.2 (a) : Quantités basiques de R_1

3.2 (b) : deux unités additionnelles de R_1

3.2 : L'influence de la consommation de R_1 sur le makespan et le coût des R_2

Tableau 3.2 Le temps opératoire normal j et les quantités nécessaires deR_1.

Job	**J_1**	**J_2**	**J_3**	**J_4**	**J_5**
a_j	15	5	4	4	10
rb_j	1	1	1	1	1

Les expériences du scénario 1: consistent en la minimisation du makespan et seulement du coût des ressources R_1,$E_1 = w_1.C_{max} + w_2.Cr_1$. Lors de chaque

expérience de ce type, nous varions les paramètres comme suit:$(w_1, w_2) = \{(0.3, 0.7), (0.5, 0.5), (0.7, 0.3)\}$, pour chaque paire de poids, β varie comme suit : β={0.5, 1, 2}. Nous avons opté pour la variation de ces paramètres ; $w_1, w_2 et\ \beta$, tout en négligeant le coût des ressources R_2, dans le but d'étudier l'effet des ressourcesR_1 sur le système, et aussi pour confirmer si la négligence du coût deR_2 influence ou non sur le résultat final. La remise en question de l'importance de minimiser Cr_2 est venue d'une hypothèse intuitive qui dit que " les ressources R_2 sont directement liées aux temps opératoires, et vue que la minimisation du makespan minimise les temps opératoire, il est évident que même le coût des ces ressources R_2 sera minimisé." Cette hypothèse sera confirmée ou infirmée par l'analyse des résultats des expériences résumées dans le tableau 3.3.

Tableau 3.3 Résultats des simulations en minimisant E_1

(w_1, w_2)	(0.3,0.7)			(0.5,0.5)			(0.7,0.3)		
B	0.5	1	2	0.5	1	2	0.5	1	2
C_{max}	**15**	**15**	**15**	**15**	**15**	**10**	**14.5**	**12.5**	**8.5**
Cr_1	**5**	**5**	**5**	**5**	**5**	**10**	**6**	**10**	**12**
Cr_2	**39.5**	**40.25**	**39.5**	**39.5**	**39.5**	**24.5**	**38.75**	**32**	**18.8**
E_1	8	8	8	10	10	10	11.95	11.75	9.55
$E_2 = E_1 + w_2 Cr_2$	35.65	36.17	35.65	29.75	29.75	22.25	23.57	21.35	15.19

La première remarque est que le solveur alloue plus de ressourcesR_1 lorsque $w_1 > w_2$, parce que la minimisation du makespan importe plus que la minimisation duCr_1, donc plus de ressources sont consommées pour réduire les temps opératoires. Il a aussi été remarqué que lorsque le temps réduit par unité de ressource « β » augmente, la quantité de ressourcesR_1 allouée augmente. Cela s'explique par le fait qu'une seule unité de ressourceR_1 réduit plusieurs unités de temps, surtout lorsque le poids du makespan est plus grand que celui du coût, l'augmentation des coûts, dans la fonction objectif, sera négligeable par rapport à la réduction du makespan engendrée. Finalement, dans les expériences où (w_1, w_2) = (0.3, 0.7), malgré la variation du β, aucune différence entre les fonctions objectif n'a été remarquée. Par ailleurs, nous avons remarqué que la fonction du coût des R_2 peut augmenter inutilement de façon aléatoire, figure 3.3 illustre cette possibilité. Tout d'abord nous rappelons la fonction du Cr_2dans l'équation (3.17)

$$Cr_2 = Cu_2 \sum_{j=1}^{N} (\gamma(a_j - \beta.r_{1j}) + \gamma'(\alpha(p_j - 1))) \qquad (3.17)$$

Tel que pj est la position dans laquelle la tache j est exécutée. La première partie de la somme « $\gamma(a_j - \beta.r_{1j})$ » ne peut être réduite que par l'ajout de quantités supplémentaires des ressources R_1, alors que la deuxième partie de la somme « $\gamma'(\alpha(p_j - 1))$ » peut être réduite en affectant le même nombre de tâches à chaque machine. Dans la figure 3.4 il y a deux différents ordonnancement, en calculant le Cr_2 de l'ordonnancement de la figure 3.3(a) et 3.3(b), nous trouvons deux valeurs différentes, bien que le makespan et leCr_1 restent les mêmes:

Premier ordonnancement, figure 3.3(a):

$C_1 = 15,\ C_2 = 13 + 3.\alpha\ et\ C_3 = 10$

Le coût des ressources R_2est donc défini comme suit :

$$Cr_2(a) = \gamma.\sum_j a_j + \gamma'.3.\alpha = 38 + 3.\alpha$$

Deuxième ordonnancement, figure 3.3(b):

$C_1 = 15,\ C_2 = 9 + \alpha\ et\ C_3 = 14 + \alpha$

Le coût des ressourcesR_2est donc défini comme suit :

$$Cr_2(a) = \gamma.\sum_j a_j + \gamma'.3.\alpha = 38 + 2.\alpha$$

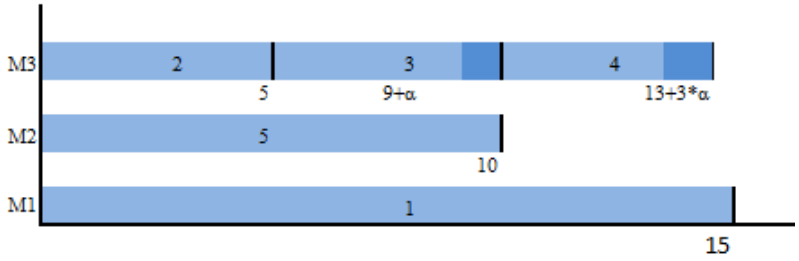


Figure 3.3 (a) Ordonnancement minimisant E1

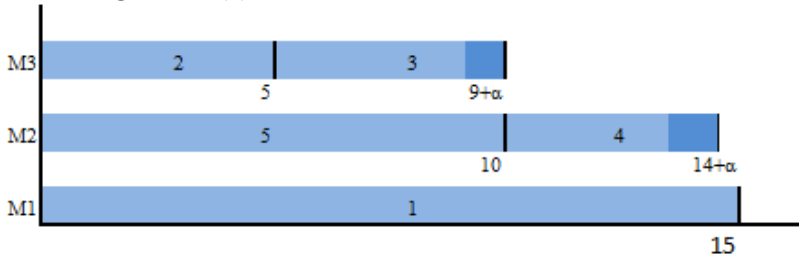


Figure 3.3 (b) Ordonnancement minimisant E2

Figure 3.3 : L'impact de l'affectation des tâches sur le coût de R_2

Lorsque la tâche J_4 est réaffectée de la machine 3 à la machine 2, la somme de la détérioration décroit de 3.α à 2.α. Quelque soit la valeur de α, Si $13 + 3.\alpha \leq 15$et donc$14 + \alpha \leq 15$, l'ordonnancement (b) garde la même valeur du makespan en minimisant le Cr_2.

A partir de là, nous rejetons l'hypothèse intuitive posée sur l'importance de minimiser le coût de R_2, et nous reconsidérons sa minimisation dans la fonction objectif.

Les expériences du scénario 2: consistent en la minimisation du makespan et des deux coûts Cr_1etCr_2. $E2 = w_1.C_{max} + w_2.(Cr_1 + Cr_2)$. La même procédure du premier type d'expériences est poursuivi; les poids varient (w_1, w_2)= (0.3, 0.7); (0.5, 0.5); (0.7, 0.3), β varie de 0.5, 1 à 2. Les résultats des expériences sont résumés dans le tableau 3.4. En comparant les deux tableaux 3.3 et 3.4 qui résument les résultats du premier et du deuxième type d'expériences, nous remarquons un changement important dans les valeurs de la fonction E_2. Cela est dû au fait que,

lorsque E1 est considéré, R_1gardent une certaine importance, dans la fonction objectif, par rapport aux temps opératoire. Dans le tableau 3.3, lorsque (w_1, w_2)= (0.3, 0.7), seulement les quantités basiques des ressources R_1 sont consommées, même quand β=2. Or que lorsque E_2 est considéré, l'importance deR_1 par rapport aux temps opératoire diminue, parce que deux fonctions dépendantes du temps opératoire sont à minimiser C_{max} et Cr_2. Dans le tableau 3.4, lorsque (w_1, w_2)=(0.3, 0.7), et même avec la plus petite valeur de β des quantités additionnelles à la quantité basique sont consommées, pour réduire le temps opératoire. Pour des valeurs plus grandes de β, la quantité maximale des ressources R_1est consommée.

Tableau 3.4 Résultats des simulations en minimisant E_2.

(w_1, w_2)	**(0.3,0.7)**			**(0.5,0.5)**			**(0.7,0.3)**		
B	0.5	1	2	0.5	1	2	0.5	1	2
C_{max}	**14.5**	**5.5**	**6.5**	**10**	**5.5**	**6.5**	**10**	**5.5**	**6.5**
Cr_1	**6**	**29**	**17**	**24**	**29**	**17**	**24**	**29**	**17**
Cr_2	**38.75**	**3.4**	**4.25**	**25.25**	**3.5**	**4.25**	**25.25**	**3.5**	**4.25**
E_2	35.67	24.33	16.8	29.62	19	13.87	21.77	13.6	10.92

Par ailleurs, le comportement du modèle par rapport aux premières et deuxièmes expériences ne change pas trop ; l'augmentation du poids w_1 au détriment de w_2 engendre toujours une augmentation dans l'allocation des ressources R_1. Ainsi lorsque β est plus grand, plus de ressources R_1 sont allouées.

3.3.2 Analyse théorique

L'analyse numérique précédente, a défini le rôle et l'influence de chaque type de ressource sur le modèle, mais seulement pour une instance donnée. Dans cette sous-section, et grâce aux déductions de l'analyse numérique précédente, nous menons une analyse théorique, qui est plus robuste et plus générale.

Nous supposons que $\overline{C_{max}}$ et $\overline{Cr_2}$ sont, respectivement, les valeurs du makespan et du coût des ressources R_2 d'un ordonnancement optimal, avec l'allocation des quantités basiques minimale des ressources R_1. Pour cet optimal ordonnancement m' est la somme des machines k' ayant un temps de production final égale au makespan: $C_{max} - C_{k'} = 0$, quelque soit k'.

Lemme 1: Tant que $w_2.Cu_1 < w_1.\frac{\beta}{m'} + w_2.Cu_2.\gamma.\beta$, la quantité des ressources R_1 allouées doit être augmenté par m' unités pour réduire la fonction objectif. À chaque machine k' une seule unité de ressourcesR_1 est allouée.

Le lemme reste correct jusqu'à ce que la quantité consommée deR_1 atteigne son maximum.

Démonstration du lemme:

Pour la consommation minimale deR_1:

$$\overline{OF} = w_1.\overline{C_{max}} + w_2.\overline{Cr_2} + w_2.Cr_1$$

Si on augmente la consommation des ressources R_1par une seule unité, le makespan ne décroît pas s'il y a plusieurs machines qui ont un temps de production total égale au makespan. Pour cela, nous augmentons la quantité de ressourcesR_1par m' unités.

Si la consommation deR_1augmente par n=n'.m' unités, Cr_2 diminuerait par$Cu_2.\gamma.n$ unités, et le makespan par $\left[\frac{n}{m'}\right]$ unités.

$$OF = w_1.\left(\overline{C_{max}} - \frac{n}{m'}.\beta\right) + w_2.(\overline{Cr_2} - Cu_2.(n.\beta).\gamma) + w_2.(Cr_1 + n.Cu_1)$$

$$= w_1.\overline{C_{max}} + w_2.\overline{Cr_2} + w_2.Cr_1 + n.w_2.Cu_1 - w_2.Cu_2.(n.\beta).\gamma - \frac{n}{m'}.w_1.\beta$$

If: $w_2.Cu_1 < w_2.Cu_2.\beta.\gamma - \frac{\beta}{m'}.w_1$

Et donc : $OF < \overline{OF}$

Littéralement, le lemme 1 veut dire que si la valeur des ressourcesR_2 avec celle du makespan ensemble, est supérieure à la valeur des ressourcesR_1, l'augmentation dans la consommation de R_1 améliore la fonction objectif. A partir de là, nous déduisons un deuxième lemme:

Lemme 2: Si la quantité optimale de ressources à consommer atteigne son maximum, la fonction objectif serait réduite à minimiser seulement le makespan et le coût des ressourcesR_2.

3.3.3 Exemples illustratifs

Dans cette partie, nous résolvons le problème pour une instance de 6 tâches et 2 machines. Cette instance sera projetée sur deux cas réels qui ont été introduit dans la première section. Le temps opératoire normal des tâches, quantités basiques deR_1 de l'instance sont résumés dans le tableau 3.5, les autres paramètres sont fixés selon le cas d'étude.

Tableau 3.5 Instances de l'exemple illustrant et la solution selon le cas

Jobs	1	2	3	4	5	6
$\mathbf{a_j}$	12	8	7	11	5	10
$\mathbf{r_{1j}}$	2	1	1	0	0	0
La solution du 1er cas :	C_{max}=28.5, Cr_1=4, Cr_2=68.4					
La solution du 2em cas :	C_{max}=29.7, Cr_1=8, Cr_2= 17.5					

- 1er cas: Ponçage des carreaux de granite

Dans cet exemple, les tâches consistent en le ponçage de différents carreaux de granite, les ressources R_1 représentent l'eau utilisée pour le ponçage, R_2 représente l'électricité consommée par la machine. Nous supposons que α=0.5, γ=0.6kWh/h, γ'=0.8kWh/h et β=0.3, cette valeur de β est supposée ainsi parce qu'une grande quantité de ressourcesR_1 est nécessaire pour accélérer le processus. Aussi, il est connu que l'électricité coûte plus que l'eau en industrie, $Cu_1 < Cu_2$ donc. Cu_1=1,

Cu_2=2. Nous supposons que la minimisation du coût des ressources importe plus que la minimisation du makespan, et donc (w_1, w_2)= (0.3, 0.7).

- 2^{em} cas: Détartrage dentaire

Dans cette exemple, les tâches consistent en l'opération de détartrage de dents pour des patients, les ressources R_1 représentent l'agent détartrant, R_2 est l'électricité consommé par la chaise dentaire. Nous supposons que α=0.9, γ=γ'=0.3kWh/h et β=1, de petite quantité de cette agent détartrant peut accélérer le processus. Cu_1=2, Cu_2=1. Ces valeurs sont fixées ainsi parce que le coût de l'agent détartrant est supérieur à celui de l'électricité. Quant aux poids des fonctions, nous supposons que w_1=w_2=0.5, parce que le dentiste est une partie de la machine, le temps d'utilisation de la machine est un temps d'occupation pour le dentiste ainsi que pour le patient qui souhaite l'exécution la plus rapide de la tâche, donc le makespan est aussi important que le coût des ressources.

Dans le tableau 3.5, nous présentons les solutions du problème projeté sur le premier et le deuxième exemple. Nous remarquons que, pour les deux cas, seulement les quantités basiques deR_1 sont consommées. Pour le premier cas, le temps réduit par unité de ressource β est relativement faible, donc pour réduire une unité de temps et deR_2, il faudrait de grande quantité deR_1, et cela coûtera plus chère. Si β avait été supérieur à 0.7, des quantités additionnelles deR_1 auraient été consommées. Pour le deuxième cas, seulement la quantité basique de R_1 est consommée parce que le coût de ces ressources est plus élevé que celui de R_2, et parce que la consommation de R_2 par unité de temps est relativement basse.

Grâce à l'analyse numérique et théorique réalisées dans les deux sections précédentes, nous avons pu résumer le comportement du modèle en un digramme, présenté dans la figure 3.4, tel que Pk est le nombre de positions occupées dans la machine k, N est l'ensemble des tâches non ordonnancées encore.

3.4 Adaptation d'une MOSA

L'ordonnancement est un outil d'aide à la prise de décision de court terme, par cela on entend que la résolution d'un problème d'ordonnancement doit se faire en temps réel. Cependant, vue la complexité des problèmes d'ordonnancement, un modèle mathématique, pour les résoudre, requiert une grande capacité de calcule et peut prendre plusieurs heures, des fois même des jours, et la taille des problèmes qu'il peut résoudre est limitée aux petites et, rarement, moyennes instances. D'où la nécessité de développer une approche de résolution approximative.

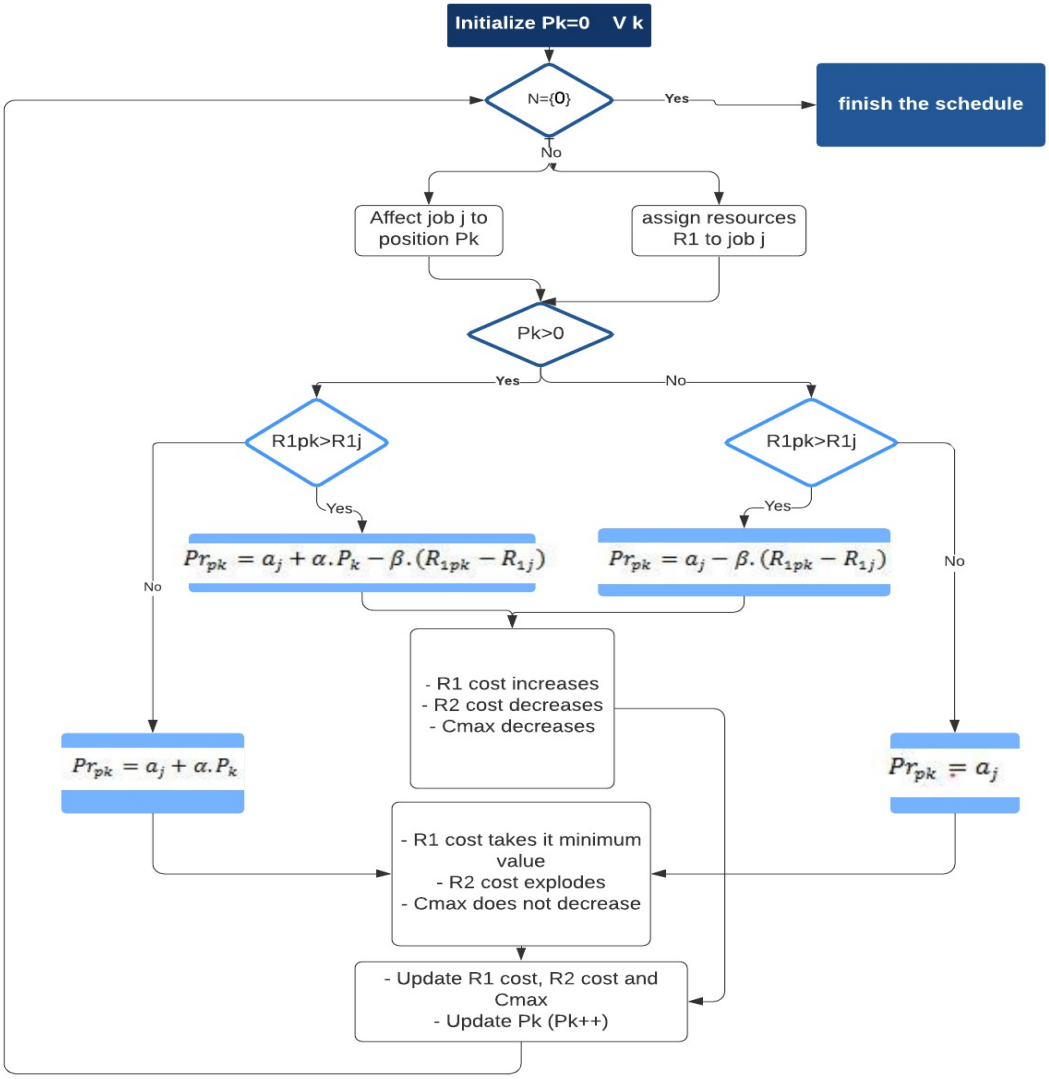


Figure 3.4 : Diagramme du processus d'ordonnancement

Dans cette section, nous développons un recuit simulé multi-objectif "MOSA", qui fournira des solutions non-optimales, mais satisfaisantes, en un temps de calcul acceptable. Développée par (S. Kirkpatrick et al., 1983), le recuit simulé est l'une des plus simples métaheuristiques. Comme mentionné dans le chapitre II, cette métaheuristique utilise une seule solution initiale, une température initiale et une probabilité d'acceptation. Par une méthode de recherche locale, une solution voisine est générée. Les deux solutions, initiale et voisine, sont comparées et la meilleure est sélectionnée comme solution initiale pour la prochaine itération. La comparaison entre les deux solutions se fait par le calcule de la fonction objectif, ou par le calcule de la probabilité que la solution voisine mène vers une meilleure solution. Les différentes étapes de l'implémentation d'un recuit simulé sont détaillées dans Algorithme 2.

Le recuit simulé multi-objectif garde les mêmes étapes détaillées en Algorithme 2, le changement intervient dans le calcul de la probabilité d'une

solution, comme mentionné dans le chapitre II. Donc l'algorithme utilisé pour la résolution du problème considéré dans ce chapitre est détaillé dans algorithme 3.

3.4.1 Les détails de l'implémentation de MOSA

a) Solution initiale: La définition d'une solution consiste en l'affectation des tâches aux des machines et l'allocation de ressources aux positions occupées, la séquence des tâches n'influence pas la fonction objectif. Donc, pour définir une solution initiale, i)-nous générons aléatoirement des valeurs binaires dans une matrice dont le nombre de lignes égale au nombre des tâches, et le nombre de colonnes égale aux nombre de machines. ii)-nous allouons seulement les quantités basiques des ressourcesR_1.

Algorithme 2: Pseudo-code du recuit simulé

Initialiser (S_i, T_i, L and P_a)
Tant que: critère d'arrêt pas atteint, répéter:
    générer une solution voisine : S'=PERTURB (S_i)
    Calculer ΔE, ΔE= OF (S') - OF (S_i)
    Si ΔE<= 0
        $S_i = S'$ {Mouvement accepté}
    Sinon:
        calculer la probabilité P de S'

$$P(s') = e^{\frac{\Delta E}{T_i}}$$

        Si: P≥ Pa:
            $S_i = S'$ {Mouvement accepté}
        Sinon:
            $S_i = S_i$ {Mouvement rejeté}
        Fin
    Fin
    $T_{i+1}=\alpha.T_i$
    Incrémenter i.
Fin

Algorithme 3: Pseudo-code de MOSA

Initialiser (S_i, T_i, L and P_a)
Tant que: critère d'arrêt pas atteint, répéter:
    générer une solution voisine : S'=PERTURB (S_i)
    Calculer ΔE, la somme desΔE_i, $\Delta E_i = f_i$ (S') - f_i (S_i), $\Delta E = \sum_i w_i \Delta E_i$
    Si ΔE<= 0
        $S_i = S'$ {Mouvement accepté}
    Sinon:
        calculer les probabilités $P_i$pour chaque fonction $f_i$de la solution S'

$$P_i(s') = e^{\frac{\Delta E}{T_i}}$$

        Pondérer les probabilités $P_i$en une seule probabilité P :

$$P = \prod_i P_i^{w_i}$$

Si: P≥ Pa:
$S_i = S'$ {Mouvement accepté}
Sinon:
$S_i = S_i$ {Mouvement rejeté}
Fin
Fin
$T_{i+1}=\alpha.T_i$
Incrémenter i.
Fin

b) Génération d'une solution voisine: la génération d'un voisin est un mouvement qui mène vers une solution proche. Cette notion du voisinage est très importante pour la diversification des solutions. Une technique de génération de voisin, bien choisie, permet le balayage de tout l'espace des solutions faisables. Toutefois, avant de définir cette technique, il est important de bien formuler le chromosome. Tableau 3.6 représente le chromosome utilisé dans l'algorithme 3. La première ligne contient les tâches, la second les machines où chaque tâche est exécutée, la troisièmes ligne contient les positions des tâches dans la machine, et la dernière ligne contient la quantité de ressourcesR_1 allouée à la tâche.

Une solution voisine est générée en deux mouvement: i)-Le premier mouvement est une mutation aléatoire appliquée sur une case aléatoire dans la ligne des machines. Cela permet l'accès à toutes les solutions possibles, parmi eux, des solutions où des positions lacunes dans la machines sont crées, pour corriger ce défaut, l'algorithme réaffecte les positions après chaque mutations. ii)-Le deuxième mouvement est l'allocation d'une quantité aléatoire de ressourcesR_1 à une tâche aléatoire. Cette quantité est générée selon une fonction de distribution uniforme U[a,b], tel que a est la quantité basique deR_1 pour la tâche choisisse, et b est la quantité maximale, $a = rb_{1j}$, $b = \frac{a_j}{\beta}.A\%$.

Tableau 3.6 Représentation d'une solution dans l'algorithme MOSA

Jobs:	j_1	j_5	j_3	j_2	j_4	j_7	j_6
Machines	2	2	3	1	2	1	3
Position	1	3	2	1	2	2	1
Resources R1	1	1	1	1	1	1	1

3.4.2 Les paramètres du MOSA

a) Température: la température initiale diminue au fur et à mesure des itérations, elle contrôle la probabilité d'acceptation d'une solution non-améliorée. Cette probabilité augmente pour les grandes valeurs de T, permettant à l'algorithme une certaine flexibilité de chercher même au voisinage d'une mauvaise solution, ce qui peut mener vers des solutions meilleures. Lorsque la température T diminue, la flexibilité de

recherche aux voisinages des mauvaises solutions reste possible mais peu probable.

Théoriquement, plus la température initiale Ti est grande, plus elle permet plus de flexibilité à l'algorithme. Mais, elle ne doit pas être assez grande au point de ralentir l'exécution du programme.

b) Probabilité d'acceptation: La valeur de la probabilité d'acceptation est souvent prise entre 0.5 et 0.8. Des valeurs plus petites peuvent éloigner l'algorithme de l'optimum, tant que des valeurs plus grandes, peuvent exclure des chemins plus courts vers l'optimum.

c) Taux de refroidissement α: c'est le taux avec lequel la température diminue, il doit être inférieur à 1, mais assez proche de lui, pour permettre un balayage précis des solutions voisines. il est souvent pris entre 0.7 et 0.99.

d) Critère d'arrêt: c'est le paramètre qui décide de l'arrêt des itérations. Théoriquement, c'est le point où la température s'annule, (Kim et al., 2002). Par ailleurs, l'algorithme prend du temps pour atteindre cette valeur. Donc nous décidons que l'algorithme s'arrête après L itérations sans amélioration.

3.4.3 Fixation des paramètres de MOSA

Pour déterminer les valeurs des paramètres définis dans la sous-section précédente, nous varions les valeurs des paramètres de MOSA comme suit : (Ti = {300 400 500}), (α=0.99), (P={0.55, 0.6, 065, 0.7}), (L={30, 50}). Nous obtenons 24 combinaisons, comme illustré dans le tableau 3.7. Ensuite, nous générons 10 différentes instances de petites tailles, et nous les exécutons avec le programme de MOSA pour chaque combinaison et pour chaque instance. Cela pour calculer le décalage des solutions de MOSA données par chaque combinaison par rapport aux solutions optimales. Les solutions optimales de ces petites instances sont obtenues par le modèle mathématique, Lingo 10 comme solveur. Les résultats de ces expériences sont résumés dans le tableau 3.7.

Tableau 3.7 Le décalage de MOSA par rapport à l'optimum pour différentes valeurs de (Ti, α, P, L)

(Ti, α, P, L)	**Gap%**	**(Ti, α, P, L)**	**Gap%**	**(Ti, α, P, L)**	**Gap%**
(300, 0.99, 0.55, 30)	0.089	(400, 0.99, 0.55, 30)	0.074	(500, 0.99, 0.55, 30)	0.063
(300, 0.99, 0.55, 50)	0.066	(400, 0.99, 0.55, 50)	0.055	(500, 0.99, 0.55, 50)	0.059
(300, 0.99, 0.6, 30)	0.081	**(400, 0.99, 0.6, 30)[a]**	**0.054**	(500, 0.99, 0.6, 30)	0.056
(300, 0.99, 0.6, 50)	0.067	(400, 0.99, 0.6, 50)	0.052	(500, 0.99, 0.6, 50)	0.058
(300, 0.99, 0.65, 30)	0.082	(400, 0.99, 0.65, 30)	0.070	(500, 0.99, 0.65, 30)	0.086
(300, 0.99, 0.65, 50)	0.069	(400, 0.99, 0.65, 50)	0.064	(500, 0.99, 0.65, 50)	0.064
(300, 0.99, 0.7, 30)	0.081	(400, 0.99, 0.7, 30)	0.093	**(500, 0.99, 0.7, 30)[a]**	**0.052**
(300, 0.99, 0.7, 50)	0.064	(400, 0.99, 0.7, 50)	0.058	(500, 0.99, 0.7, 50)	0.058

[a] Les combinaisons qui ont donné les meilleures résultats.

3.5 Développement d'un algorithme de décomposition "2-steps-algorithm"

La résolution d'un tel problème d'ordonnancement consiste en l'affectation des tâches aux machines et l'allocation des quantités optimales des ressourcesR_1 aux tâches, dans le but de minimiser le makespan et le coût des ressourcesR_1et R_2. Pour améliorer les résultats de MOSA, nous proposons un algorithme de décomposition, utilisant les deux lemmes proposés précédemment. À travers cet algorithme, nous visons à alléger la lourdeur des calculs en divisons le problème en deux sous-problèmes :

- Sous-problème d'affectation de tâches.
- Sous-problème d'allocation des ressources.

L'algorithme de décomposition, appelé 2-steps-algorithm, consiste essentiellement à :

- Trouver la meilleure affectation des tâches aux machines, avec la consommation des quantités basiques de R_1, de façon à minimiser C_{max}et Cr_2. Cela en utilisant une version améliorée de l'algorithme MOSA, précédemment défini.
- Déterminer la quantité optimale des ressources R_1 à allouer aux tâches. Cela en utilisant les deux lemmes proposés dans la section 3.4.2.

Les étapes de l'algorithme « 2-steps algorithm » sont détaillées dans **algorithme 5**

3.5.1 Les détails de l'implémentation du « 2-steps algorithm »

Pour le premier sous-problème, qui est l'affectation des tâches aux machines, une version améliorée de MOSA est proposée, cette amélioration apparait dans la solution initiale, qui n'est plus générée aléatoirement, mais à partir d'une heuristique « heuristique LPT/SPT » qui est détaillée dans algorithme 4.

Algorithme 4: Pseudo-code de l'heuristique LPT/SPT

```
Initialiser (itération i=1, N, M, L(i)={N})
Tant que: Tant que L(i) est non-vide, répéter:
    Ordonner les tâches de la liste L(i) selon la loi LPT
        Affecter une tâche de L(i) à chaque machine, seulement les M premières tâches sont affectées
    Réordonner les tâches restantes dans la liste L(i) selon la loi SPT
        Affecter une tâche de L(i) à chaque machine, seulement les M premières tâches sont affectées
    Incrémenter i.
Fin
```

Dans la littérature, lorsque l'objectif à minimiser est le makespan, l'heuristique LPT est largement utilisée pour générer une solution initiale. Toutefois, notre problème ne s'agit pas de minimiser seulement le makespan mais aussi le coût des ressources R_2, et dans la section 3.4.1 nous avons montré la relation entre la minimisation de la détérioration et la minimisation du Cr_2. Donc, nous proposons une heuristique appelée « LPT/SPT heuristique » qui divise uniformément les tâches aux machines pour minimiser le Cr_2. Cette heuristique affecte à chaque machine le même nombre de tâches ; et assure que les tâches qui ont le plus large temps opératoire soient affectées avec celles qui ont le plus petit temps opératoire.

Algorithme 5: Pseudo-code de l'algorithme de décomposition « 2-steps algorithm »

Affectation des Tâches
  Adaptation de l'algorithme MOSA
  Définir *OF* et ses paramètres
Fin
Allocation des ressources
  Initialiser $(w_1, w_2, Cu_1, Cu_2, \beta, \gamma, m')$
  Si $w_2.Cu_1 < w_1.\frac{\beta}{m'} + w_2.Cu_2.\gamma.\beta$:
    Ajouter m' unités deR_1, chaque unité à la plus large tâche de chaque machine k'
    Calculer $\overline{OF}$, la nouvelle fonction objectif
  Fin
  Si $\overline{OF} \leq OF$
    $OF = \overline{OF}$
    Répéter l'étape 1
  Fin
  $S_i = S_i$ {Mouvement rejeté}
Fin

3.6 Analyse des performances des deux algorithmes

Plusieurs expériences sont effectuées, pour évaluer les performances du modèle mathématique, MOSA et de l'algorithme de décomposition « 2-steps algorithm ». Les algorithmes ont été programmés en Java, le modèle mathématique en lingo 10. les programmes ont été exécutées dans un micro-ordinateur de Samsung, intel core i3 CPU et une RAM de 8GB.

3.6.1 Génération des instances

Avant la génération des instances, quelques facteurs doivent être pris en considération, comme le nombre de tâche, le nombre de machines et la dispersion du temps opératoires dans chaque instance. Aucun Benchmark qui convient à notre problème n'a été retrouvé dans la littérature. Nous avons donc assemblé des

informations de différents articles pour créer nos propres instances. Les informations utilisées pour générer des instances sont résumées dans le tableau 3.8.

Les tailles des instances ont été inspirées de (Lin and Ying, 2015), la dispersion des temps opératoire de (Fanjul-Peyro et al., 2017) et (Su and Lien, 2009). Le temps opératoire normal est généré aléatoirement selon une fonction de distribution uniforme U [1, 50]. Quant aux quantités basiques des ressourcesR_1, elles ont été générées aléatoirement selon une fonction de distribution uniforme U [0, 3].

De (Tigane et al., 2018) et (Ji et al., 2013) nous avons remarqué que le taux de détérioration α et le temps réduit par unité de ressource β sont générés entre 0 et 1. Se référant à ces papiers, nous avons générer α, β et γ selon une fonction de distribution uniforme U [0, 1]. Quant à γ', il est généré comme γ, mais en gardant la possibilité qu'il soit légèrement supérieur à celui-ci. La somme des deux poids w_1 et w_2 doit être égale à 1 selon (Collette and Siarry, 2002). Nous considérons les trois scénarios, considérés par (Tigane et al., 2018) : $(w_1, w_2) = (0.3, 0.7)$, $(w_1, w_2) = (0.5, 0.5)$et $(w_1, w_2) = (0.7, 0.3)$.

Tableau 3.8 Génération des instances

Les paramètres à générer	**Les valeurs des paramètres**		**Références**
La taille des instances	Small	10N 2M	(Lin and Ying, 2015)
	Medium	20N 4M	
	Large	100N 10M	
Dispersion du temps opératoire normal	N(1, 50)		(Su and Lien, 2009) (Fanjul-Peyro et al., 2017)
La consommation en R_1	N(0, 3)		(Tigane et al., 2018) (Ji et al., 2013)
α, β, γ, γ'	N(0, 1)		
Les poids et coûts unitaires	$(w_1, w_2) = (0.3, 0.7)$	$(Cu_1, Cu_2) = (1,2)$	(Tigane et al., 2018), (Collette and Siarry, 2002)
		$(Cu_1, Cu_2) = (1,1)$	
		$(Cu_1, Cu_2) = (2,1)$	
	$(w_1, w_2) = (0.5, 0.5)$	$(Cu_1, Cu_2) = (1,2)$	
		$(Cu_1, Cu_2) = (1,1)$	
		$(Cu_1, Cu_2) = (2,1)$	
	$(w_1, w_2) = (0.7, 0.3)$	$(Cu_1, Cu_2) = (1,2)$	
		$(Cu_1, Cu_2) = (1,1)$	
		$(Cu_1, Cu_2) = (2,1)$	

3.6.2 Développement d'une borne inférieure

Une solution pour le problème considéré est une somme pondérée des valeurs du makespan, du coût deR_1et du coût deR_2. Les quantités optimales deR_1sont calculées grâce au lemme 1 qui nécessite les paramètres suivants : w_1, w_2. Cu_1, Cu_2, β, γ, m'. En calculant la borne inférieur m' n'est pas défini, donc

nous calculons leCr_1pour m'=1, m'=2, m'=3…m'=M, et nous prenons la plus grande valeur.

$\forall\, m' = \{1, 2, 3 \ldots . . M\}$

Si

$$w_2 . Cu_1 < w_1 . \frac{\beta}{m'} + w_2 . Cu_2 . \gamma . \beta$$

Donc:

$$R_1 = R_{1max} = \sum_{j=1}^{N} rb1_j + \left[\sum_{j=1}^{N} \frac{a_j}{\beta} * A\% \right]$$

Sinon:

$$R_1 = \sum_{j=1}^{N} rb1_j$$

Et la borne inférieur de Cr_1 est définie par l'équation (3.20)

$$LB_{Cr1} = Cu_1 . R_1 \tag{3.20}$$

La valeur minimale atteignable de Cr_2 est obtenue en affectant le même nombre de tâches à chaque machine, par cela les unités de temps ajoutées par la détérioration sont minimisées, et sont calculées par l'équation (3.21), et la borne inférieur deCr_2 est définie par l'équation (3.22)

$$\sum_{i=1}^{M} \alpha . \left(\frac{N}{M} - 1\right) = \alpha . (N - M) \tag{3.21}$$

$$LB_{Cr2} = Cu_2 . \left(\sum_{j=1}^{N} a_j + \alpha . (N - M) - \beta . \left(R_1 - \sum_{j=1}^{N} rb_{1_j}\right)\right) \tag{3.22}$$

Quant à la borne inférieur du makespan, définie par l'équation (3.23), elle est conventionnellement définie par le ratio de la somme des temps opératoires aux nombre de machines (Sule and Sule, 2008), comme le temps opératoire ici dépend de la détérioration et de la quantité de ressources consommées, nous ajoutons à la borne inférieure du makespan les unités de temps de la détérioration de l'équation (3.21) et enlevons les unités de temps réduites par les ressourcesR_1consommées.

$$LB_{Cmax} = \left(\sum_{j=1}^{N} a_j + \alpha . (N - M) - \beta . \left(R_1 - \sum_{j=1}^{N} rb_{1_j}\right)\right) \Big/ M \tag{3.23}$$

La borne inférieure de toute la fonction objectif est définie dans l'équation (3.24), par l'ensemble des trois bornes inférieures présentées dans les équations (3.20), (3.22) et (3.24).

$$LB_{OF} = w_1 . LB_{Cmax} + w_2 . (LB_{Cr1} + LB_{Cr2}) \quad (3.24)$$

Applicabilité de la borne inférieure : pour illustré l'applicabilité de la borne inférieure, nous proposons un exemple numérique de 6 tâches et 3 machines. Les paramètres de l'exemple sont présentés dans le tableau 3.9

Deux scénarios sont considéré, lorsque $(Cu_1, Cu_2) = (1,1)$et lorsque $(Cu_1, Cu_2) = (2,1)$.

Tableau 3.9 Exemple sur l'applicabilité de la borne inférieure

Jobs	**J1**	**J2**	**J3**	**J4**	**J5**	**J6**
Aj	9	3	8	4	6	6
rb1j	1	1	1	1	1	1
α=1, β=1, γ=1, γ'=1 et $(w_1, w_2) = (1,1)$.						

Le solveur fourni les valeurs de la fonction objectif suivantes :

Pour $(Cu_1, Cu_2) = (2,1)$ $Cr_1 = 12,\ Cr_2 = 39,\ C_{max} = 13$

Pour $(Cu_1, Cu_2) = (1,1)$ $Cr_1 = 34,\ Cr_2 = 11,\ C_{max} = 4$

La borne inférieure, quant à elle, fourni presque les mêmes valeurs de la fonction objectif :

Le calcule se fait pour m'=1, m'=2 and m'=3, et nous choisissons la plus grande valeur de Cr_1

Pour$(Cu_1, Cu_2) = (2,1)$ $LB_{Cr_1} = 12, LB_{Cr_2} = 39,\ LB_{C_{max}} = 13$

Pour$(Cu_1, Cu_2) = (1,1)$ $LB_{Cr_1} = 34, LB_{Cr_2} = 11,\ LB_{C_{max}} = 3.66$

Au moins pour cet exemple, la borne inférieure fourni des solutions presque aussi bonne que le modèle mathématique.

3.6.3 Configuration des expériences

Les performances d'une approche de résolution sont définies par la qualité de ses solutions, et parce qu'un problème d'ordonnancement doit être résolu en temps réel, les performances d'une approche sont aussi relatives aux temps de calcul qu'elle nécessite pour résoudre le problème. Pour cette raison, nous comparons les solutions de MOSA et de « 2-steps algorithm » avec les solutions optimales du modèle mathématique, pour les petites et moyennes instances. Cela nous permet d'examiner la qualité des solutions approximatives des deux algorithmes, et aussi d'examiner l'efficacité du modèle mathématique en termes de temps de calcul. Dans le tableau 3.10, nous résumons les temps de calcul du modèle mathématique et du « 2-steps algorithm ». Nous ne considérons pas le temps de calcul de MOSA dans ce tableau, parce qu'il est presque le même que celui du « 2-steps algorithm ». La

comparaison entre les solutions approximatives et les solutions exactes se fait par le calcul du décalage ou de l'erreur moyenne, appelée gap. Ce gap est calculé pour trois valeurs de (w_1, w_2) et trois valeurs de (Cu_1, Cu_2), c.à.d. il est calculé 9 fois pour chaque instance générée, pour les deux algorithmes. Pour les petites et moyennes instances, Gap1 représente le décalage des solutions de MOSA par rapport aux solutions exactes, il est calculé par l'équation (3.25). Gap2 représente le décalage des solutions de « 2-steps algorithm » par rapport aux solutions exactes, il est calculé par l'équation (3.26)

$$Gap1 = \frac{\big(\text{Solution(MOSA)} - \text{Solution(optimale)}\big) * 100}{\text{Solution(optimale)}} \tag{3.25}$$

$$Gap2 = \frac{\big(\text{Solution}(2 - \text{steps alg.}) - \text{Solution(optimale)}\big) * 100}{\text{Solution(optimale)}} \tag{3.26}$$

Pour les grandes instances, le modèle mathématique ne nous fourni pas les solutions exactes. Toutefois, pour vérifier la qualité des solutions des deux algorithmes, nous développons une borne inférieure, et nous comparons les solutions de MOSA et de « 2-steps alg. » avec la borne inférieure par le calcul du Gap4 et Gap2, qui sont représentés par les équations (3.27) et (3.28), respectivement.

$$Gap4 = \frac{(\text{Solution(MOSA)} - \text{LB}) * 100}{\text{LB}} \tag{3.27}$$

$$Gap5 = \frac{(\text{Solution}(2 - \text{steps alg.}) - \text{LB}) * 100}{\text{LB}} \tag{3.28}$$

Ainsi la qualité de la borne inférieure n'est pas assurée, elle peut être loin de la solution optimale. Pour cela, nous calculons Gap3, qui est le décalage de la borne inférieure des solutions optimales, Gap3 est représenté par l'équation (3.29). Le Gap3 est calculé que pour les petites et moyennes instances, qui peuvent être résolues exactement par le modèle mathématique. Cependant, il servira de repère pour analyser les gaps des solutions approximatives par rapport aux solutions exactes.

$$Gap3 = \frac{(solution(optimale) - LB) * 100}{solution(optimale)} \tag{3.29}$$

Le tableau 3.11 résume les différentes configurations des expériences de l'analyse réalisée, il précise les Gaps calculé pour chaque instances et chaque type de comparaison ; la comparaison entre MILP et MOSA, la comparaison entre MILP

et 2-steps algorithm, la comparaison entre MILP et la borne inférieure LB, la comparaison entre LB et MOSA, et finalement la comparaison entre LB et 2-steps algorithm. Aussi le tableau précise les figures où se trouve chaque Gap.

3.6.4 Résultats et discussion

Le premier objectif de cette analyse numérique est d'examiner l'efficacité des deux algorithmes ; MOSA et « 2-steps algorithm ». Le deuxième objectif est de surligner l'impact des paramètres (w_1, w_2)et (Cu_1, Cu_2) sur les performances des deux algorithmes MOSA et 2-steps-algorithm. Les temps de calcules du modèle mathématique et du 2-steps-algorithm sont résumés dans le tableau 3.14, et les résultats des expériences comparant entre le modèle mathématique, MOSA et le 2-steps-algorithm sont résumés dans les figures 3.5, 3.6 et 3.8, la figure 3.7 illustre la qualité de la borne inférieure en représentant son décalage par rapport à l'optimum.

À partir du tableau 3.10, il est remarquable que le temps de calcul du modèle mathématique augmente rapidement et non-uniformément d'une instance à une autre, il atteint les 5 heures lors de la résolution des moyennes instances, tandis que le 2-steps-algorithm résout les petites et moyennes instances en pas plus de 0.5seconde, et donne des solutions approximatives très proches des solutions optimales. Par cela, nous concluons qu'en termes de temps de calcule, le 2-steps-algorithm est plus efficace que le modèle mathématique.

Les résultats, dans les deux figures 3.5 et 3.6 comparent entre les solutions approximatives de MOSA et « 2-steps alg. » et les solutions optimales du modèle mathématique, pour les petites et moyennes instances, respectivement.

Les valeurs du Gap1, représentés en pourcentage, montrent que les solutions fournies par MOSA sont des solutions satisfaisantes et peuvent être acceptées avec un gap maximal de 4.2% pour les petites instances, et un gap maximal de 10% pour les moyennes instances. Tandis que les valeurs du Gap2, aussi représentés en pourcentage, montrent que les solutions fournies par « 2-steps alg. » sont meilleures, et s'approches des solutions optimales avec un gap nul pour la majorité des petites instances, et un gap maximal de 1% pour les moyennes instances. Plusieurs études, dans la littérature, ont utilisées cette approche de comparaison ; Shahvari and Logendran, (2017) ont comparé entre les résultats d'un algorithme de recherche Tabou et les résultats du modèle mathématique, et ils ont obtenu un gap maximal de 16%.

Comme il est clair dans la figure 3.5 et 3.6, le Gap1 diminue et Gap2 s'approche du 0 lorsque le coût de R_1est supérieur au celui de R_2. Cela confirme que le « 2-steps alg. » donne de meilleurs résultats pour de l'allocation de R_1que pour l'affectation des tâches. En plus, la figure 3.6 montre une augmentation dans le Gap1 et Gap2 lorsque le poids du makespan est supérieur à celui des coûts, cela s'explique par l'efficacité des algorithmes dans la minimisation des coûts.

La figure 3.7 représente le décalage « Gap3 » de la borne inférieure par rapport à l'optimum, calculé pour les instances qui peuvent être résolues par le modèle mathématique. Son but est d'analyser l'efficacité de la borne inférieure. Nous remarquons, à travers cette figure, que le décalage de la borne inférieure varie de 1% à 9% lorsque$w_1 < w_2$, et de 1 à 7% lorsque$w_1 > w_2$.

Tableau 3.10 Temps de calcul moyen pour le modèle mathématique et le « 2-steps alg. ».

Instances	**Small**		**Medium**		**Large**
	Solver CPU time(sec)	2-step alg. CPU time (sec)	Solver CPU time (sec)	2-step alg. CPU time (sec)	2-step alg. CPU time(sec)
I1	11	0.26	7200	0.52	3.7
I2	18	0.24	10800	0.57	3.8
I3	2940	0.19	7200	0.55	3.5
I4	25	0.20	4800	0.53	3.8
I5	660	0.23	4710	0.52	4.2
I6	581	0.21	1900	0.56	4
I7	610	0.28	3600	0.51	3.9
I8	20	0.26	2400	0.45	3.9
I9	427	0.28	7200	0.48	4.1
I10	390	0.27	4230	0.57	3.4

Tableau 3.11 Configuration des expériences

Type de comparaison		**MILP-MOSA**	**MILP-2-steps alg.**	**MILP-LB**	**LB-MOSA**	**LB-2-steps alg.**
Instance						
(w_1, w_2)		{(0.3-0.7), (0.5-0.5), (0.7-0.3)}				
(Cu_1, Cu_2)						
Petite instance	{(1, 2), (1, 1), (2, 1)}	Gap1 Figure 3.5	Gap2 Figure 3.5	Gap 3 Figure 3.7		
Moyenne instance		Gap1 Figure 3.6	Gap2 Figure 3.6			
Grande instance					Gap4 Figure3.8	Gap5 Figure 3.8

Finalement, la figure 3.8 représente le décalage des deux algorithmes par rapport à la borne inférieure, Gap4 et Gap5. Le décalage de MOSA varie de 10%à25% et le décalage de « 2steps alg. » varie de 7% à16%. Bien que le « 2-steps alg. » est moins décalé que MOSA, mais son gap parait trop grand. Cependant, de la figure 3.7, nous remarquons que la borne inférieure est déjà décalée de l'optimum, son gap atteint des fois les 9%. Du moment où de le repère est lui-même être décalé de 10% de l'optimum, un décalage de 16%de la solution du repère est acceptable. (Liaw et al., 2003)ont proposé une borne inférieure le décalage peut atteindre les 42%, et ils l'ont utilisé pour développer un algorithme *« branch and bound »*.

A partir des résultats résumés dans les figures 3.5, 3.6, 3.7 et 3.8, et de l'analyse précédente, nous pouvons conclure que le « 2-steps alg. » surpasse le MOSA en termes de qualité des solutions. Cet algorithme est aussi plus efficace que le modèle mathématique en fournissant des solutions de bonne qualité, pour les petites et moyennes instances, et des solutions satisfaisantes, pour les grandes instances, en un temps de calcul de quelque secondes. Il est aussi à conclure que le « 2-steps alg. » est plus efficace pour la minimisation des coûts que pour la minimisation du makespan.

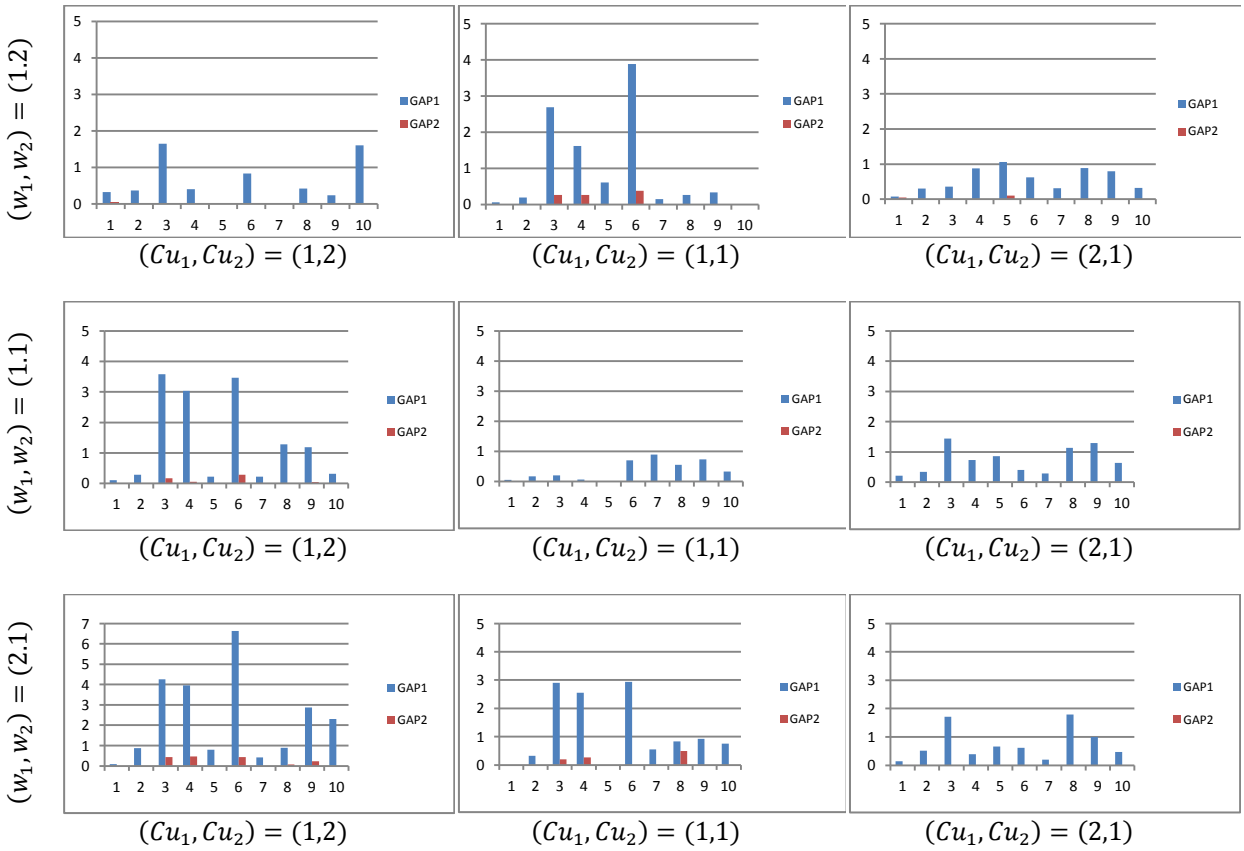


Figure 3.5 : Les performances de MOSA et de "2steps-algorithm" pour les petites instances

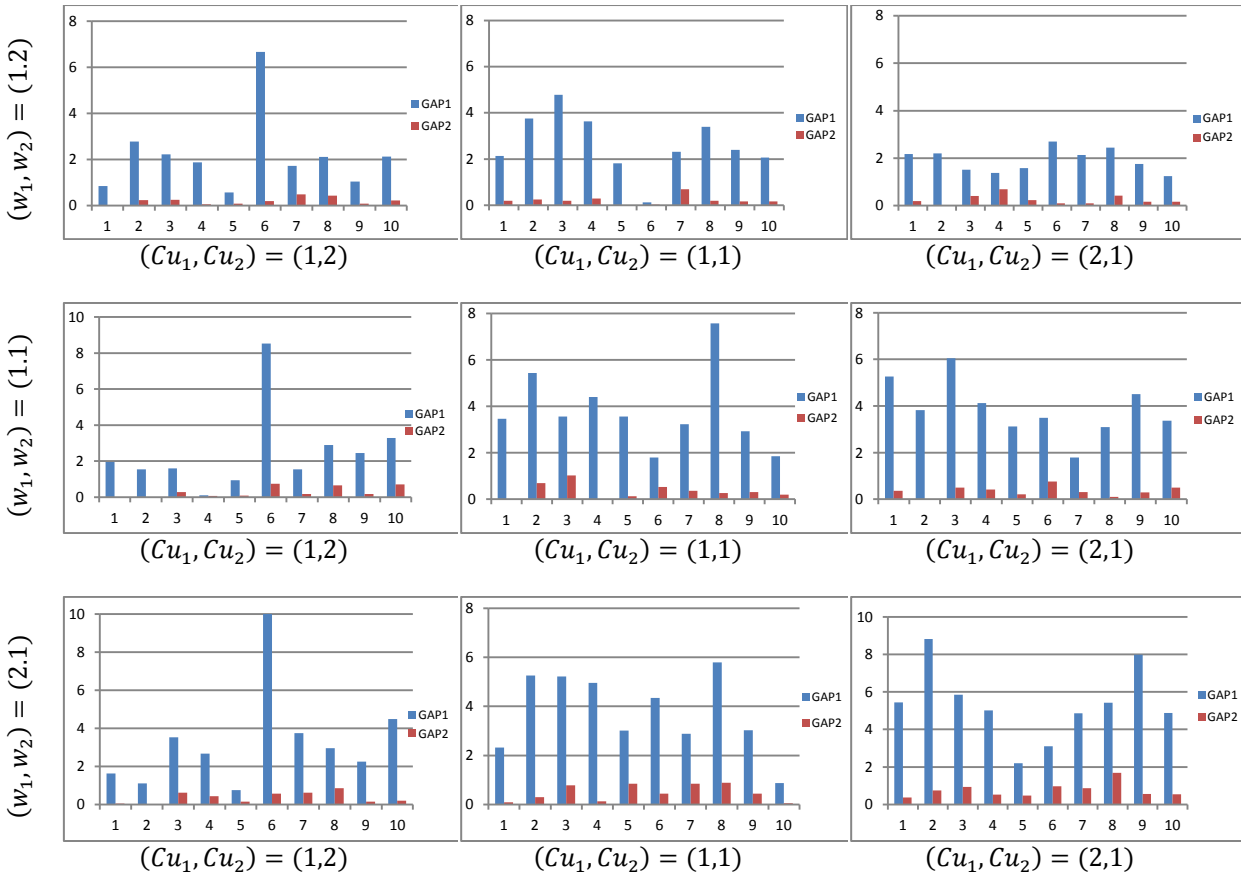


Figure 3.6 : Performance de MOSA et "2-steps algorithm" pour les moyenne instances

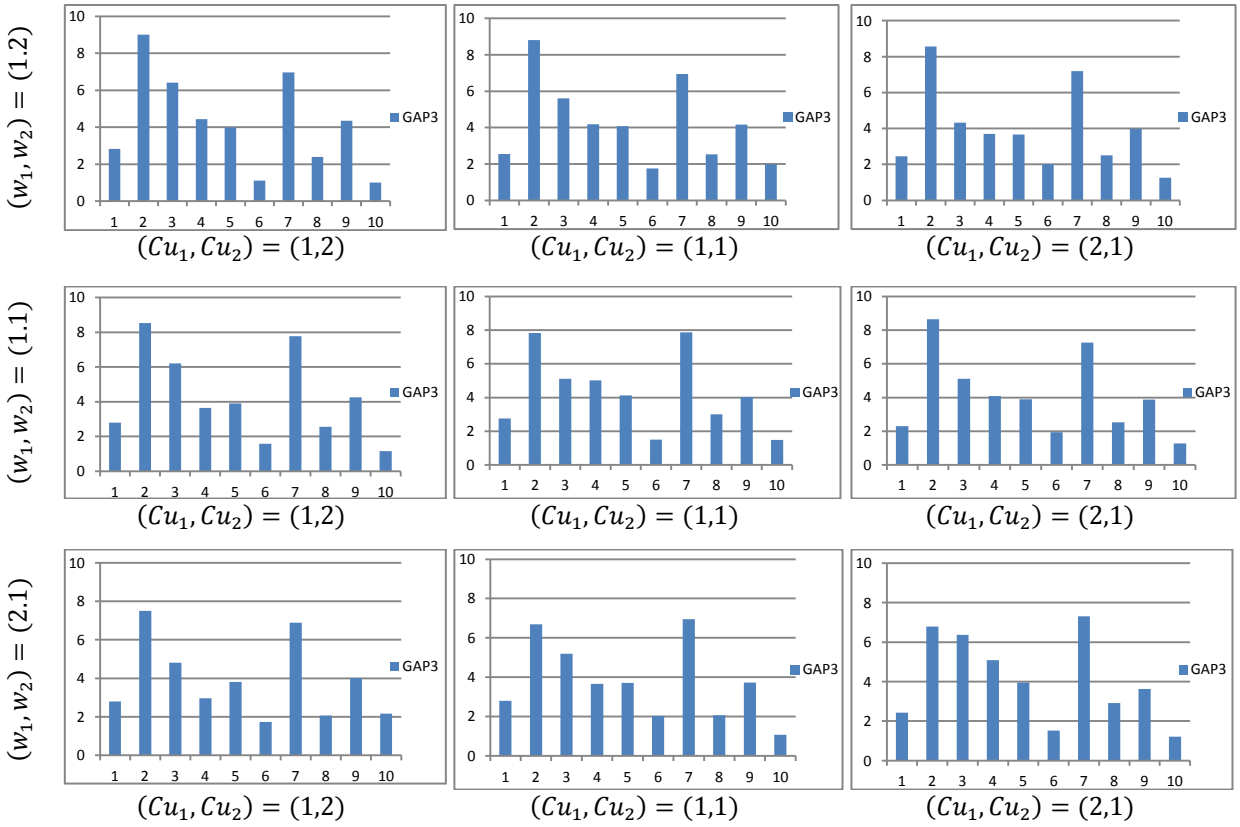


Figure 3.7 : Analyse de performance de la borne inférieure

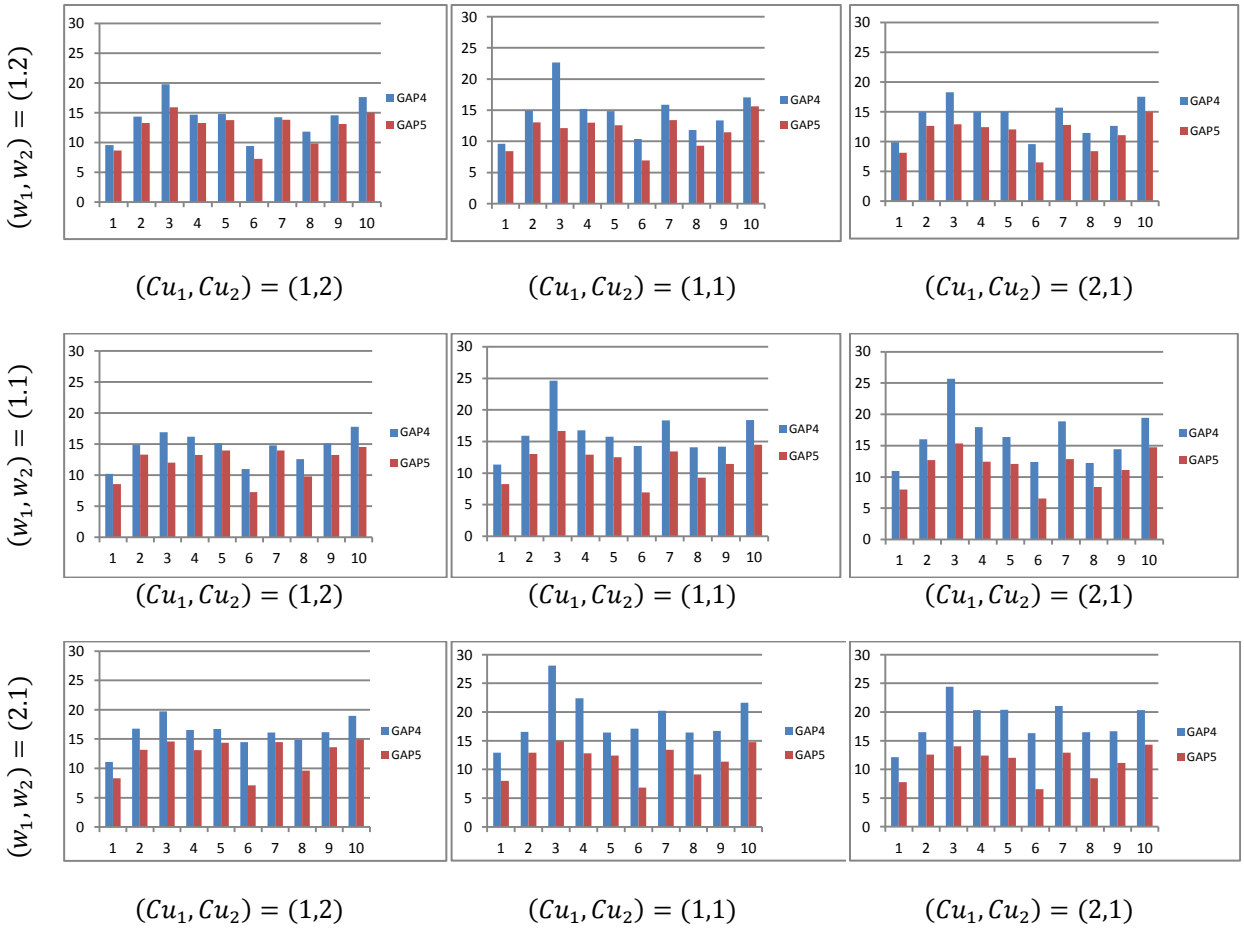


Figure 3.8 : Analyse de performance de MOSA et de "2-steps algorithm" pour les grandes instances

3.7 Synthèse

L'étude menée dans ce chapitre nous a permis l'obtention de quelques résultats, qui nécessitent plus d'attention, ainsi que l'application de ces résultats nécessite d'être surlignée.

Les résultats se résument en plusieurs points importants, à commencer par le modèle mathématique qui a prouvé sont efficacité pour les petites taille du problème, mais qui nécessite une grande capacité de calcul pour les moyennes tailles du problème, prenant plusieurs heures pour résoudre une instance de 20 tâches et 4 machines. Par la suite, l'analyse combinatoire du modèle mathématique proposé a permis le développement de deux lemmes, dont l'un assure l'allocation optimale des ressources R_1sous certaines conditions. Vers la recherche d'une approche de résolution efficace pour les moyennes et grandes instances, le recuit simulé multi objectif classique a fait preuve d'efficacité dans la résolution de notre problème, en fournissant des solutions approximatives satisfaisantes dans un temps de calcul très raisonnable. Par ailleurs, une version améliorée du recuit simulé multi objectif, où la solution initiale est générée par une heuristique développée « l'heuristique LPT/SPT », s'avère plus efficace, surtout dans la minimisation du makespan. En outre, l'application du lemme 1 dans l'allocation des ressources R_1aux tâches

améliore encore plus la qualité des solutions. Les deux derniers points ; l'amélioration du MOSA et l'application du lemme 1, ont mené vers le développement d'un algorithme de décomposition « 2-steps algorithm ». Celui-ci est la meilleure approche parmi les trois approches abordées (modélisation mathématique, le recuit simulé multi objectif et le 2-steps-algorithm) ; il surpasse le modèle mathématique en termes de temps de calcul et par sa capacité de résoudre des tailles illimitées du problème, et il surpasse le MOSA en termes de qualité des solutions fournies.

L'applicabilité des résultats de cette étude apparait dans la solution fournie par le modèle mathématique, MOSA ou le 2-steps algorithm, et qui définit l'affectation des tâches aux machines, et l'allocation des ressources aux tâches, permettant aux gestionnaires de l'atelier de connaitre préalablement la charge sur machines et de définir le besoin en ressourcesR_1, cela facilite la prise de décision en assurant une meilleure gestion des stocks et un processus de production harmonieux.

Une bonne solution ; optimale ou proche de l'optimale, minimise le temps de production total et les coûts des deux ressources, ce qui optimise l'exploitation du système de production, améliore son efficacité, son rendement et par la suite augmente la profitabilité de toute l'organisation.

3.8 Conclusion

Dans ce chapitre, nous avons étudié le problème de minimisation du makespan et des coûts de ressources consommables dans un environnement de machines parallèles identiques avec effet de détérioration et deux différents types de ressourcesR_1etR_2. Le temps opératoire d'une tâche est une fonction de sa position dans la machine et de la quantité des ressourcesR_1. En outre, la consommation de chaque tâche en ressources R_2est une fonction du temps opératoire actuel de la tâche. Pour résoudre ce problème, nous avons proposé un modèle mathématique, puis deux lemmes, ensuite nous avons développé un recuit simulé multi-objectif « MOSA », et finalement nous avons développé un algorithme de décomposition « 2-steps-algorithm » qui utilise une version améliorée de MOSA pour l'affectation des tâches, et le lemme 1 pour une allocation optimale des ressources.

Les solutions fournies par MOSA, « 2-steps algorithm » sont comparées aux solutions optimales fournies par le modèle mathématique, pour les petites et moyennes instances, ensuite, elles sont comparées à une borne inférieure, pour les grandes instances. Ainsi les temps opératoires du modèle mathématique et du 2-steps algorithme sont comparés. À travers cette comparaison, nous avons prouvé que les deux algorithmes ; MOSA et « 2-steps algorithm » sont capables de résoudre rapidement les petites, moyennes et grandes tailles du problème ; en un temps de calcul qui n'atteigne pas les 5 secondes, en maintenant une qualité de solutions à un niveau acceptables. Le « 2-steps algorithm » surpasse le MOSA en termes de qualité des solutions.

L'étude menée dans ce chapitre considère deux types de ressources de la même catégorie; ressources consommables continues. L'une de ces ressources R_2 est consommée en fonction du temps opératoire actuel, elle a un coût qui pèse sur l'entreprise. Dans une autre version du problème, cette ressource peut être considérée comme une énergie à minimiser. L'autre ressource R1, qui contrôle le temps opératoire, est consommable et disponible en quantités suffisantes, c.à.d. illimitées. Cependant, une autre version du problème peut considérer des ressources renouvelables disponibles en quantités limitées, qui eux aussi contrôlent le temps opératoire; comme des ressources humaines qualifiées. Dans ce cas, la séquence des tâches et l'allocation des ressources doivent être dynamique pour une exploitation optimale de ces mêmes ressources. Il parait intéressant d'étudier le même problème étudié dans ce chapitre « ordonnancement de machines parallèles identiques avec effet de détérioration », avec la considération de l'énergie comme fonction à optimiser et sous les contraintes des ressources renouvelables ; humaines et matériels.

Dans le chapitre suivant, nous allons modéliser et résoudre le problème d'ordonnancement de machines parallèles identiques avec effet de détérioration, optimisation de la consommation totale de l'énergie, avec considération des ressources humaines qualifiées et de ressources partagées.

Chapitre 4

Problème d'ordonnancement à machines parallèles sous des contraintes non-conventionnelles : Application industrielle

Résumé : Dans ce chapitre nous étudions un problème d'ordonnancement à machines parallèles, où l'effet de détérioration et des contraintes de ressources renouvelables : ressources humaines qualifiées et outils sont considérés, le makespan et la consommation totale en énergie sont minimisés. Nous proposons un modèle mathématique pour résoudre le problème, mais vue sa complexité élevée, le modèle mathématique ne résout que les petites instances, et en un large temps de calcul. Dans ce contexte, nous proposons un algorithme de décomposition, qui résout le problème en trois parties. Pour confirmer l'efficacité de cet algorithme, nous menons une analyse de performances où le temps de réponse et la qualité des solutions de l'algorithme sont examinés.

Sommaire :

4.1 Introduction

L'ordonnancement est un domaine très vaste dont l'expansion est continue au fil des temps. Compte tenu des différentes contraintes qui parviennent dans les systèmes de production, il devient de plus en plus difficile à établir les plans d'ordonnancement. Pour cette raison, une bonne modélisation du problème, en tenant compte des différents paramètres et contraintes du système, est nécessaire. Toutefois, pour parvenir à l'objectif cité ci-dessus, la machine ne doit pas être considérée comme l'unique ressource du système, et ne doit pas être considérée comme ayant des performances constantes et inchangeables. Dans ce cas, et premièrement l'exécution d'une tâche requiert, en plus de la machine, une ou très souvent plusieurs autres ressources comme les ressource humaines, les outils…etc. à savoir que la modélisation d'une contrainte de ressources humaines exige d'envisager la différence entre les compétences de celles-ci, où la notion de ressources qualifiées apparait. Deuxièmement, le temps opératoire d'une tâche se détériore en fonction des performances de la machines, qui elles sont influencées par le nombre de tâches exécutées précédemment. à partir de ces suppositions, nous avons formulé un problème qui n'a pas été étudié dans la littérature. Il s'agit d'un problème d'ordonnancement à machines parallèles avec effet de détérioration, sous contraintes de ressources humaines qualifiées et de ressources partagées, comme les outils. L'objectif d'étudier ce problème étant de minimiser la consommation totale de l'énergie et du makespan. La prise en compte de la consommation énergétique du système est due à son importance vis-à-vis de la théorie du développement durable et de la protection de l'environnement.

Le problème introduit ci-dessus peut avoir plusieurs applications ; comme le problème d'ordonnancement de bloque opératoire dans un hôpital, où les blocs opératoire sont les machines parallèles, les chirurgiens sont les ressources humaines qualifiées et un outil tel que le laparoscope est la ressource partagées, la détérioration dans cet exemple apparait non pas sur les machines, mais sur les ressources humaines. Il existe une autre application de ce problème, celle-ci fera l'objet d'un cas d'étude dans ce chapitre. L'application se trouve dans un atelier de poterie artisanale, situé à la commune de Nedroma, Tlemcen ville en Algérie. Grâce à quelques visites effectuées sur les lieux, en octobre 2020, et grâce à des questionnaires destinées aux gérants de l'usine, nous avons pu collecter certaines données, qui nous ont permis de formuler ce problème.

Premièrement nous avons constaté que l'atelier est de type flow-shop flexible parce que toutes les tâches passent par trois étages en séries, au niveau du 2ème étage nous trouvons plusieurs machines parallèles identiques. Cet atelier produit différents types de pièces de poterie, tel que Les Tadjine, Tandjia, vase, jarre-à-eau, les assiettes…etc. Le processus de production comprend trois étapes principales comme montrer dans la figure 4.1.

a) **Préparation de l'argile :** il existe plusieurs types d'argile utilisée dans la production de poterie ; certaines types d'argile sont importés de l'extérieur

de l'atelier, et sont déjà prêt à l'utilisation. Cependant, d'autres types d'argiles sont préparés dans l'atelier, pour convenir aux besoins spécifiques des artisans.

b) **Façonnage et décoration :** cet étage de la chaine comprend plusieurs machines en parallèles, les machines sont des tours potiers. L'exécution d'une tâche dans cette étape nécessite de l'argile pré-mélangée, un tour potier, un artisan, et des fois des outils pour les finitions et décorations. Les machines sont identiques, les artisans quant à eux, ils doivent être qualifiés et très perspicaces pour bien façonner la pièce, et avec l'épaisseur adéquate. Les artisans nécessitent un outil pour les finitions et la décoration de certaines pièces.

c) **Cuisson :** Cette étape finale consiste à porter les pièces façonnées à haute température, jusqu'à ce qu'elles durcissent.

Nous avons pu représenter les trois étapes principales de production dans l'atelier artisanal dans la figure 4.1, où nous détaillons les ressources nécessaires lors du façonnage des pièces de poterie.

Deuxièmement, nous avons constaté que l'étape du façonnage représente un goulot d'étranglement pour toute la chaine. Cela est dû à la complexité du processus de façonnage qui nécessite plusieurs ressources, parmi ceux des artisans qualifiés, dont la disponibilité est contrainte et cruciale à l'atelier.

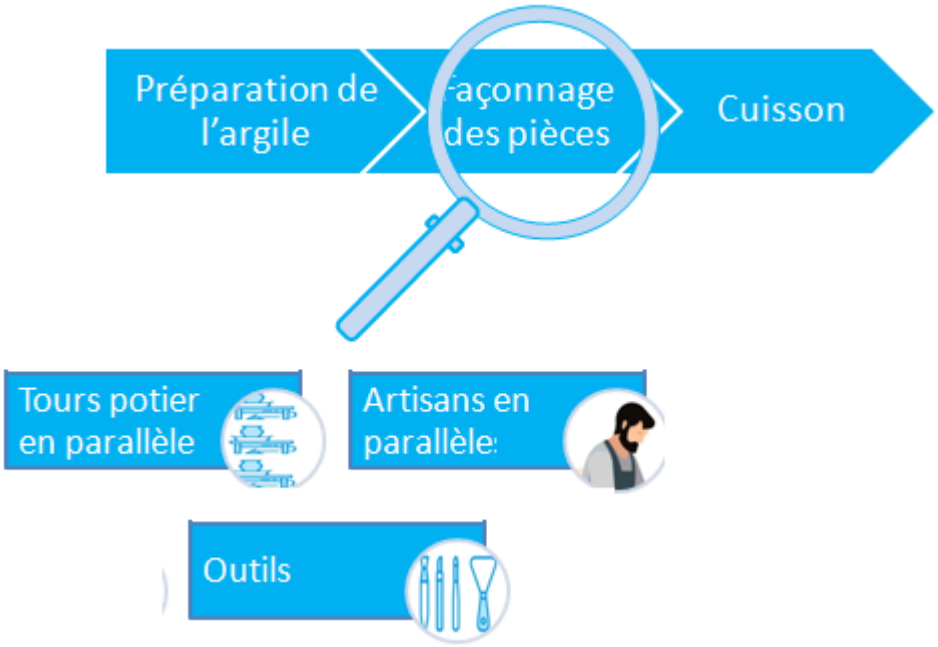


Figure 4.1 : Processus de fabrication de la poterie, et ressources nécessaire au façonnage des pièces.

Dans ce contexte, et pour optimiser l'exploitation des artisans disponibles, et leur fournir un milieu de travail plus approprié, nous modélisons le problème détecté

comme un problème d'ordonnancement à machines parallèles avec détérioration des machines, contraintes de disponibilité des ressources humaines et des d'outils. Les artisans travaillent en parallèle, chacun sur une des machines disponibles, ils ont des compétences différentes par rapport au type de la pièce à façonner ; c.à.d. un artisan « A » peut être plus rapide dans le façonnage d'une tâche qu'un autre artisan « B ». Dans le façonnage de certaines pièces, un artisan nécessite un outil pour les finitions ; par le mot « finitions », nous voulons dire que l'artisan n'occupe pas l'outil dès le début du façonnage, mais il l'occupe seulement pendant les dernières unités de temps. Cet aspect, dans l'utilisation des outils peut paraitre semblable à la notion existante des opérations-couplées, où une tâche est définie par deux opérations à exécuter consécutivement, abordée par (Potts and Whitehead, 2007) et (Meziani et al., 2019). Toutefois, l'intervention d'une ressource renouvelable, comme les outils, au milieu du traitement de la tâche n'a pas été considérée avant.

Le problème d'ordonnancement de machines parallèles classique consiste en l'affectation des tâches aux machines. Chaque machine ne peut traiter qu'une seule tâche, et une tâche doit être traitée par une seule machine sans préemption. Ce problème a été classé NP-difficile par (Lenstra, 1977), même pour la plus simple version de deux machines parallèles identiques. Il est à noter que cette version de problèmes classiques considère la machine comme unique ressource, contrairement à la version traitée dans ce chapitre, qui considère deux types de ressources renouvelables et qui est beaucoup plus complexe.

Ce chapitre est divisé comme suit : la prochaine section décrit le problème à traiter. La section qui la suive présente un modèle original, basé sur la programmation linéaire. La section 4 décrit la méthode utilisée dans la décomposition et la résolution du problème. La section 5 analyse les performances de la méthode approximative, précédemment présentée. Et dans la section 6, un exemple réel du cas d'étude est considéré et traité.

4.2 Description du problème

Le problème considéré dans ce chapitre peut être formulé comme suit : Il y a N tâches $j = \{j_1, j_2 \dots . j_N\}$ indépendantes à exécuter dans l'une des M machines parallèles et identiques, avec l'utilisation de l'un des artisans $r = \{r_1, r_2 \dots . r_R\}$ disponibles et un outil t. L'outil, s'il est nécessaire pour une tâche, il n'est occupé qu'après certaines unités de temps, cela est illustré dans la figure 4.2. Le temps opératoire d'une tâche dépend de sa position et dépend discrètement de l'artisan qui l'exécute, il est présenté dans l'équation (4.1). Ce problème peut être formulé par un problème d'affectation de tâches aux machines, d'allocation de ressources qualifiées aux tâches en ayant un contrôle sur le temps opératoire des tâches, et de séquençage de tâches dans la machine pour optimiser l'exploitation des outils.

$$Pr_j = \sum_{r=0}^{R} a_{jr}.Y_{jr} + \alpha.p_j \qquad (4.1)$$

Oùa_{jr}est le temps opératoire de la tâche j exécutée par l'artisan r. Y_{jr}décide de l'artisan qui exécute la tâche j. α est le taux de détérioration et $p_j$est la position où la tâche j est affectée.

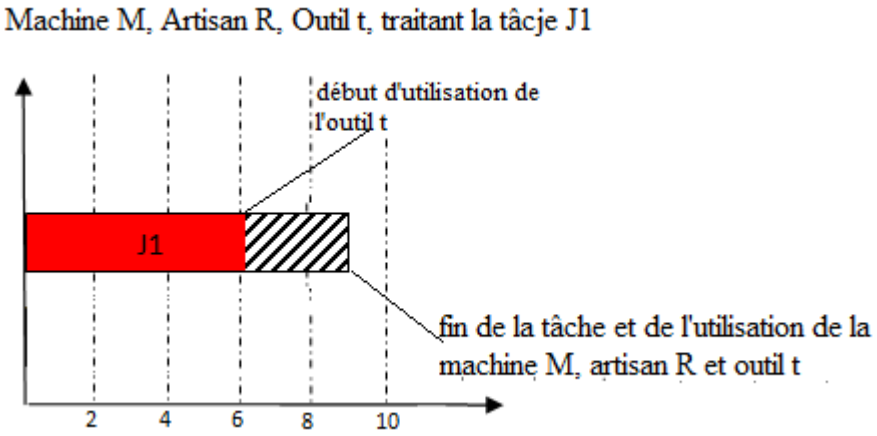


Figure 4.2 : L'utilisation d'un outil dans l'exécution de la tâche J1

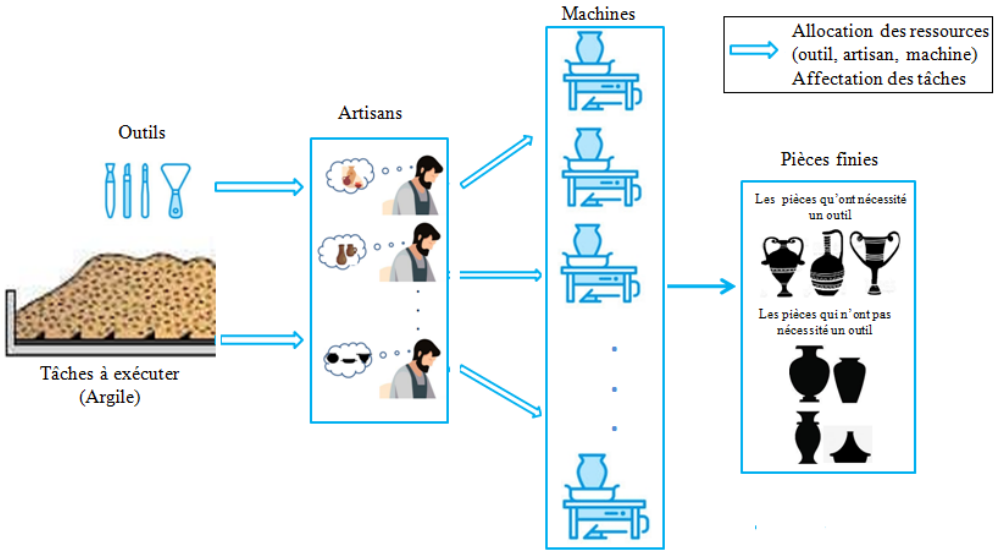


Figure 4.3 : L'affectation de tâches et l'allocation des artisans et outils aux machines

4.2.1 Les hypothèses

Pour pouvoir étudier le problème d'ordonnancement en question, nous avons considéré des hypothèses qui permettent la formulation du système de production dans le contexte du problème étudié. Ces hypothèses sont résumées dans la manière suivante:

- Les machines sont identiques ; les temps opératoire ne changent pas en fonction des machines.
- Une machine exécute une seule tâche à la fois.

- Les tâches ne sont préemptives, chacune doit être exécutée par une et une seule position d'une machine.
- Le temps opératoire d'une tâche est une fonction linéairement dépendante de la position de la tâche, et discrètement dépendante de l'artisan qui exécute la tâche
- Les artisans sont disponibles en nombres très limités
- L'exécution des tâches consomme une quantité d'énergie TEC qui dépend de la consommation de la machine et de la consommation publique.

4.2.2 Les objectifs

L'objectif de cette étude est de trouver la meilleure affectation des tâches aux artisans et aux machines, et la meilleure séquence des tâches, de tel sorte que deux critères soient minimisés ; Le makespan et le TEC. Le makespan est une fonction connue, largement considérée dans la littérature. Le TEC est moins considéré, il est formulé différemment d'une étude à une autre. Dans cette étude, nous considérons le TEC comme la somme de deux consommations : l'énergie consommée durant l'exécution d'une tâche, et la consommation publique.

4.2.3 Les contraintes

Les contraintes à considérer sont des contraintes de capacité de machine, contraintes de disponibilité des artisans et contraintes de disponibilité de l'outil.

Les contraintes de capacité de la machine formulent les limites d'une machine à traiter une et une seule tâche par position. Les contraintes de disponibilité des artisans restreignent l'utilisation des artisans au nombre disponible. Elles formulent aussi la capacité d'un artisan à traiter une et une seule tâche à la fois. Quant aux contraintes de disponibilité de l'outil, elles avancent ou reculent l'exécution d'une tâche jusqu'à ce que l'outil nécessaire à celle-ci soit disponible.

4.3 Modélisation mathématique « programmation linéaire du problème »

La modélisation mathématique est la méthode la plus adaptée dans les études des problèmes d'ordonnancement. Cela est en raison des avantage qu'elle fourni dans l'analyse ou la résolution de ce type de problème. Par ailleurs, cette méthode, pour des problèmes aussi complexe que celui considéré dans ce chapitre, peut s'avérée très consommatrice de capacité de calcul et de temps de calcul, à un point que le solveur devienne incapable de résoudre le problème même en utilisant le calculateur le plus performant qui existe. Cela était le cas d'un premier modèle proposé pour modéliser notre problème. Ce modèle était basé sur des variables quadruplement indexées, dont certaines étaient indexées par le temps. Etant donné que des variables étaient indexées par le temps et conditionnées par la date de début

de la tâche, le temps opératoire, date de début et de fin d'une tâche devrait être entiers, la raison pour laquelle seulement la partie entière de la détérioration et de la date du début d'utilisation de l'outil étaient considérées. Ce modèle ne pouvait résoudre même les plus petites instances du problème. Cela nous a motivé à développer un autre modèle, plus original et moins complexe, et dont le principe du fonctionnement diffère du premier. Ce modèle représente la contribution majeure de l'étude menée dans ce chapitre.

Les objectifs et contraintes littéralement définis ci-dessus vont être modélisés par des équations (égalité ou/et inégalité). La formulation de ces équations nécessite les données qui définissent le problème, les sorties qui définissent une solution donnée et les variables de décision qui eux permettent au modèle le choix de la ressource et de la machine qui traite la tâche, ainsi que la séquence de la tâche. Ces données, sorties et variables de décision sont définies dans les sections suivantes.

4.3.1 Les données

La modélisation mathématique du problème exige la formulation des paramètres qui définissent le système de production, soit les données qui sont définies comme suit :

a_{ir}	Temps opératoire normal d'une tâche i traitée par l'artisan r
T_i	L'outil nécessaire à la tâche i
Ts_i	Début d'utilisation de l'outil par la tâche i
N	Nombre de tâches i
M	Nombre de machines k
R	Nombre d'artisans r
T	Nombre d'outil o
A	Taux de détérioration
A	Un grand nombre
$\{w_1, w_2\}$	Le poids de la première et deuxième fonction, respectivement.
γ_1	L'énergie consommée par unité de temps pendant l'exécution des tâches.
γ_2	La consommation publique d'énergie, par unité de temps

4.3.2 Les sorties

La résolution d'un problème d'ordonnancement, par n'importe quelle méthode de résolution, signifie la fixation de certains paramètres qui permettent la formulation d'une solution. Dans notre étude, ces paramètres nommés les sorties sont définis comme le suivant :

T_{pr}	L'outil utilisé dans la position p de la machine k

Ts_{pr}	Le début d'utilisation de l'outil par position de ressources
P_{pr}	Le temps opératoire de la position p ressource r
d_{pr}	Le début de la position p ressource r
C_{pr}	La fin de la position p ressource r
P_{pk}	Le temps opératoire de position p machine k
d_{pk}	Le début de la position p machine k
C_{pk}	La fin de la position p machine k

4.3.3 Les variables de décision

Le modèle que nous avons développé est un modèle original, dont l'implémentation nécessite trois variables élémentaires : i)-Une variableX_{ipr} qui décide de l'affectation de chaque tâche à une position d'une ressource R (artisan). ii)-Une variable$Y1_{ptr}$qui décide de la séquence des tâches utilisant le même outil. iii)-Une variableY_{prk}qui décide de l'affectation de chaque tâche à une position d'une machine. Une autre variable est utilisée ;Xr_{pr}pour indiquer les positions occupées pour chaque artisan.

$$X_{ipr} = \begin{cases} 1\ if\ job\ i\ is\ processed\ in\ position\ of\ r \\ 0\ else \end{cases}$$

$$Xr_{pr} = \begin{cases} 1\ if position\ p\ of\ r\ is\ occupied \\ 0\ else \end{cases}$$

$$Y1_{por} = \begin{cases} 1\ if\ position\ p\ of\ r\ require\ tool\ o \\ 0\ else \end{cases}$$

$$Y_{prk} = \begin{cases} 1\ if\ position\ p\ of\ k\ require\ resource\ r \\ 0\ else \end{cases}$$

4.3.4 La fonction objectif

Le problème traité dans ce chapitre étant un problème d'optimisation multi-objectif, il est formulé par deux critères, soit le makepsan et la consommation totale en énergie « TEC ». Le premier critère qui est le makespan est modélisé par l'équation (4.2), le deuxième critère est modélisé par l'équation (4.3), qui elle même est définie par la somme des deux équations (4.4) et (4.5).

$$\text{Min F1= } C_{max} \tag{4.2}$$

$$\text{Min F2=TEC= } \gamma_1 . E_1 + \gamma_2 . E_2 \tag{4.3}$$

Tel que :

$$E_1 = \gamma_1 . \sum_{p=0}^{N} \sum_{k=0}^{M} Pr_{pk} \quad (4.4)$$

$$E_2 = \gamma_2 . C_{max} \quad (4.5)$$

Donc, l'équation (4.3) devient:

$$Min\ F2 = TEC = \gamma_1 . \sum_{p=0}^{N} \sum_{k=0}^{M} Pr_{pk}\ , a. + \gamma_2 . C_{max} \quad (4.3)$$

Où Pr_{pk} est le temps opératoire dans la position p de la machine k. $\gamma_1$et $\gamma_2$représentent la consommation d'énergie par unité de temps.

Il est à noter que les deux fonctions objectif du makespan et du TEC sont dépendantes du temps opératoire, ainsi, la deuxième fonction qui est le TEC, est dépendante même du makespan. Et là se pose une question sur l'importance de minimiser les deux fonctions au lieu d'une seule. Pour cela, nous rappelons l'analyse numérique réalisée dans le troisième chapitre, section 3.4.1, et nous remarquons que la question posée ici sur l'importance de minimiser les deux fonctions du makespan et du TEC, est la même question qui a été posée sur l'importance de minimiser le makespan et des coûts de ressources dont la consommation dépend du temps opératoire. Donc cette analyse numérique, dans la section 3.4.1, a prouvé que la négligence d'une ressource qui dépend du temps opératoire peut engendrer des consommations inutiles de cette ressource. À travers ces résultats, nous pouvons conclure que même dans l'étude menée dans ce chapitre, la négligence du TEC peut engendrer des consommations inutiles de l'énergie, et qui seront des coûts supplémentaires non-nécessaires sur l'entreprise.

Pour illustrer ce qui est dit, le même exemple que dans la section 3.4.1 est considéré, nous supposons que les tâches 2, 3 et 4 sont traitées par un artisan et les tâches 1 et 5 sont traitées par un autre artisan, nous supposons aussi qu'aucune tâche ne nécessite un outil dans son exécution. Nous résolvons l'exemple de deux manières, la première manière consiste à minimiser le makespan seulement, la solution est présentée dans la figure 4.4 (a), la deuxième manière consiste à minimiser les deux fonctions pondérées; makespan et TEC, la solution est présentée dans la figure 4.4 (b). Nous remarquons que dans la solution où seulement le makespan est minimisé, la fonction objectif égale à 25, et le TEC qui n'a pas été minimisé est calculé comme suit :

$$TEC = \gamma_1 . \sum_{p=0}^{N} \sum_{k=0}^{M} Pr_{pk}\ , a. + \gamma_2 . C_{max}$$

A.N :

1*25+1*(5+4.5+5)=25+14.5=39.5.

Tandis que dans la solution où le makespan et le TEC sont minimisés, les trois machines, le makespan reste à 25, mais le TEC diminue de 0.5 unité, il est calculé par :

$$TEC = \gamma_1 . \sum_{p=0}^{N} \sum_{k=0}^{M} Pr_{pk} , a. + \gamma_2 . C_{max}$$

A.N :

1*25+1*(5+4.5+4.5)=25+14=39

Une augmentation de 0.5 unité d'énergie peut paraitre négligeable, mais il est important de savoir que cette petite augmentation s'est arrivée dans une très petite instance, mais dans les cas réels où l'instance se mesure par plus que 50 tâches, l'augmentation dans la consommation d'énergie deviendra importante.

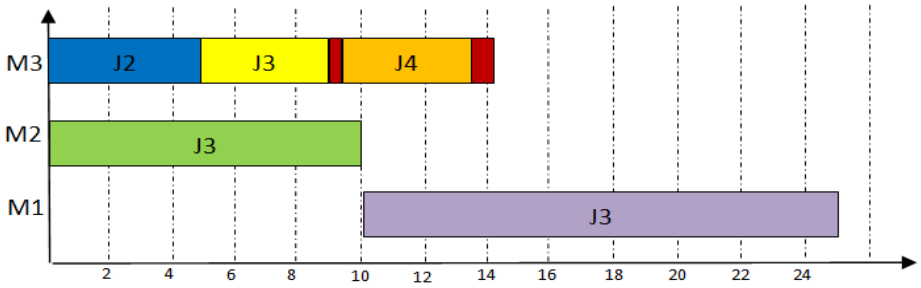


Figure 4.4 (a) Minimisation du makepsan seulement

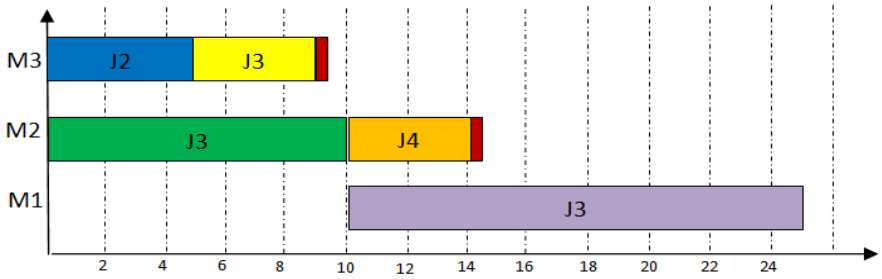


Figure 4.4(b) Minimisation du makespan et TEC pondérés

Figure 4.4 : Ladifférence entre la minimisation du makespan seulement et du makespan avec le TEC

4.3.5 Les contraintes

Dans cette section, nous formulons mathématiquement les contraintes représentées dans la section 4.2.3 de façon est ce que l'exécution du modèle mathématique soit possible. À travers cette formulation, nous pouvons combiner entre l'effet de détérioration et l'allocation dynamique des ressources « artisans et outils », sans perte de précision.

Avant d'expliquer le principe de fonctionnement du modèle, il est important de préciser qu'une contrainte dynamique assure que le programme vérifie qu'elle soit respectée, à unité de temps, cela requiert une grande capacité de calcul, mais reste faisable tant que les temps opératoire, temps de début et de fin des tâches sont entiers. En revanche, lorsque l'effet de détérioration intervient, les temps opératoire, de début et de fin des tâches ne sont plus entiers. Par conséquent, avec la considération d'un seul chiffre après la virgule, le programme est forcé de vérifié que la contrainte dynamique est respectée en chaque fraction de chaque unité, là, la

capacité de calcul nécessaire sera énorme et la résolution du problème sera presque impossible.

Pour éviter ce problème, nous avons développé un modèle qui affecte les tâches aux artisans et à l'outil de la même façon que lorsqu'il les affecte aux machines; par l'utilisation des contraintes de capacités. Par cela, le début d'une tâche traité par un artisan r, utilisant l'outil t est relié à la fin de la tâche précédemment traitée par le même artisan et à la fin de la tâche qui occupe l'outil t.

Les équations (4.6), (4.7) et (4.8) assurent qu'un artisan r traitant une tâche i ne soit pas alloué à une autre avant la finalisation de la tâche i. Ces équations attribuent à chaque artisan des positions successives dans le temps, et affectent chaque tâche à une seule position. Le temps de début et de fin de chaque position de chaque artisan sont définis par les équations (4.10), (4.11) et (4.17). Cette technique d'allocation dynamique des ressources "artisans" réduit remarquablement la capacité de calcul nécessaire. L'équation (4.9) définit le temps opératoire d'une tâche i traitée par un artisan r. De la même façon les tâches sont affectées chacune à une position de l'outil nécessaire. Les équations (4.12) et (4.13) définissent l'outil nécessaire dans la position p de la ressource r, et le temps de début de son utilisation. L'équation (4.14) spécifie si l'outil t est utilisé ou pas dans la position p de la ressource r. L'équation (4.15) assure que le temps de début d'utilisation de l'outil t dans la position p de la ressource r est supérieur au temps de fin de chaque position précédente de chaque ressource utilisant le même outil. Les équations (4.16) et (4.17) et (4.18) assurent que chaque position p d'une machine k traite au plus une tâche, et que chaque tâche soit traitée par une seule machine. L'équation (4.19) définit la valeur de la détérioration à considérer dans une position p d'une machine k. L'équation (4.20) définit le temps opératoire actuel d'une position p d'une machine k. L'équation (4.21), (4.22) et (4.23) définissent le temps de début et de fin d'une position p d'une machine k. L'équation (4.24) assure que les tâches qui utilisent le même artisan et le même outil ne soient pas superposées.

$$\sum_{i=0}^{N} X_{ipr} \leq 1 \qquad \forall\ p=1\ldots P,\ r=1\ldots R \qquad (4.6)$$

$$\sum_{p=0}^{P}\sum_{r=0}^{R} X_{ipr} = 1 \qquad \forall\ i=1\ldots N \qquad (4.7)$$

$$\sum_{i=0}^{N} X_{ipr} = Xr_{pr} \qquad \forall\ p=1\ldots P,\ r=1\ldots R \qquad (4.8)$$

$$\sum_{i=0}^{N} X_{ipr}.a_i = P_{pr} \qquad \forall p=1\ldots P,\ r=1\ldots R \qquad (4.9)$$

$$d_{1r} \geq 0 \qquad \forall\ p=1\ldots P,\ r=1\ldots R \qquad (4.10)$$

$$d_{pr} + P_{pr} = C_{pr} \qquad \forall\ p=1\ldots P,\ r=1\ldots R \qquad (4.11)$$

$$\sum_{i=0}^{N} X_{ipr}.T_i = T_{pr} \qquad \forall\ p=1\ldots P,\ r=1\ldots R \qquad (4.12)$$

$$\sum_{i=0}^{N} X_{ipr}.Ts_i = Ts_{pr} \qquad \forall\ p=1\ldots P,\ r=1\ldots R \qquad (4.13)$$

$$Y1_{por}.o + A.\left(\left(T_{pr} - o\right).\left(T_{pr} - o\right)\right) \geq T_{pr} \qquad \forall\ p=1\ldots P,\ o=1\ldots T,\ r=1\ldots R \qquad (4.14)$$

$$C_{bm}.Y1_{bom} \leq \left(d_{pr} + re_{pr}.P_{pr}\right) + A.(1 - Y1_{por}) \qquad \forall\ p=1\ldots P,\ o=1\ldots T,\ r=1\ldots R,\ b \leq p,\ m=1\ldots R,\ \{b,m\} \neq \{p,r\} \qquad (4.15)$$

$$C_{(p-1)r}.Xr_{pr} \leq d_{pr} \qquad \forall\ p=1\ldots P,\ r=1\ldots R \qquad (4.16)$$

$$\sum_{k=1}^{M} Y_{prk} \leq 1 \qquad \forall\ p=1\ldots P,\ r=1\ldots R \qquad (4.17)$$

$$\sum_{r=1}^{R} Y_{prk} \leq 1 \qquad \forall\ p=1\ldots P,\ k=1\ldots M \qquad (4.18)$$

$$\sum_{p=0}^{P}\sum_{k=0}^{M} Y_{prk} = \sum_{p=0}^{P} Xr_{pr} \qquad \forall\ r=1\ldots R \qquad (4.19)$$

$$\sum_{r=0}^{R}\sum_{b=0}^{p} Y_{brk}.\alpha \leq \det(p,k) \qquad \forall\ k=1\ldots M,\ p=1\ldots P,\ b=1\ldots p \qquad (4.20)$$

$$\sum_{r=0}^{R} P_{pr} + det_{pk}.Y_{prk} = P_{pk} \qquad \forall\ p=1\ldots P,\ k=1\ldots M \qquad (4.21)$$

$$\sum_{r=0}^{R} d_{pr}.Y_{prk} \leq d_{pk} \qquad \forall\ p=1...P,\ r=1...R,\ k=1...M \qquad (4.22)$$

$$C_{pk} = d_{pk} + P_{pk} \qquad \forall\ k=1\ldots M,\ p=1\ldots P \qquad (4.23)$$

$$C_{bm}.Y_{brm} \leq d_{pk} + A * (1 - Y_{prk}) \qquad \forall\ k=1\ldots M,\ p=1\ldots P,\ r=1\ldots R,\ m=1\ldots M,\ b<p \qquad (4.24)$$

$$d_{pk} \geq C_{(p-1)k} \qquad \forall\ k=1\ldots M,\ p=1\ldots P \qquad (4.25)$$

4.4 Agrégation des deux fonctions objectif

Avant de programmer de le modèle sur Lingo 10, nous agrégeons les deux fonctions en utilisant, comme le chapitre précédent, la méthode scalaire de la pondération présentée dans (Talbi, 2009), représentée par l'équation (4.26) :

$$f(x) = \sum_{i=0}^{n} w_i . f_i(x) \quad (4.26)$$

Tel que $w_i$sont les poids de chaque fonctionf_i, et $\sum w_i = 1$. Les deux fonctions objectif après l'agrégation deviennent une seule fonction représentée par l'équation (4.27)

$$OF = w_1 . C_{max} + w_2 . TEC \quad (4.27)$$

De l'équation (4.3), l'équation (4.27) devient comme l'équation (4.28)

$$OF = w_1 . C_{max} + w_2 . (\gamma_1 . \sum_{p=0}^{N} \sum_{k=0}^{M} Pr_{pk} + \gamma_2 . C_{max}) \quad (4.28)$$

Après quelques simplifications, la fonction multi objectif agrégée devient :

$$OF = (w_1 + w_2 . \gamma_2) . C_{max} + w_2 . \gamma_1 . \sum_{p=0}^{N} \sum_{k=0}^{M} Pr_{pk} \quad (4.28)$$

Tableau 4.1 Données de l'exemple 1

setups / **jobs**	a_{jr}		T_{pr}	Ts_j
	R_1	R_2		
J1	7	5	0	0
J2	7	6	0	0
J3	11	9	0	0
J4	5	2	1	0.8
J5	12	9	1	0.5
J6	6	6	1	0.9
$w_1 = 0.5, w_2 = 0.5, \gamma_1 = 1, \gamma_2 = 1$, **α=0.5**				

Exemple Illustratif: "exemple 1"

Pour illustrer le fonctionnement du modèle mathématique, nous considérons un exemple numérique de 6 tâches, 3 machines et 2 artisans. Les données de l'exemple sont résumées dans le tableau 4.1

La solution optimale est résumée dans la figure 4.5, où le makespan égale à 21 unités de temps et le TEC est calculé comme suit : TEC=41.5x1+21x1=62.5.

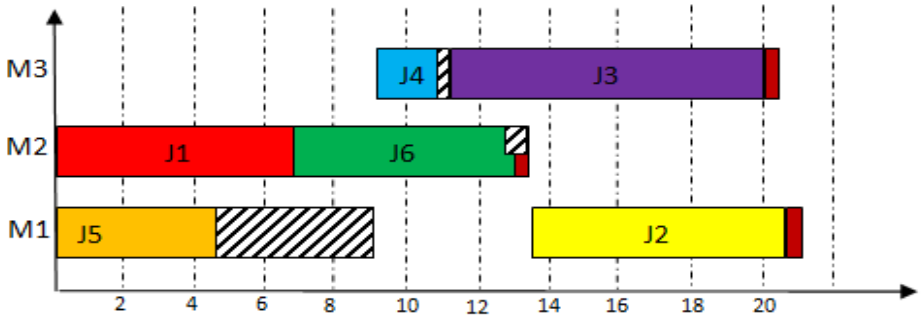


Figure 4.5 : Diagramme de Gantt représentant la solution de l'exemple 1

Ces résultats ont été obtenus par l'affectation des tâches J1, J6 et J2 à l'artisan R1, et les tâches J5, J5 et J3 à l'artisan R2, le séquençage des tâches ; J5 et J1 se traitent en parallèles vue que l'une nécessite un outil et l'autre n'en nécessite pas, ensuite, l'affectation de deux tâches à chaque machine. La résolution de ce problème a pris 37 minutes et 54 secondes.

Le problème d'ordonnancement considéré dans ce chapitre est un problème NP-difficile, dont la résolution exacte nécessite de grandes capacités de calcul. Malgré les efforts fournis pour le développement d'un modèle mathématique original et efficace, celui-ci reste limité à la résolution d'un ensemble restreint de petites instances, et avec un temps de réponse inabordable. Cela se voit dans l'exemple 4.1 qui malgré sa petite taille, il a fallu au solveur plus qu'une demi-heure à le résoudre. Cependant, l'adaptation d'une méthode approximative est une bonne alternative, et parmi les méthodes approximatives qui ont prouvé leur efficacité dans la résolution de ce type de problème sont les méthodes de décomposition, (Ovacik and Uzsoy, 1997b) et (Agnetis et al., 2014).

4.5 Développement d'une méthode de décomposition

L'ordonnancement est un outil de prise de décision à court terme, donc la résolution d'un problème d'ordonnancement doit se faire en temps réel. Sachant que le problème modélisé ci-dessus est classé parmi les problèmes d'optimisation NP-difficiles, les méthodes exactes, tel que le modèle mathématique, se trouvent incapables de le résoudre en temps réel, surtout lorsqu'il s'agit de résoudre des tailles importantes du problème. De très bonnes alternatives aux méthodes exactes, sont les méthodes approximatives, plus exactement les métaheuristiques. Et comme a été remarqué dans le deuxième chapitre et prouvé dans le troisième chapitre, les algorithmes de décomposition sont des approches très efficaces dans la résolution des problèmes d'ordonnancement non-spécifié avec contraintes de ressources; problème d'affectation de tâches et d'allocation de ressources. Donc, nous développons, dans cette section, un algorithme de décomposition basé sur la métaheuristique du recuit simulé.

La résolution du problème considéré consiste en l'allocation des artisans aux tâches, ensuite l'affectation des tâches aux machines, et la fixation les temps de

début et de fin pour chaque tâche et selon la disponibilité des artisans et machines, finalement le séquençage des tâches selon l'utilisation de l'outil t. Par conséquent, le problème est décomposé en trois sous-problèmes :

- **Le sous-problème d'allocation d'artisans** : Ce sous-problème consiste en l'allocation des artisans à chaque tâche, de façon à minimiser le makespan sur les artisans.
- **Le sous-problème d'affectation de tâches** : Dans cette partie, les artisans sont déjà alloués chacun à une tâche, donc ce sous-problème consiste en l'affectation des tâches aux machines de façon à minimiser le makespan des machines et la consommation totale de l'énergie.
- **Le sous-problème de séquençage des tâches** : Ce sous-problème consiste à revoir l'emplacement de chaque tâche de façon à optimiser l'exploitation des ressources partagées « outils » et donc minimiser le makespan et le TEC.

4.5.1 Résolution du premier sous-problème: "Allocation des artisans aux tâches"

Pour la résolution du sous problème d'allocation des artisans aux tâches, quelques paramètres doivent être pris en considération, notamment le temps opératoire des tâches qui varie de manière considérable d'un artisan à un autre. Cet aspect de variation dans le temps opératoire d'un artisan à un autre nous permet de considérer le sous-problème comme un problème d'ordonnancement à machines parallèles non-uniformes. Celui-ci s'avère un problème d'optimisation NP-difficile, dont aucune méthode exacte ne peut le résoudre efficacement. Donc nous développons un recuit simulé pour le résoudre. Dans cette étape de l'algorithme de décomposition, le recuit simulé minimise le makespan des artisans.

- **L'algorithme du recuit simulé:**

Le recuit simulé a été défini dans la section 2.4.2.2, et détaillé dans la section 3.5. Dans ce chapitre, nous conservons l'utilisation de cette métaheuristique pour résoudre le premier sous-problème, qui est l'allocation des artisans aux tâches.

L'utilisation et l'adaptation de cette métaheuristique à un problème nécessite la fixation des paramètres relatifs au problème, ces paramètres sont la solution initiale et la technique de génération du voisinage, ils sont illustrés dans la figure 2.2 du deuxième chapitre.

- La solution initiale: le sous-problème considéré dans cette sous-section peut être considéré comme un problème d'ordonnancement de machines parallèles non-uniformes, où les artisans sont comparés aux machines. De l'étude de (Villa et al., 2018) la solution initiale de l'algorithme 2 « multi-pass heuristics based on assignment » est formée par l'affectation de chaque tâche à la machine la plus rapide. Cependant, en suivant le même principe, nous affectant chaque tâche à l'artisan qui la traite le plus rapidement.

– Génération de la solution voisine : le chromosome est défini par le tableau 4.2, où les lignes sont les artisans et les colonnes sont les tâches. Un mouvement de génération d'une solution voisine consiste en l'extirpation d'une tâche, aléatoirement choisie, de l'artisan qui a le plus grand temps d'achèvement, et l'affectation de cette même tâche à l'artisan qui a le plus petit temps d'achèvement.

Quant aux autres paramètres; la température initiale, le taux de refroidissement, la probabilité d'acceptation et le critère d'arrêt, ils sont tous pris du chapitre III. Le pseudo code de l'algorithme utilisé dans cette étape est le même que le pseudo-code de **l'algorithme 2**, chapitre III, section 3.5.

Tableau 4.2 Représentation d'une solution dans l'algorithme du recuit simulé

R / Job	**J1**	**J2**	**J3**	**J4**	**…**	**Jn**
R1						
R2		X	X			X
…						
Rn	X			X		

4.5.2 Résolution du deuxième sous-problème: "Affectation des tâches aux machines"

L'affectation des tâches aux machines ne serait pas un problème si le temps opératoire ne dépendait pas de sa séquence dans la machine. Cela dit que chaque artisan occuperait une machine, et les autres machines resterons inoccupées jusqu'à la fin de l'ordonnancement, Cette propriété se trouve dans le problème considéré par (Abdeljaoued et al., 2018). Mais dans le problème que nous considérons le temps opératoire augmente par effet de détérioration selon la séquence de la tâche dans la machine. Donc, pour minimiser l'effet de la détérioration sur le makespan et sur la fonction de l'énergie, il est important d'équilibrer entre les nombres de tâches traitées par chaque machine, cela a été prouvé dans le chapitre III (qui a fait preuve d'une publication (Sekkal and Belkaid, 2020)). Ainsi, du moment où les temps opératoire générés ne sont pas largement dispersés (cela est illustré dans la section 6.4.1), y a moins de risque que le makespan soit gonflé par une tâche ayant un temps opératoire aberrant. Malgré cela, les temps d'achèvement des machines sont vérifiés lors de l'affectation de chaque tâche, une technique est implémentée pour sélectionner entre une machine ayant le plus petit temps d'achèvement et une autre ayant le moins de charge.

4.5.3 Résolution du troisième sous-problème: "Séquençage des tâches"

La résolution de ce problème consiste en le séquençage des tâches de façon à ce que l'utilisation de l'outil t soit optimisée. Du moment où l'outil est utilisé à la fin du traitement de certaines tâches, la séquence des tâches influence l'exploitation des

outils et donc des artisans et des machines. Pour cette raison, nous utilisons une recherche locale afin de trouver la séquence des tâches de façon à maximiser le taux d'utilisation de l'outil.

La recherche locale permet de trouver, parmi un espace de séquences aléatoirement généré, celle qui minimise le makespan et l'énergie. Pour chaque séquence générée, nous adoptons une technique de positionnement de tâches de façon à ce que le temps de début d'utilisation de l'outil pour une tâche soit harmonisé avec le temps de fin d'utilisation de l'outil par la tâche précédente. Cela est mathématiquement formulé par: $d_{jt} + Ts_j . P_{jt} \geq f_{it}$ tel qued_{jt}est le temps de début de la tâche j qui nécessite l'outil t, f_{j-1}est la date de fin de traitement de la tâche i qui a débuté avant la tâche j et utilisé l'outil t. Donc le temps à partir duquel l'outil peut être réutilisé par la tâche j est défini par: f_i, et le temps de début de la tâche j utilisant l'outil t est: $d_{tj} = f_{it} - Ts_j . P_j$. En prenant en considération la solution définie par le recuit simulé, une tâche j doit avoir un temps de début selon l'artisan qui l'exécuted_{rj}, un temps de début selon sa position dans la machined_{pkj}, et un temps de début selon sa séquence par rapport à l'outil qu'elle nécessited_{tj} Donc le temps de début d'une tâche j exécutée par l'artisan r, dans la position p de la machine k utilisant l'outil t est: $d_{jrpkt} = max(d_{tj}, d_{rj}, d_{pkj})$.

L'implémentation de cette heuristique nécessite l'initialisation de quelques paramètres:$n_R$le nombre de tâches traitées par chaque artisan, $List_1$est la liste des tâches qui ne nécessitent pas d'outil pour leur exécution. $List_t$ est la suite de listes des tâches qui nécessitent l'outil t pour leur exécution, tel que t=1…T, avec T le nombre d'outils.

Les deux sous-problèmes (l'affectation de tâches aux machines et et de séquençage des tâches) sont résolus par **l'algorithme 6.**

Algorithme 6: Heuristique du séquençage des tâches

Initialiser $N, M, R, n_R,$ $\text{List}_1, \text{List}_t$, T
Mettre à jour n_R pour chaque artisan R
Commencer par la machine m=0
  Si $n_R > \frac{N}{M}$
    Affecter$\frac{N}{M}$ tâche à la machine m.
    Mettre à jour m=m+1
    Pour chacune des tâches restantes :
      Sélectionner m_{min}machine avec temps d'achèvement minimal $C_{m_{min}}$
      Sélectionner m_{charge}machine avec charge minimale, et temps d'achèvement$C_{m_{charge}}$
      Si : $C_{m_{charge}} - C_{m_{min}} > det.charge$
        Affecter la tâche à la machine m_{min}
      Sinon
        affecter la tâche à la machine m_{charge}

FinSi
FinSi
Si $n_R < \frac{N}{M}$
Affecter n tâches de l'artisan R à la machine et compléter par les tâches non affectées des autres artisans.
Mettre à jour m=m+1
FinSi
Mettre à jour $P_{jpk}, d_{jpk}, C_{jpk}$
Après chaque affectation vérifier :
Si $d_{jpk} < d_{rj}$:
$d_{jpk} = d_{rj}$
Si $d_{jpk} > d_{rj}$
$d_{rj} = d_{jpk}$
Et mettre à jour tous les d_{Jr} tel J=j…nT.
FinSi
Recherche locale
Tant que (L<50), refaire :
Appliquer un changement sur la séquence pré-requise, pour obtenir une solution voisine S'
Affecter les tâches avec $Ts_j = 0$ à$List_1$
Affecter les tâches avec Tsj>0 à une liste des$List_t$, selon l'outil nécessaire à la tâche j.
Mettre à jour d_{jt}, C_{jt}
Après chaque affectation, vérifier :
Si $d_{jt} < d_{jpk}$
$d_{jt} = d_{jpk}$
FinSi
Si $d_{jt} > d_{jpk}$
$d_{jpk} = d_{tj}$
FinSi
Et mettre à jour tous les d_{Jpk} tel J=j…N.
Calculer la fonction objectif OF
Si OF(S')<OF(S)
Accepter S' et mettre à jour S=S'
Mettre à jour L : L=L-1 ;
Sinon
Rejeter S' et garder S=S.
Mettre à jour L : L=L+1 ;
FinSi
Fin

Pour illustré le fonctionnement de l'algorithme de décomposition, nous résolvons le problème proposé dans l'exemple 4.1 par cet algorithme, présentons la solution dans la figure 4.6. Il est remarquable que l'algorithme affecte les tâches J1, J2 et J6 à l'artisan R_1, et les tâches J3, J5 et J4 à l'artisan R_2. Ensuite il affecte deux tâches à chaque machine et séquence les tâches de façon à ce que deux tâches utilisant le même artisan et le même outil ne soient pas superposées dans le temps.

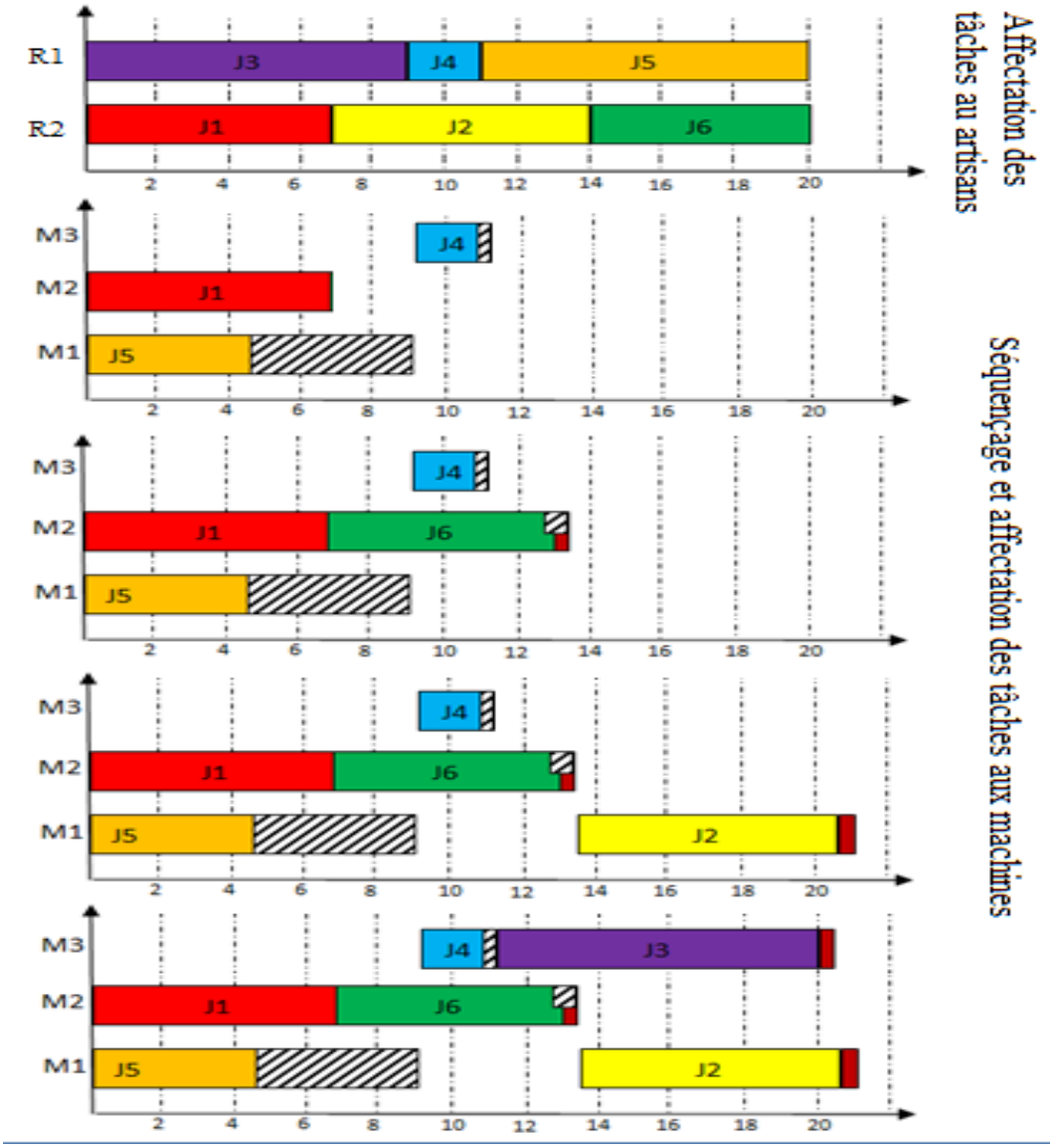


Figure 4.6 : La solution obtenue par l'approche de décomposition

4.6 Analyse de performance de la méthode de décomposition

Dans cette section, nous analysons les performances de l'algorithme de décomposition en le comparant avec le modèle mathématique pour les petites instances, et avec une borne inférieure pour les moyennes et grandes instances.

4.6.1 Génération des instances

Les instances utilisées dans la comparaison entre le modèle mathématique et l'algorithme sont inspirées de l'étude (Agnetis et al., 2014) et (Abdeljaoued et al., 2018) et du cas d'étude de la poterie artisanale. La génération des instances et le choix des paramètres du modèle sont détaillés comme suit:

Le nombre de machines varie de 3 à 30 comme suit: M={3, 5, 10}. Chacun de ces nombres de machines est combiné à différents nombres de ressources

"artisans". Nous ne considérons pas le cas où le nombre de ressources est plus grand que celui des machines, parce qu'il n'existe pas dans le cas réel; les artisans sont trop précieux tandis que les machines sont disponibles dans l'atelier et dans le marché à des prix raisonnables. Donc, nous identifions deux classes: Classe (A): lorsque R=M et Classe(B) lorsque le nombre de ressources R est légèrement inférieure au nombre des machines M. Quant aux nombres des tâches, ils sont générés par classes; comme les trois batteries de testes considérées dans (Abdeljaoued et al., 2018): Classe(1) 3 tâches par artisan, Classe(2) 5 tâches par artisan et Classe(3) 10 tâches par artisan.

À partir du cas d'étude de la poterie artisanale, les temps opératoire des tâches sont légèrement dispersés; deux intervalles Ier1 et Ier2 sont définis pour générer les temps opératoire minimaux à partir d'une fonction de distribution uniforme. Ier1=[2, 6], et Ier2=[4, 8]. Le temps opératoires des tâches qui sont traitées par les artisans les moins qualifiés sont définis par l'ajout d'une portion de temps δ aux temps opératoire minimaux. δ est définie aléatoirement par une fonction de distribution uniforme U[0, a], tel que "a" est défini par le logarithme népérien du nombre des artisans: a=Ln(R). Ce "a" définit la différence entre les compétences des artisans, il est formulé ainsi parce qu'un grand nombre d'artisans disponibles signifie que chaque artisan est spécifié à des tâches précises et est beaucoup moins qualifié pour les autres tâches. Tandis qu'un petit nombre d'artisans disponibles signifie que les artisans sont expérimentés et qu'ils acquièrent des compétences polyvalentes.

La date de début d'utilisation Ts_j de l'outil dépend de la tâche, elle est fixée généralement durant les dernières unités de temps de l'exécution de la tâche. Donc, elle est généré aléatoirement par une fonction de distribution uniforme U [0.5, 1], lorsque la valeur de Ts_j est de 1, cela veut dire que l'outil n'est pas utilisé. Le nombre des différents outils est généré selon le nombre de tâches ayant un $Ts_j < 1$. Autrement dit, pour chaque ensemble de tâches ayant un $Ts_j < 1$ un outil spécifique est disponible. L'ensemble de tâches est généré aléatoirement par une fonction uniforme U [5, 10]. Le taux de détérioration est commun à toutes les machines et toutes les tâches, il est généré aléatoirement par une fonction de distribution uniforme U[0, 1]. Les poids des deux fonctions objectif sont pris comme suit: $\{w_1, w_2\} = \{0.3,0.7\}, \{0.5, 0.5\} and \{0.7,0.3\}$. Ces instances sont résumées dans le tableau 4.3.

Une instance est définie par la combinaison (M, R, N, P=U [a, b]): i)-nombre de machines, ii)-nombre de ressource, iii)-nombre de tâches, iv)-intervalle du temps opératoire. Nous avons 3 différents nombres de machines, 2 nombres de ressources pour chaque machine, 2 différents intervalles du temps opératoire, et 3 classes pour le nombre de tâches. Toutefois, pour le dernier nombre de machines ; M=10, seulement deux classes pour le nombre de tâches sont considérées. Le nombre de combinaisons est donc de (3x2x2x3)-1x2x2=32 combinaisons. L'algorithme est codé dans java, les testes sont fait par un ordinateur Samsung 4GB avec un processeur i3.

4.6.2 Proposition d'une borne inférieure

Pour analyser les performances de l'algorithme pour les moyennes et grandes instances, nous développons une borne inférieure, qui sert comme référence pour s'assurer de la qualité des solutions fournies.

Tableau 4.3 Résumé des instances générées pour l'analyse numérique

The instances	Machines	Resources
Small	M=3	R=3
		R=2
Medium	M=5	R=5
		R=3
Large	M=10	R=10
		R=8

X

Number of jobs by craftsman, their processing time and their tool's use-start-time						
Pm	U [2, 6]			U [4, 8]		
classes	3 tâches/R	5 tâches/R	10 tâches/R	3 tâches/R	5 tâches/R	10 tâches/R
	I1	I2	I3	I4	I5	I6
	I7	I8	I9	I10	I11	I12
	I1	I2	I3	I4	I5	I6
	I7	I8	I9	I10	I11	I12
	I1	I2	I3	I4		
	I5	I6	I7	I8		

La formulation d'une borne inférieure doit prendre en considération les trois contraintes principales; la contraintes de disponibilité des artisans, la contrainte de disponibilité de l'outil, la contrainte de détérioration. La considération de ces trois contraintes permet la construction de la borne inférieure, en commençant par le makespan des artisans, ensuite le décalage causé par les outils et finalement, le makespan des machines. Puisque les artisans travaillent en parallèle, le makespan des artisans est formulé par la somme des temps opératoire de toutes les tâches, divisée sur le nombre d'artisans. Le décalage dans le temps causé par les outils est formulé en deux étapes : i) lors de la première étape nous calculonsST_t l'ensemble des sommes des temps d'utilisation pour chaque outil, équation (4.29). ii) lors de la deuxième nous sélectionnons la plus grande somme$\overline{ST}$ parmi lesST_t, équation (4.30). Quant à la détérioration, nous supposons que sont effet est minimisé par l'affectation du même nombre de tâches à chaque machines. La borne inférieure du makespan est donc représentée par l'équation (4.31), et celle du TEC est représentée par l'équation (4.32). La borne inférieure de la fonction objectif sera donc la somme pondérée des deux fonctions$LB(Cmax)\ et\ LB(TEC)$, équation (4.33).

$$ST_t = \sum_j (1 - Ts_{tj}) \,.\, Pmin_{tj} \qquad t=1\ldots T \qquad (4.29)$$

$\overline{ST}$=max(ST_t) t=1…T (4.30)

$$LB(Cmax) = \frac{\sum_j (Ts_j . P_{min})}{R} + \overline{ST} + (\frac{N}{M} - 1).\alpha \quad (4.31)$$

$$LB(TEC) = \gamma_1 . \sum_j P_{min} + \alpha . \left(\frac{N}{M} - 1\right) + \gamma_2 . LB(Cmax) \quad (4.32)$$

$$LB(OF) = w_1 . LB(Cmax) + w_2 . LB(TEC) \quad (4.33)$$

4.6.3 Configuration des expériences

L'implémentation d'une méthode de résolution exacte ou approximative efficace passe primordialement par une analyse de ses performances. Ces performances se mesurent par le temps de réponse de la méthode et/ou par la qualité de solutions qu'elle fournies. Conséquemment, nous résumons le temps de calcul du modèle mathématique et de l'algorithme dans le tableau 4.4. Ensuite, nous examinons la qualité des solutions fournies par l'algorithme proposé de deux méthodes ; la première méthode est le calcul du décalage « GAP 1» des solutions de l'algorithme par rapport aux solutions du modèle mathématique, le « Gap1 » est calculé par l'équation 4.34. Et parce que le modèle mathématique ne fourni de solutions que pour certaines petites instances, la deuxième méthode est le calcul du décalage « Gap2 » des solutions de l'algorithme par rapport à la borne inferieure proposée, le « Gap2 » est calculé par l'équation 4.35. En revanche, la borne inférieure elle-même ne fourni pas de solutions exactes, et aucune méthode n'existe pour confirmer son efficacité. Cependant, nous mesurons le décalage « Gap3 » de la borne inférieure par rapport aux solutions exactes du modèle mathématique (pour les instances qui peuvent être résolues par le modèle mathématique). Ce « Gap3 » servira de repère pour analyser la qualité des solutions de l'algorithme selon leur décalage de la borne inférieure, le « Gap3 » est calculé par l'équation (4.36).

$$Gap1 = \frac{\left(\text{Solution(Algorithme)} - \text{Solution(MILP)}\right) * 100}{\text{Solution(MILP)}} \quad (4.34)$$

$$Gap2 = \frac{(\text{Solution(Algorithme)} - \text{LB}) * 100}{\text{LB}} \quad (4.35)$$

$$Gap3 = \frac{\left(\text{LB} - \text{Solution(MILP)}\right) * 100}{\text{Solution(MILP)}} \quad (4.36)$$

4.6.4 Résultats et discussion

L'objectif de mener tous ces expériences est, premièrement d'examiner les performances de l'algorithme proposé, et deuxièmement d'analyse son comportement pour les différents poids des deux fonctions objectif$(w_1, w_2) = \{(0.3, 0.7), (0.5, 0.5), (0.7, 0.3)\}$. Cependant, pour l'analyse de performances, nous comparons, à partir du tableau 4.4, le temps de calcul du modèle mathématique et le

temps de calcul de l'algorithme. Ensuite, nous résumons les décalages dans la figure 4.7, figure 4.8, figure 4.9, figure 4.10 et la figure 4.11 ; « Gap1 », pour les petites instances est représenté dans la figure 4.7, « Gap2 » pour les petites instances est représenté dans la figure 4.9, « Gap2 » pour les moyennes grandes instances dans les figures 4.10 et 4.11, respectivement, et « Gap3 » dans la figure 4.8.

Tableau 4.4 Comparaison entre les temps de réponse du MILP et de l'algorithme

Taille de l'instance	**L'instance**	**Le temps de réponse de la méthode**		
		MILP	Algorithme	
			Moyen	Max
Petites instances	I1 et I4	De 10min à 2h	0.2 sec	0.5 sec
	I2	>20h	0.4 sec	0.5 sec
	I3, I5 :12	-	1.1 sec	1.5 sec
Moyennes instances	I1 : I12	-	2.2 sec	2.9 sec
Grandes instances	I1 :I12	-	35 secs	50 secs

À partir du tableau 4.4 nous remarquons que le temps de calcul du modèle mathématique est très grand, et il s'accroît exponentiellement en fonction de la taille du problème, il peut aller de 10 minutes jusqu'à 2 heures pour résoudre un problème de 3 machines, 2 artisans et 6 tâches, et il dépasse les 20 heures pour résoudre un problème de 3 machines, 2 artisans et 10 tâches. Tandis que le temps de calcul de l'algorithme de décomposition est relativement stable, sachant que, pour un problème de 3 machines, 2 artisans et 6 tâches, il ne dépasse pas 0.5 seconde, et pour un problème de 3 machines, 2 artisans et 10 tâches, il ne dépasse pas 1.5 seconde, en ce qui concerne les petites instances. Pour les moyennes instances, où le modèle mathématique perd ces capacités à résoudre le problème, le temps de calcul de l'algorithme de décomposition ne dépasse pas les 2.9 secondes, il est moyennement égale à 2.2 secondes. Quant aux grandes instances, le temps de calcul de l'algorithme augmente remarquablement, mais sans dépasser les 50 secondes.

À partir de la figure 4.7, nous remarquons que le « Gap1 », qui représente le décalage en pourcentage de l'algorithme par rapport au modèle mathématique est relativement faible. Il ne dépasse pas le 1.6%, et est nul pour le quatrième exemple de l'instance I1, pour l'instance I2, et pour le cinquième exemple de l'instance I4. Ces valeurs du « Gap1 » confirment le bon fonctionnement de l'algorithme de décomposition, et ils assurent son efficacité pour ces petites instances.

Pour chaque instance où le « Gap1 » est non-nul, nous remarquons une augmentation dans sa valeur quand $w_2 > w_1$. Autrement dit, pour un exemple d'une instance donnée, le « Gap1 » atteint sa valeur maximale lorsque $(w_1, w_2) = (0.3, 0.7)$, il diminue lorsque $(w_1, w_2) = (0.5, 0.5)$, et il atteint sa valeur minimale lorsque $(w_1, w_2) = (0.7, 0.3)$. Néanmoins, quelque soit les valeurs des poids

(w_1, w_2) le décalage des solutions de l'algorithme par rapport aux solutions optimales reste faible.

Dans la figure 4.8, nous résumons le « Gap3 », qui représente le décalage de la borne inférieure par rapport au modèle mathématique. Ce « Gap3 » nous permet, premièrement, d'examiner l'efficacité de la borne inférieure, deuxièmement, il nous servira comme repère pour analyser la qualité des solutions de l'algorithme de décomposition. Toutefois, nous constatons que ce « Gap3 » varie d'un exemple à un autre, de 1% à 11%, sachant que la taille des exemples est de 3 machines, 2 artisans, 6 tâches, mis à part l'exemple de l'instance 2 qui contient 10 tâches. Cette variation signifie que la borne inférieure peut être proche de l'optimum, pour certaines instances, comme elle peut être loin de l'optimum, pour d'autres instances.

De la figure 4.9, nous décelons une variation considérable entre les valeurs du « Gap2 » des différentes petites instances. Pour I1 et I4, où seulement 6 tâches sont à ordonnancer, le « Gap2 » ne dépasse pas les 5%. Tandis que pour I9 et I12, où les exemples contiennent 30 tâches, le « Gap2 » dépasse les 15% et atteint les 20%. Quant aux « Gap2 » des moyennes instances, représentés dans la figure 4.10, nous décelons presque la même variation dans les valeurs du décalage en pourcentage, tel que le « Gap2 » est relativement faible pour les instances où seulement 3 tâches sont générées par artisan ; I1 et I4, et il est plus important pour les instances où 10 tâches sont générées par artisan ; I9 et I12. Autrement dit, la marge d'erreur de l'algorithme par rapport à la borne inférieure augmente en fonction de la taille de l'instance ; à savoir le nombre d'artisans et le nombre de tâches à exécuter.

Les résultats du décalage en pourcentage de l'algorithme de décomposition par rapport à la borne inférieure, pour les grandes instances, sont représentés dans la figure 4.11. De cette figure, nous remarquons que le « Gap2 » pour les grandes instances prend des valeurs plus importantes, comparées au « Gap2 » des petites et moyennes instances. Cependant, les mêmes variations remarquées pour les petites et moyennes instances sont remarquées pour les grandes, à savoir la valeur du « Gap2 » augmente pour les instances ayant un nombre de tâches plus grand.

Les valeurs du « Gap2 » varient en fonction de la taille de l'instance en question, que ce soit pour les petites, moyennes ou grandes instances, et varient aussi en fonction des valeurs que prennent les poids(w_1, w_2). Autrement dit, le « Gap2 » augmente lorsque$(w_1 < w_2)$ et diminue lorsque$(w_1 > w_2)$.

La variation dans les valeurs du « Gap1 » ou du « Gap2 » qui est en fonction des poids (w_1, w_2) s'explique par le fait que la valeur numérique de la deuxième fonction « TEC » est beaucoup plus importante que la valeur numérique de la première fonction objectif qui est le makespan, donc lorsqu' un décalage dans le makespan se mesure par une ou deux unités, le décalage dans le TEC se mesure par une dizaine d'unités. En accordant un poids supérieur au TEC, nous rehaussons son décalage par rapport à l'optimum. Cependant, malgré l'augmentation des valeurs du « Gap2 » pour certaines instances, nous pouvons toujours dire que les solutions sont

satisfaisantes étant donné la complexité élevée du problème, et sachant que la borne inférieure elle-même peut être décalée de l'optimum, surtout pour les instances de grandes tailles où l'utilisation des outils est très fréquente.

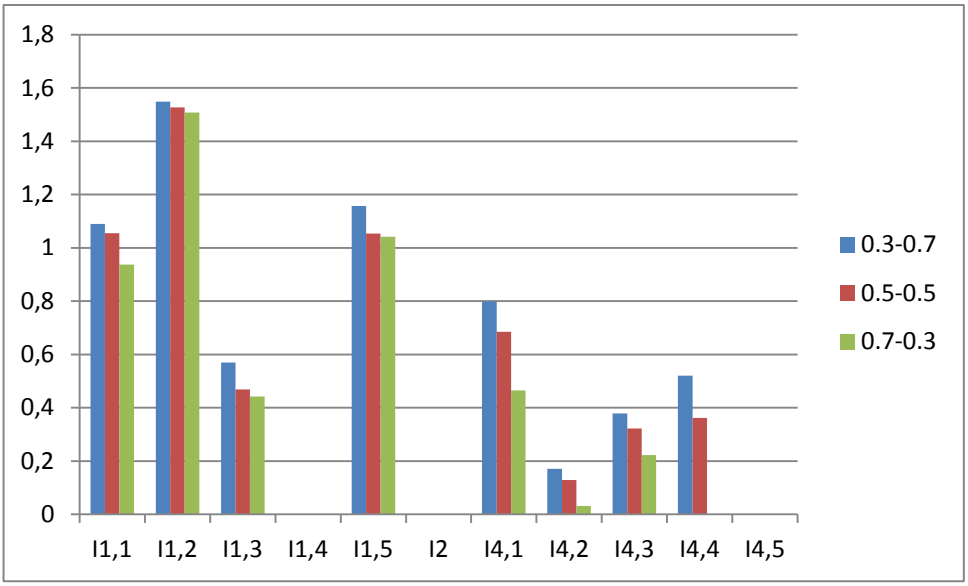


Figure 4.7 : « Gap1 » le pourcentage de décalage de l'algorithme par rapport au MILP, pour quelques petites instances

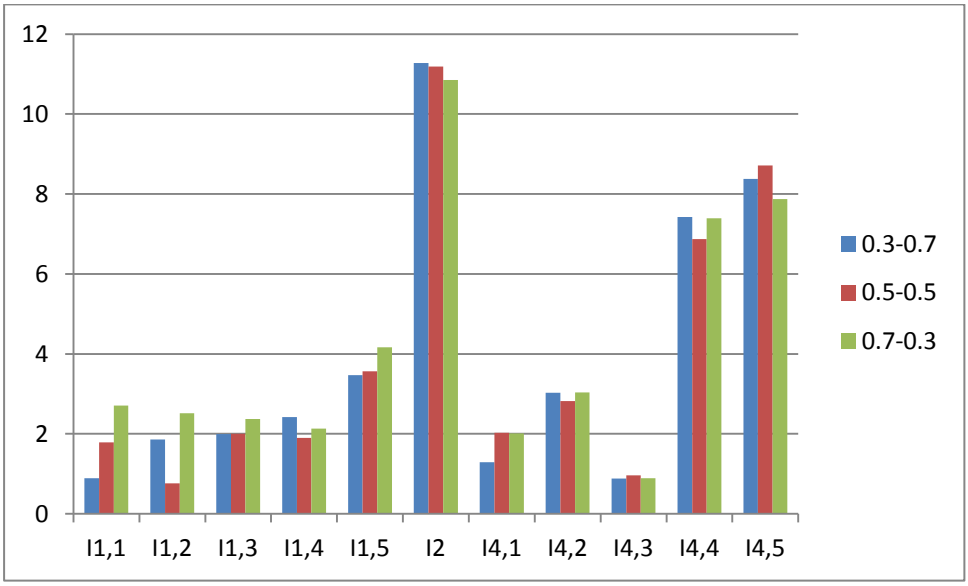


Figure 4.8 : « Gap3 » le pourcentage de décalage de la borne inférieur par rapport au MILP, pour quelques petites instances

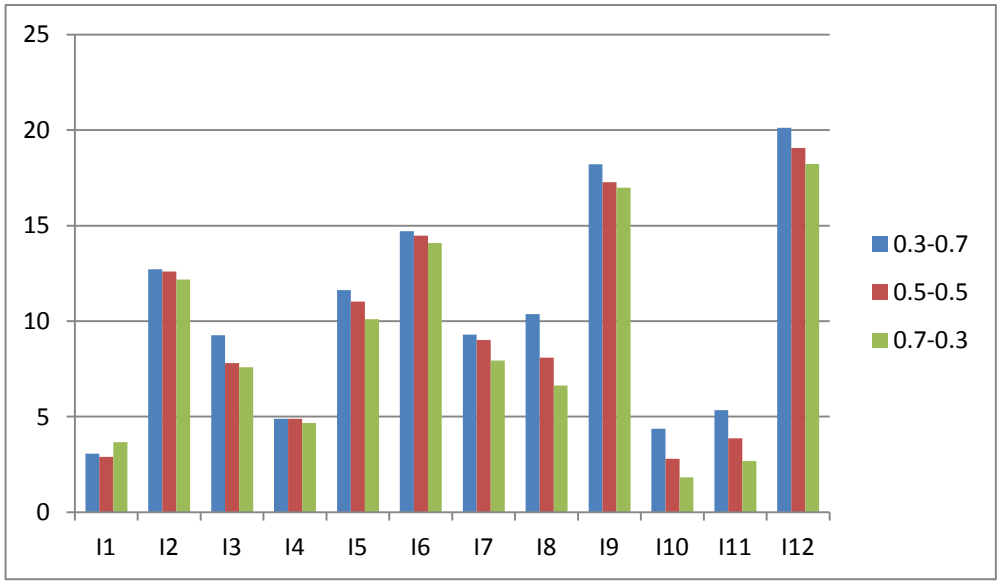


Figure 4.9 : « Gap2 » le pourcentage de décalage de l'algorithme par rapport à la borne inférieure, pour les petites instances.

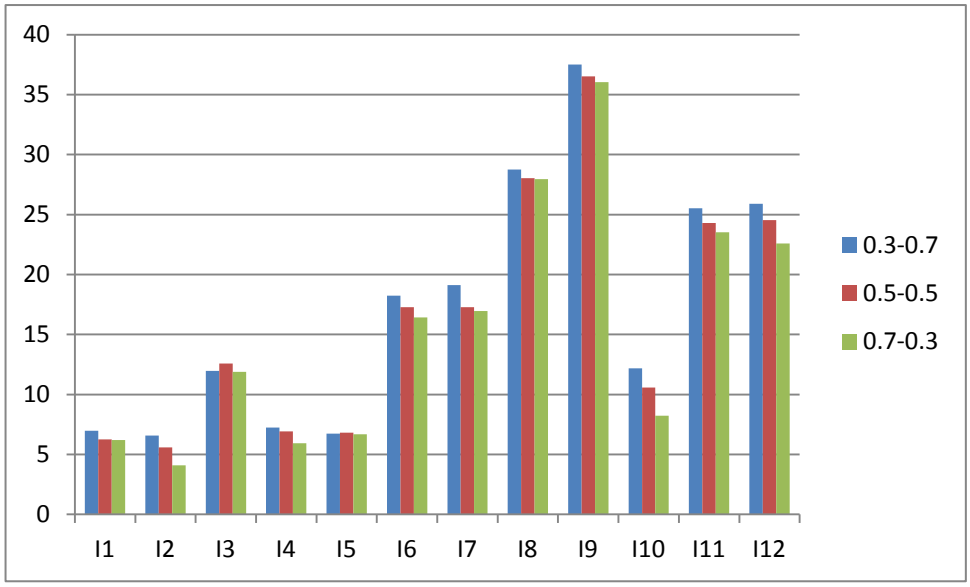


Figure 4.10 : « GAP2 » Pourcentage de décalage de l'algorithme par rapport à la borne inférieure, pour les moyennes instances.

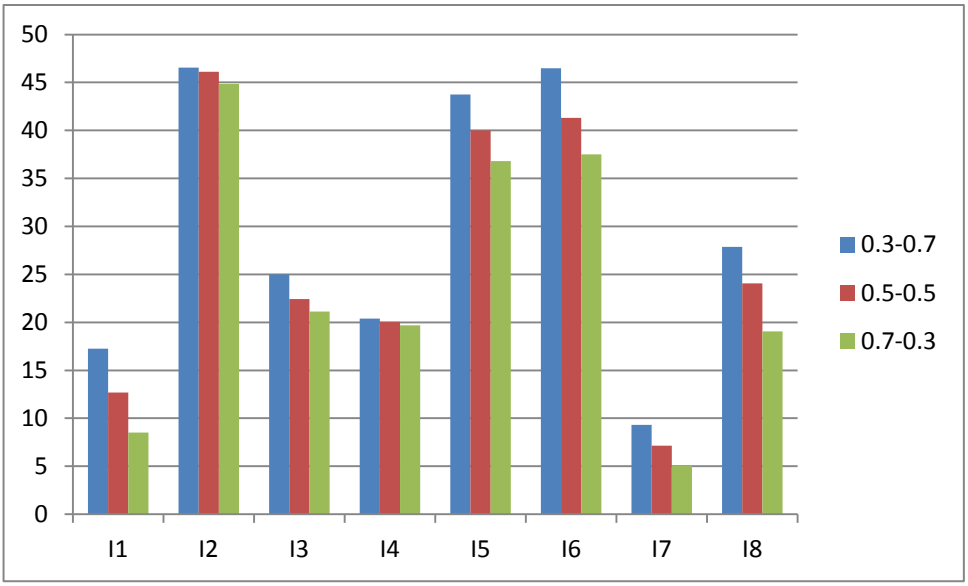


Figure 4.11 : « GAP2 » Pourcentage de décalage de l'algorithme par rapport à la borne inférieure, pour les grandes instances.

Pour conclure l'interprétation de ces résultats, nous pouvons dire que l'algorithme de décomposition résout le problème de façon efficace ; en termes de temps de réponse et en termes de qualité des solutions fournies.

4.7 Synthèse

Les résultats obtenus montrent que l'approche proposée résout efficacement le problème. Donc, nous déduisons que ces résultats peuvent être exploitables dans le milieu industriel contenu des améliorations apportées sur le temps de production total et sur l'exploitation des différentes ressources. La démarche utilisée a été introduite structurellement à commencer par le modèle mathématique qui a fourni des solutions pour les petites instances, et qui a servi comme repère pour analyser la qualité des solutions approximatives. Ensuite, l'approche de décomposition qui a permis la décomposition du problème en plusieurs sous-problèmes moins complexes, le sous-problème d'affectation des tâches au ressources humaines qualifiées avec la minimisation du makespan des ressources, le sous-problème de séquençage des tâches pour comme objectif l'exploitation optimale des outils, et le sous-problème d'affectation de tâches aux machines pour comme objectif la minimisation du makespan des machines et la consommation totale de l'énergie. Pour la minimisation du premier sous-problème, la métaheuristique du recuit simulé a été adaptée, et a fait preuve d'efficacité. Pour la résolution du deuxième et troisième sous-problème, un algorithme a été développé et implémenté. Lors d'une analyse de performances, les solutions construites par le recuit simulé et l'algorithme, pour les problèmes de petites tailles, ont été comparées aux solutions du modèle mathématique, et pour les problèmes de moyennes et grandes tailles, ont été comparées à une borne inférieure. Les résultats ont assuré que les solutions approximatives, fournies par l'algorithme sont d'une qualité satisfaisante.

4.8 Conclusion

Dans ce chapitre, nous avons étudié un problème d'ordonnancement multi-objectif, qui s'agit de minimiser le makespan et la consommation totale de l'énergie dans un environnement de machines parallèles identiques avec effet de détérioration, sous contraintes de deux types de ressources, à savoir ressources humaines qualifiées et ressources partagées. Par ressources humaines qualifiées, nous signifions que le temps opératoire d'une tâche dépend, discrètement, de la ressource humaine qui exécute la tâche. Quant aux ressources partagées, ce sont des outils de finitions, dont l'occupation ne débute pas au début d'exécution de la tâche, mais aux derniers instants de l'exécution. L'étude de ce problème est passée, premièrement, par la modélisation mathématique en un modèle de programmation linéaire. Mais vue la complexité élevée du problème, le développement d'une approche approximative pour une résolution efficace été une nécessité. Donc, nous avons décomposé le problème en trois sous-problèmes moins complexes, à savoir sous-problème d'affectation de tâches aux ressources humaines, sous-problème de séquençage de tâches, et sous-problème d'affectation de tâches aux machines. Par la suite, nous avons proposé un recuit simulé pour résoudre le premier sous-problème, et un algorithme pour résoudre le deuxième et troisième sous-problème.

Les performances de cet algorithme de décomposition ont été comparées aux performances du modèle mathématique, en termes de temps de réponse, et en termes de qualité de solutions fournies, mais seulement pour quelques petites instances. Pour les grandes instances les solutions de l'algorithme ont été comparées à une borne inférieure. Les résultats montrent que l'algorithme surpasse le modèle mathématique en termes de temps de réponse, et que la qualité de ses solutions est satisfaisante.

Conclusion générale

Le contexte de notre travail concerne l'ordonnancement des machines parallèles avec effet de détérioration, et sous contraintes de plusieurs types de ressources. Nous avons élaboré deux différentes investigations, sur deux différents problèmes : la première, détaillée dans la chapitre III, est une investigation sur le problème d'ordonnancement d'un atelier de machines parallèles identiques avec détérioration et deux types de ressources consommables, dont l'une est une ressource flexible. L'objectif étant multicritères, il minimise le makespan et les coûts des deux ressources. La deuxième investigation est présentée dans le chapitre IV. Il s'agit d'une investigation sur un problème d'ordonnancement de machines parallèles avec effet de détérioration et contraintes de deux ressources renouvelables, dont l'une est une ressource humaine qualifiée, l'autre est une ressource partagée "outils". Toujours dans le contexte de l'optimisation multicritères, l'objectif de cette investigation est de minimiser le makespan et la consommation totale en énergie.

Cependant, avant l'élaboration de ces investigations, nous avons détaillé dans un premier chapitre les notions de bases de l'ordonnancement, et dans un deuxième chapitre les différentes méthodes de résolution destinées aux problèmes d'ordonnancement.

Dans le premier chapitre, nous avons défini les modèles de l'ordonnancement classique, ceux qui ont servi de bases pour le développement théorique des modèles de l'ordonnancement actuel. Par la suite nous avons défini des modèles de l'ordonnancement actuel où l'effet de détérioration et les contraintes de ressources sont considérés, c es modèles autour desquels tourne notre étude. À la fin de ce chapitre, nous avons synthétisé un état de l'art sur les études des modèles de l'ordonnancement actuel. à Travers cet état de l'art, nous avons retrouvé des manques dans la littérature, autrement dit, des domaines non explorés. Ces manques ont fait le contexte de notre problématique qui s'agit « des problèmes d'ordonnancement avec effet de détérioration, et plusieurs types de ressources (renouvelables et consommables). »

La théorie de l'ordonnancement, étant un outil de prise de décision de court terme, a pour objectif de modéliser le système de production en question en un modèle d'ordonnancement, et ensuite formuler les objectifs de l'entreprise sous formes de critères à optimiser. Par conséquent, un problème d'ordonnancement devient un problème d'optimisation, dont la résolution nécessite des approches

performantes, autrement dit, des approches avec un temps de réponse abordable, fournissant des solutions à un niveau de qualité satisfaisant. Cependant, dans le deuxième chapitre, nous avons classé les méthodes de résolutions, et présenté l'avantage et l'inconvénient de chacune. Ensuite, nous avons étudié la littérature, où les différentes méthodes de résolution sont utilisées pour les problèmes d'ordonnancement abordés dans le premier chapitre, à savoir problème d'ordonnancement avec effet de détérioration, sous contraintes de ressources consommables ou renouvelables. Cela nous a permis de sélectionner les approches que nous avons jugé les plus efficaces pour résoudre les problèmes étudiés dans le troisième et quatrième chapitre, à savoir, la programmation linéaire, la méthode de pondération pour les objectifs multicritères, la métaheuristique du recuit simulé multi-objectif et les approches de décomposition.

Dans le troisième chapitre, où un problème de minimisation du makespan et des coûts de ressources dans un environnement de machines parallèles identiques avec effet de détérioration et consommation de deux types ressources est traité, nous nous sommes intéressés à la modélisation mathématique du problème par un modèle de programmation linéaire. Ensuite, nous avons réalisé une analyse combinatoire de ce modèle. Cette dernière nous a permis le développement d'un lemme. Mais vue la complexité élevée du problème, le modèle mathématique, étant une approche de résolution exacte, n'est plus en mesure de résoudre efficacement le problème. Pour cette raison, nous avons utilisé la métaheuristique du recuit simulé multi-objectif « MOSA ». Pour une meilleure adaptation de cette métaheuristique, une analyse de sensibilité été faite. Cette analyse nous a permis de trouver la meilleure combinaison des différents paramètres de « MOSA ». Par la suite, nous avons développé un algorithme plus performant, basé sur une approche de décomposition, appelé le « 2-steps algorithm », où une version améliorée du « MOSA » est utilisée avec le lemme proposé lors de l'analyse combinatoire. Cet algorithme vise en premier temps à décomposer le problème en deux sous-problèmes moins difficiles à résoudre séparément. Le premier sous problème est un problème d'affectation de tâches tout en minimisant le makespan et le coût du deuxième type de ressources, le deuxième sous problème est un problème d'allocation du premier type de ressources. Le « 2-steps algorithme » résout le premier sous-problème en utilisant une version améliorée de « MOSA », où la solution initiale est définie par l'heuristique LPT/SPT, ensuite il résout le deuxième sous-problème en utilisant le lemme développé à partir de l'analyse combinatoire. À la fin de ce chapitre, nous avons développé une borne inférieure, et réalisé une analyse de performances qui compare entre les trois approches utilisées, à savoir le modèle mathématique, le « MOSA » et le « 2-steps algorithm ». Les résultats étaient comme le suivant : pour les problèmes de petites tailles le modèle mathématique donne des solutions exactes, mais le « 2-steps algorithm » donne les mêmes solutions en un temps de calcul réduit. Pour les problèmes de moyennes et grandes tailles, le « 2-steps-algorithm » surpasse les deux autres méthodes, et donne des solutions jugées satisfaisantes, pour leur décalage acceptable de la borne inférieure.

L'étude réalisée dans ce chapitre a fait l'objet d'une publication dans un journal de classe A.

Dans le quatrième chapitre, où un problème de minimisation du makespan et de la consommation totale de l'énergie est traité dans un environnement de machines parallèles identiques avec effet de détérioration, et sous contraintes de disponibilité des ressources humaines qualifiées et d'autre partagées "outils", nous avons proposé un modèle mathématique de programmation linéaire originale qui peut résoudre certaines tailles du problème mieux que les autres modèles de la littérature. En dépit de cela, la complexité élevée du problème fait que le modèle mathématique soit limité aux petites tailles du problème, c'est pourquoi nous avons développé un algorithme en utilisant une approche de décomposition. Cette approche nous a permis de décomposé le problème en trois sous-problèmes moins complexes à résoudre séparément. Le premier sous-problème est un problème d'affectation de tâche aux ressources humaines qualifiées, tel que le makespan des ressources est minimisé. Ce sous-problème est résolu par le recuit simulé. Le deuxième sous-problème est problème séquençage de tâche, et le troisième sous-problème est un problème d'affectation de tâches aux machines. Ces deux sous-problèmes sont résolus par un algorithme que nous avons proposé, tel que le makespan et la consommation totale en énergie sont minimisés. Une analyse de performances des deux méthodes, à savoir le MILP et l'algorithme de décomposition, est réalisée dans la fin de ce chapitre. Les résultats de cette analyse confirment que l'algorithme de décomposition surpasse le MILP en termes de temps de réponse, et de capacité de résoudre les moyennes et grandes tailles du problème. Une partie de cette étude a fait l'objet d'une communication internationale.

Au finale, il est important de noter que la contribution de notre étude prend trois volets:

Le premier volet s'appuie sur la formulation de deux nouveaux modèles d'ordonnancement, qui n'ont pas été formulés ni étudiés avant, à savoir, un modèle d'ordonnancement de machines parallèles avec considération de plusieurs types de ressources consommables et effet de détérioration. Et un autre modèle d'ordonnancement de machines parallèles avec considération de l'effet de détérioration, de ressources humaines qualifiées et d'autres ressources partagées, dont l'allocation est dynamique. Ce dernier modèle qui a été projeté sur un cas d'étude d'un atelier de poterie artisanale.

Le deuxième volet propose un nouveau modèle mathématique de programmation linéaire qui modélise le problème d'ordonnancement à machines parallèles avec effet de détérioration, ressources humaines qualifiées et outils. Ce dernier est un modèle original, dont le principe de fonctionnement diffère des modèles qui existent dans la littérature.

L'étude, étant réalisée sur de nouveau modèle d'ordonnancement, le troisième volet consiste en le développement de nouvelles approches de résolution approximatives, mais efficaces.

Perspectives

Dans ce contexte, nous pouvons conclure que l'étude réalisée dans cette thèse représente un potentiel plus au moins considérable grâce aux nouveaux modèles qu'elle a proposé et aux approches de résolutions développées. Les modèles d'ordonnancement qui peuvent servir, non seulement, dans le processus d'ordonnancement des ateliers où plusieurs ressources sont utilisées ou consommées, mais aussi, qui peuvent servir comme base pour le développement de nouveaux modèles encore plus robuste et réaliste. Toutefois, le domaine de l'ordonnancement avec différents types de ressources reste très large, et mérite une réflexion plus approfondie quant à ses implications sur la prise de décision opérationnelle et l'exploitation optimale des capacités du système de production. Par conséquent, ce travail nécessite un enrichissement pour la création de nouvelles pistes de recherches dans les domaines suivants:

Développement d'une méthode de résolution exacte : La résolution des problèmes considérés dans cette étude est basée en sa grande majorité sur des approches approximatives, surtout la résolution du problème considéré dans le quatrième chapitre. Toutefois, et en dépit des résultats promettant de l'analyse de performances réalisée, le décalage des solutions approximatives par rapport aux solutions optimales, pour les grandes instances, n'est pas confirmé. Pour cette raison, le développement d'une approche de résolution exacte, tel que la méthode de séparation et évaluation appelée "B.A.B", rapportera une contribution d'un potentiel important dans le domaine de l'ordonnancement avec effet de détérioration et sous contraintes de différentes ressources.

Exploitation du modèle mathématique pour développer une matheuristique: lors des deux études réalisées dans le premier et deuxième chapitre, nous avons développé des modèles mathématiques de programmation linéaire. Ces modèles n'ont été exploités que pour la résolution des petites, et quelques moyennes instances. Or, qu'en utilisant des techniques de relaxation, ces modèles mathématiques peuvent être exploités dans la construction d'une partie de la solution pour de plus grandes tailles du problème. Donc, nous proposons le développement d'une matheuristique, qui est une méthode de résolution qui combine entre le modèle mathématique et un algorithme approximatif.

Ainsi, l'étude du quatrième chapitre, réalisée sur l'usine de poterie artisanale peut être améliorée et exploitée pour une application réelle, ce qui permettra à l'usine de : premièrement mieux gérer et satisfaire les artisans selon leurs compétences, ces artisans qui sont une ressource très rare et difficile à gérer, deuxièmement mieux exploiter les machines en parallèles « tours potiers », qui sont disponibles en quantité très souvent supérieure à celle des artisans, et donc cette étude fait en sorte que toutes les machines soit bien exploiter, troisièmement, bien coordonner entre l'ensemble des artisans pour qu'ils puissent tous travailler de façon harmonieuse, selon la disponibilité des outils.

Référence

Abdeljaoued, M.A., Saadani, N.E.H., Bahroun, Z., 2018. Heuristic and metaheuristic approaches for parallel machine scheduling under resource constraints. Oper. Res. https://doi.org/10.1007/s12351-018-0412-3

Agnetis, A., Murgia, G., Sbrilli, S., 2014. A job shop scheduling problem with human operators in handicraft production. Int. J. Prod. Res. 52, 3820–3831. https://doi.org/10.1080/00207543.2013.831220

Alidaee, B., Womer, N., 1999. Scheduling with time dependent processing times: Review and extensions. J. Oper. Res. Soc. 50, 711–720.

Bansal, S.P., 1980. Single machine scheduling to minimize weighted sum of completion times with secondery criterion- A branch and bound approach. Eur. J. Oper. Res. 177–181.

Belkaid, F., 2014. Investigation sur l'ordonnancement des systèmes à machines parallèles (Thèse de doctorat). Université de Tlemcen, Tlemcen, Algérie.

Belkaid, F., Sari, Z., Souier, M., 2013. A Genetic Algorithm for the Parallel Machine Scheduling Problem with Consumable Resources: Int. J. Appl. Metaheuristic Comput. 4, 17–30. https://doi.org/10.4018/jamc.2013040102

Belkaid, F., Yalaoui, F., Sari, Z., 2016a. An Efficient Approach for the Reentrant Parallel Machines Scheduling Problem under Consumable Resources Constraints: Int. J. Inf. Syst. Supply Chain Manag. 9, 1–25. https://doi.org/10.4018/IJISSCM.2016070101

Belkaid, F., Yalaoui, F., Sari, Z., 2016b. Investigations on Performance Evaluation of Scheduling Heuristics and Metaheuristics in a Parallel Machine Environment, in: Talbi, E.-G., Yalaoui, F., Amodeo, L. (Eds.), Metaheuristics for Production Systems, Operations Research/Computer Science Interfaces Series. Springer International Publishing, Cham, pp. 191–222. https://doi.org/10.1007/978-3-319-23350-5_9

Blazewics, J.J., Kubiak, W., 1987. Minimizing Mean Flow-Time with Parallel Processors and Resource Constraints. Acta Inform. 513–524.

Blazewics, J.J., Lenstra, K., Rino ov Kan, 1983. Scheduling subject to resource constraints. Discrete Appl. Math. 11–24.

Blazewicz, J., Lenstra, J.K., 1983. Scheduling subject to ressource constraints: classification and complexity 11–24.

Boufellouh, R., Belkaid, F., 2020. Bi-objective optimization algorithms for joint production and maintenance scheduling under a global resource constraint: Application to the permutation flow shop problem. Comput. Oper. Res. 122, 104943. https://doi.org/10.1016/j.cor.2020.104943

Browne, S., Yechiali, U., 1990. scheduling deteriorating jobs on a single processor. Oper. Res. 38, 495–498.

Carlier, J., Chrétienne, P., 1988. Problème d'ordonnancement, Masson. ed. Paris.

Carlier, J., Rinnooy Kan, A.H.G., 1982. Scheduling subject to nonrenewable-resource constraints. Oper. Res. Lett. 1, 52–55. https://doi.org/10.1016/0167-6377(82)90045-1

Celano, G., Costa, A., Fichera, S., 2008. Scheduling of unrelated parallel manufacturing cells with limited human resources. Int. J. Prod. Res. 46, 405–427. https://doi.org/10.1080/00207540601138452

Chen, Z.-L., 2004. Simultaneous Job Scheduling and Resource Allocation on Parallel Machines. Ann. Oper. Res. 129, 135–153. https://doi.org/10.1023/B:ANOR.0000030685.31167.11

Chen, Z.-L., 1996. Parallel machine scheduling with time dependent processing times. Discrete Appl. Math. 70, 81–93. https://doi.org/10.1016/0166-218X(96)00102-3

Cheng, T.C.E., 1990. A state-of-the-art review of parallel-machine scheduling reseach. Eur. J. Oper. Res. 271–292.

Cheng, T.C.E., Ding, Q., Lin, B.M.T., 2004. A concise survey of scheduling with time-dependent processing times. Eur. J. Oper. Res. 152, 1–13. https://doi.org/10.1016/S0377-2217(02)00909-8

Cheng, T.C.E., Kang, L.Y., Ng, C.T., 2007. Due-date assignment and parallel-machine scheduling with deteriorating jobs. J. Oper. Res. Soc. 58, 1103–1108. https://doi.org/10.1057/palgrave.jors.2602225

Chuang, S.-P., Hsu, T.-S., Yang, C.-L., 2009. Parallel work center scheduling with release dates, due dates, and resource-dependent processing times. Int. J. Adv. Manuf. Technol. 40, 193–202. https://doi.org/10.1007/s00170-007-1313-4

Chudak, F.A., 1999. Approximation algorithms for precedence-Constrained Scheduling problems on parallel machines that run at different speeds. J. Algorithms 30, 323–343.

Chung, B.D., Kim, B.S., 2016. A hybrid genetic algorithm with two-stage dispatching heuristic for a machine scheduling problem with step-deteriorating jobs and rate-modifying activities. Comput. Ind. Eng. 98, 113–124. https://doi.org/10.1016/j.cie.2016.05.028

Cochran, J.K., Horng, S.-M., Fowler, J.W., 2003. A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. Comput. Oper. Res. 30, 1087–1102. https://doi.org/10.1016/S0305-0548(02)00059-X

Collette, Y., Siarry, P., 2002. Optimisation multiobjectif. Eyrolles, Paris.

Daniels Richard L., Barbara J., H., Joseph B., M., 1996. Scheduling Parallel Manufacturing Cells with Resource Flexibility. Manag. Sci. 42, 1260–1276.

Daniels, R.L., Barbara J., H., Joseph B., M., 1997. An analysis of heuristics for the parallel-machine flexible-resource scheduling problem. Anu. Oper. Researsh 70, 439–472.

Daniels, R.L., Hua, S.Y., Webster, S., 1999. Heuristics for parallel-machine fexible-resource scheduling problems with unspeciÞed job assignment. Comput. Oper. Res. 143–155.

Dell'Amico, M., Iori, M., Martello, S., Monaci, M., 2008. Heuristic and Exact Algorithms for the Identical Parallel Machine Scheduling Problem. Inf. J. Comput. 20, 333–344. https://doi.org/10.1287/ijoc.1070.0246

Ding, J., Shen, L., Lü, Z., Peng, B., 2019. Parallel machine scheduling with completion-time-based criteria and sequence-dependent deterioration. Comput. Oper. Res. 103, 35–45. https://doi.org/10.1016/j.cor.2018.10.016

Dréo, J. (Ed.), 2006. Metaheuristics for hard optimization: methods and case studies. Springer, Berlin.

Edis, E.B., Oguz, C., 2012. Parallel machine scheduling with flexible resources. Comput. Ind. Eng. 63, 433–447. https://doi.org/10.1016/j.cie.2012.03.018

Edis, E.B., Oguz, C., Ozkarahan, I., 2013. Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. Eur. J. Oper. Res. 230, 449–463. https://doi.org/10.1016/j.ejor.2013.02.042

Esquirol, P., Lopez, P., 1999. L'ordonnancement. Économica, Paris.

Fanjul-Peyro, L., Perea, F., Ruiz, R., 2017. Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. Eur. J. Oper. Res. 260, 482–493. https://doi.org/10.1016/j.ejor.2017.01.002

Fanjul-Peyro, L., Ruiz, R., 2011. Size-reduction heuristics for the unrelated parallel machines scheduling problem. Comput. Oper. Res. 38, 301–309. https://doi.org/10.1016/j.cor.2010.05.005

França, P.M., Mendes, A., Moscato, P., 2001. A memetic algorithm for the total tardiness single machine scheduling problem. Eur. J. Oper. Res. 132, 224–242. https://doi.org/10.1016/S0377-2217(00)00140-5

Garcia, C., Rabadi, G., Handy, F., 2018. Dynamic resource allocation and coordination for high-load crisis volunteer management. J. Humanit. Logist. Supply Chain Manag. 8, 533–556. https://doi.org/10.1108/JHLSCM-02-2018-0019

Garey, M.R., Johnson, D.S., 1975. Complexity Results for Multiprocessor Scheduling under Resource Constraints. SIAM J. Comput. 4, 397–411. https://doi.org/10.1137/0204035

Garey, M.R., Johnson, M.R., 1979. Computers and intractability: Aguide to the theory of NP-completeness, San Francisco: Freeman. ed.

Gawiejnowicz, S., 2007. Scheduling deteriorating jobs subject to job or machine availability constraints. Eur. J. Oper. Res. 180, 472–478. https://doi.org/10.1016/j.ejor.2006.04.021

Gawiejnowicz, S., 1996. A note on scheduling on a single processir with speed dependent on a number of executed jobs. Inf. Process. Lett. 57, 297–300.

Gharehgozli, A.H., Tavakkoli-Moghaddam, R., Zaerpour, N., 2009. A fuzzy-mixed-integer goal programming model for a parallel-machine scheduling problem with sequence-dependent setup times and release dates. Robot. Comput.-Integr. Manuf. 25, 853–859. https://doi.org/10.1016/j.rcim.2008.12.005

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. 533–549.

Gökgür, B., Hnich, B., Özpeynirci, S., 2018. Parallel machine scheduling with tool loading: a constraint programming approach. Int. J. Prod. Res. 56, 5541–5557. https://doi.org/10.1080/00207543.2017.1421781

Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R., 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey, in: Annals of Discrete Mathematics. Elsevier, pp. 287–326. https://doi.org/10.1016/S0167-5060(08)70356-X

Gunther, Z., Roland, B., Michael, B., n.d. Meta-heuristic Search Concepts, Springer Heidelberg Dordrecht. ed. London.

Györgyi, P., Kis, T., 2017. Approximation schemes for parallel machine scheduling with non-renewable resources. Eur. J. Oper. Res. 258, 113–123. https://doi.org/10.1016/j.ejor.2016.09.007

Györgyi, P., Kis, T., 2015. Reductions between scheduling problems with non-renewable resources and knapsack problems. Theor. Comput. Sci. 565, 63–76. https://doi.org/10.1016/j.tcs.2014.11.007

Hsieh, P.-H., Yang, S.-J., Yang, D.-L., 2015. Decision support for unrelated parallel machine scheduling with discrete controllable processing times. Appl. Soft Comput. 30, 475–483. https://doi.org/10.1016/j.asoc.2015.01.028

Hsu, C.-J., Yang, D.-L., 2014. Unrelated parallel-machine scheduling with position-dependent deteriorating jobs and resource-dependent processing time. Optim. Lett. 8, 519–531. https://doi.org/10.1007/s11590-012-0594-1

Huang, S., Cai, L., Zhang, X., 2010. Parallel dedicated machine scheduling problem with sequence-dependent setups and a single server. Comput. Ind. Eng. 58, 165–174. https://doi.org/10.1016/j.cie.2009.10.003

Huang, X., Wang, M.-Z., 2011. Parallel identical machines scheduling with deteriorating jobs and total absolute differences penalties. Appl. Math. Model. 35, 1349–1353. https://doi.org/10.1016/j.apm.2010.09.013

Jackson, J.R., 1955. Scheduling a production line to minimize maximum tardiness, Technical Report 43. Management research project. ed. University of California, Los Angelos.

Jeng, A.A.K., Lin, B.M.T., 2005. Minimizing the total completion time in single-machine scheduling with step-deteriorating jobs. Comput. Oper. Res. 32, 521–536. https://doi.org/10.1016/j.cor.2003.08.001

Ji, M., Cheng, T.C.E., 2009. Parallel-machine scheduling of simple linear deteriorating jobs. Theor. Comput. Sci. 410, 3761–3768. https://doi.org/10.1016/j.tcs.2009.04.018

Ji, M., Cheng, T.C.E., 2007. An FPTAS for scheduling jobs with piecewise linear decreasing processing times to minimize makespan. Inf. Process. Lett. 102, 41–47. https://doi.org/10.1016/j.ipl.2006.11.014

Ji, M., Ge, J., Chen, K., Cheng, T.C.E., 2013. Single-machine due-window assignment and scheduling with resource allocation, aging effect, and a deteriorating rate-modifying activity. Comput. Ind. Eng. 66, 952–961. https://doi.org/10.1016/j.cie.2013.08.020

Johnson, S.M., 1954. Optimal two- and three-stage production schedules with setup times included. Nav. Res. Logist. Q. 1, 61–68. https://doi.org/10.1002/nav.3800010110

Kaabi-harrath, J., 2004. Contribution à l'ordonnancement des activités de maintenances dans les systèmes de production (Thèse de doctorat). Université de FRANCHE-COMPTE, France.

Kan, A.H.G.R., 1976. Machine Scheduling Problems. Springer US, Boston, MA. https://doi.org/10.1007/978-1-4613-4383-7

Kang, L., Ng, C.T., 2007. A note on a fully polynomial-time approximation scheme for parallel-machine scheduling with deteriorating jobs. Int. J. Prod. Econ. 109, 180–184. https://doi.org/10.1016/j.ijpe.2006.11.014

Kellerer, H., Strusevich, V.A., 2003. Scheduling problems for parallel dedicated machines under multiple resource constraints. Discrete Appl. Math. 133, 45–68. https://doi.org/10.1016/S0166-218X(03)00433-5

Kim, D.-W., Kim, K.-H., Jang, W., Chen, F.F., 2002. Unrelated parallel machine scheduling with setup times using simulated annealing. Robot. Comput. Integr. Manuf. 223–231.

Kim, M.-Y., Lee, Y.H., 2012. MIP models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server. Comput. Oper. Res. 39, 2457–2468. https://doi.org/10.1016/j.cor.2011.12.011

Laguna, M., Barnes, J.W., Glover, F.W., 1991. Tabu search methods for a single machine scheduling problem. J. Intell. Manuf. 2, 63–73. https://doi.org/10.1007/BF01471219

Lakshminarayan, S., Lakshmanan, R., Papineau, R.L., Rochette, R., 1978. Technical Note—Optimal Single-Machine Scheduling with Earliness and Tardiness Penalties. Oper. Res. 26, 1079–1082. https://doi.org/10.1287/opre.26.6.1079

Lawler, E.L., Sivazlian, B.D., 1978. Minimization of Time-Varying Costs in Single-Machine Scheduling. Oper. Res. 26, 563–569. https://doi.org/10.1287/opre.26.4.563

Lee, W.-C., Wu, C.-C., Chen, P., 2006. A simulated annealing approach to makespan minimization on identical parallel machines. Int. J. Adv. Manuf. Technol. 31, 328–334. https://doi.org/10.1007/s00170-005-0188-5

Lenstra, J.K., 1977. Complexity of machine scheduling problems, in: Annals of Discrete Mathematics 1. pp. 343–362.

Li, L., Wang, J.-J., 2016. Scheduling jobs with deterioration effect and controllable processing time. Neural Comput. Appl. 29, 1163–1170. https://doi.org/10.1007/s00521-016-2630-z

Li, X., Yalaoui, F., Amodeo, L., Chehade, H., 2012. Metaheuristics and exact methods to solve a multiobjective parallel machines scheduling problem. J. Intell. Manuf. 23, 1179–1194. https://doi.org/10.1007/s10845-010-0428-x

Liaw, C.-F., Lin, Y.-K., Cheng, C.-Y., Chen, M., 2003. Scheduling unrelated parallel machines to minimize total weighted tardiness. Comput. Oper. Res. 1777–1789.

Lin, S.-W., Ying, K.-C., 2015. A multi-point simulated annealing heuristic for solving multiple objective unrelated parallel machine scheduling problems. Int. J. Prod. Res. 53, 1065–1076. https://doi.org/10.1080/00207543.2014.942011

Liu, C.-H., 2014. Approximate trade-off between minimisation of total weighted tardiness and minimisation of carbon dioxide (CO_2) emissions in bi-criteria batch scheduling problem. Int. J. Comput. Integr. Manuf. 27, 759–771. https://doi.org/10.1080/0951192X.2013.834479

Liu, M., Zheng, F., Wang, S., Xu, Y., 2013. Approximation algorithms for parallel machine scheduling with linear deterioration. Theor. Comput. Sci. 497, 108–111. https://doi.org/10.1016/j.tcs.2012.01.020

Loukil, T., Teghem, J., Tuyttens, D., 2005. Solving multi-objective production scheduling problems using metaheuristics. Eur. J. Oper. Res. 161, 42–61. https://doi.org/10.1016/j.ejor.2003.08.029

Low, C., 2005. Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. Comput. Oper. Res. 32, 2013–2025. https://doi.org/10.1016/j.cor.2004.01.003

Low, C., Wu, G.-H., 2016. Unrelated parallel-machine scheduling with controllable processing times and eligibility constraints to minimize the makespan. J. Ind. Prod. Eng. 33, 286–293. https://doi.org/10.1080/21681015.2016.1139005

Lu, C., Gao, L., Li, X., Pan, Q., Wang, Q., 2017. Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm. J. Clean. Prod. 144, 228–238. https://doi.org/10.1016/j.jclepro.2017.01.011

Mazdeh, M.M., Zaerpour, F., Zareei, A., Hajinezhad, A., 2010. Parallel machines scheduling to minimize job tardiness and machine deteriorating cost with deteriorating jobs. Appl. Math. Model. 34, 1498–1510. https://doi.org/10.1016/j.apm.2009.08.023

Meziani, N., Oulamara, A., Boudhar, M., 2019. Two-machine flowshop scheduling problem with coupled-operations. Ann. Oper. Res. 275, 511–530. https://doi.org/10.1007/s10479-018-2967-z

Min, L., Cheng, W., 1999. A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines. Artif. Intell. Eng. 13, 399–403. https://doi.org/10.1016/S0954-1810(99)00021-7

Mokhtari, H., Abadi, I.N.K., Cheraghalikhani, A., 2011. A multi-objective flow shop scheduling with resource-dependent processing times: trade-off between makespan and cost of resources. Int. J. Prod. Res. 49, 5851–5875. https://doi.org/10.1080/00207543.2010.523724

Mokotoff, E., 2004. An exact algorithm for the identical parallel machine scheduling problem. Eur. J. Oper. Res. 152, 758–769. https://doi.org/10.1016/S0377-2217(02)00726-9

Mommarché, N., 2000. Algorithmes de fourmis artificielles: applications à la classification et à l'optimisation. Université François Rabelais, France.

Mosheiov, G., 2005. A note on scheduling deteriorating jobs. Math. Comput. Model. 41, 883–886. https://doi.org/10.1016/j.mcm.2004.09.004

Mosheiov, G., 1998. Multi-Machine Scheduling with Linear Deterioration. Inf. Syst. Oper. Res. 205–214.

Niu, G., Sun, S., Lafon, P., Zhang, Y., Wang, J., 2012. Two decompositions for the bicriteria job-shop scheduling problem with discretely controllable processing times. Int. J. Prod. Res. 50, 7415–7427. https://doi.org/10.1080/00207543.2011.651169

Ouazene, Y., Yalaoui, F., 2017. Identical parallel machine scheduling with time-dependent processing times. Theor. Comput. Sci. https://doi.org/10.1016/j.tcs.2017.12.001

Ovacik, I.M., Uzsoy, R., 1997a. Decomposition Methods for Complex Factory Scheduling Problems. Springer US, Boston, MA. https://doi.org/10.1007/978-1-4615-6329-7

Ovacik, I.M., Uzsoy, R., 1997b. Decomposition Methods for Complex Factory Scheduling Problems. Springer US, Boston, MA. https://doi.org/10.1007/978-1-4615-6329-7

Özpeynirci, S., Gökgür, B., Hnich, B., 2016. Parallel machine scheduling with tool loading. Appl. Math. Model. 40, 5660–5671. https://doi.org/10.1016/j.apm.2016.01.006

Pan, Z., Lei, D., Zhang, Q., 2018. A New Imperialist Competitive Algorithm for Multiobjective Low Carbon Parallel Machines Scheduling. Math. Probl. Eng. 2018, 1–13. https://doi.org/10.1155/2018/5914360

Pei, J., Liu, X., Liao, B., Pardalos, P.M., Kong, M., 2018. Single-machine scheduling with learning effect and resource-dependent processing times in the serial-batching production. Appl. Math. Model. 58, 245–253. https://doi.org/10.1016/j.apm.2017.07.028

Pinedo, M.L., 2008. Scheduling. Springer New York, New York, NY. https://doi.org/10.1007/978-0-387-78935-4

Plateau, M.-C., Rios-Solis, Y.A., 2010. Optimal solutions for unrelated parallel machines scheduling problems using convex quadratic reformulations. Eur. J. Oper. Res. 201, 729–736. https://doi.org/10.1016/j.ejor.2009.03.049

Potts, C.N., Whitehead, J.D., 2007. Heuristics for a coupled-operation scheduling problem. J. Oper. Res. Soc. 58, 1375–1388. https://doi.org/10.1057/palgrave.jors.2602272

Rashidi, E., Jahandar, M., Zandieh, M., 2010. An improved hybrid multi-objective parallel genetic algorithm for hybrid flow shop scheduling with unrelated parallel machines. Int. J. Adv. Manuf. Technol. 49, 1129–1139. https://doi.org/10.1007/s00170-009-2475-z

Richard L., D., Stella Y., H., Scott, W., 1999. Heuristics for parallel-machine ßexible-resource scheduling problems with unspeciÞed job assignment Richard L. Daniels. Comput. Oper. Res. 143–155.

Rocha, P.L., Ravetti, M.G., Mateus, G.R., Pardalos, P.M., 2008. Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. Comput. Oper. Res. 35, 1250–1264. https://doi.org/10.1016/j.cor.2006.07.015

Ruiz-Torres, A.J., López, F.J., Ho, J.C., 2007. Scheduling uniform parallel machines subject to a secondary resource to minimize the number of tardy jobs. Eur. J. Oper. Res. 179, 302–315. https://doi.org/10.1016/j.ejor.2006.03.028

Ruiz-Torres, A.J., Paletta, G., Pérez, E., 2013. Parallel machine scheduling to minimize the makespan with sequence dependent deteriorating effects. Comput. Oper. Res. 40, 2051–2061. https://doi.org/10.1016/j.cor.2013.02.018

Rustogi, K., Strusevich, V.A., 2017. Single machine scheduling with a generalized job-dependent cumulative effect. J. Sched. 20, 583–592. https://doi.org/10.1007/s10951-016-0497-6

Rustogi, K., Strusevich, V.A., 2012. Single machine scheduling with general positional deterioration and rate-modifying maintenance. Omega 40, 791–804. https://doi.org/10.1016/j.omega.2011.12.007

S. Kirkpatrick, C. D. Gelatt Jr, M. P. Vecchi, 1983. Optimization by Simulated Annealing. science 671–680.

Sekkal, N., Belkaid, F., 2020. A multi-objective simulated annealing to solve an identical parallel machine scheduling problem with deterioration effect and resources consumption constraints. J. Comb. Optim. 40, 660–696. https://doi.org/10.1007/s10878-020-00607-y

Shabtay, D., Kaspi, M., 2006. Parallel machine scheduling with a convex resource consumption function. Eur. J. Oper. Res. 173, 92–107. https://doi.org/10.1016/j.ejor.2004.12.008

Shahvari, O., Logendran, R., 2017. An Enhanced tabu search algorithm to minimize a bi-criteria objective in batching and scheduling problems on unrelated-parallel machines with desired lower bounds on batch sizes. Comput. Oper. Res. 77, 154–176. https://doi.org/10.1016/j.cor.2016.07.021

Sidney, J.B., 1977. Optimal Single-Machine Scheduling with Earliness and Tardiness Penalties. Oper. Res. 25, 62–69. https://doi.org/10.1287/opre.25.1.62

Slowinski, R., 1982. Preemptive scheduling of independent jobs on parallel machines subject to financial constraints. Eur. J. Oper. Res. 366–373.

Smith, W.E., 1956. Various optimizers of single stage production, Naval Research logistics quarterly. ed.

Soleimani, H., Ghaderi, H., Tsai, P.-W., Zarbakhshnia, N., Maleki, M., 2020. Scheduling of unrelated parallel machines considering sequence-related setup time, start time-dependent deterioration, position-dependent learning and power consumption minimization. J. Clean. Prod. 249, 119428. https://doi.org/10.1016/j.jclepro.2019.119428

Souier, M., Dahane, M., Maliki, F., 2019. An NSGA-II-based multiobjective approach for real-time routing selection in a flexible manufacturing system under uncertainty and reliability constraints. Int. J. Adv. Manuf. Technol. 100, 2813–2829. https://doi.org/10.1007/s00170-018-2897-6

Su, L.-H., Lien, C.-Y., 2009. Scheduling parallel machines with resource-dependent processing times. Int. J. Prod. Econ. 117, 256–266.

Sule, D.R., Sule, D.R., 2008. Production planning and industrial scheduling: examples, case studies, and applications, 2nd ed. ed. CRC Press, Boca Raton.

Talbi, E.-G., 2009. Metaheuristics: from design to implementation. John Wiley & Sons, Hoboken, N.J.

Tigane, M., Dahane, M., Boudhar, M., 2018. Multiobjective approach for deteriorating jobs scheduling for a sustainable manufacturing system. Int. J. Adv. Manuf. Technol. https://doi.org/10.1007/s00170-018-3043-1

Toker, A., Kondakci, S., Erkip, N., 1991. Scheduling under a non-renewable resource constraint. Oper. Res. Soc. 42, 811–814.

Toksarı, M.D., Güner, E., 2009. Parallel machine earliness/tardiness scheduling problem under the effects of position based learning and linear/nonlinear deterioration. Comput. Oper. Res. 36, 2394–2417. https://doi.org/10.1016/j.cor.2008.09.012

Torabi, S.A., Sahebjamnia, N., Mansouri, S.A., Bajestani, M.A., 2013. A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem. Appl. Soft Comput. 13, 4750–4762. https://doi.org/10.1016/j.asoc.2013.07.029

Ulungu, E.L., Teghem, J., Fortemps, P.H., Tuyttens, D., 1999. MOSA method: a tool for solving multiobjective combinatorial optimization problems. J. Multi-Criteria Decis. Anal. 8, 221–236. https://doi.org/10.1002/(SICI)1099-1360(199907)8:4<221::AID-MCDA247>3.0.CO;2-O

Valente, J.M.S., Schaller, J.E., 2010. Improved heuristics for the single machine scheduling problem with linear early and quadratic tardy penalties. Eur. J Ind. Eng. 4, 99. https://doi.org/10.1504/EJIE.2010.029572

Varadharajan, T.K., Rajendran, C., 2005. A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. Eur. J. Oper. Res. 167, 772–795. https://doi.org/10.1016/j.ejor.2004.07.020

Ventura, J.A., Kim, D., 2003. Parallel machine scheduling with earliness–tardiness penalties and additional resource constraints. Comput. Oper. Res. 30, 1945–1958. https://doi.org/10.1016/S0305-0548(02)00118-1

Vickson, R.G., 1980. Two Single Machine Sequencing Problems Involving Controllable Job Processing Times. E Trans. 12, 258–262. https://doi.org/10.1080/05695558008974515

Villa, F., Vallada, E., Fanjul-Peyro, L., 2018. Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource. Expert Syst. Appl. 93, 28–38. https://doi.org/10.1016/j.eswa.2017.09.054

Wang, H., Alidaee, B., 2019. Effective heuristic for large-scale unrelated parallel machines scheduling problems. Omega 83, 261–274. https://doi.org/10.1016/j.omega.2018.07.005

Wang, J., Guo, A.-X., Shan, F., Jiang, B., Wang, L.-Y., 2007. Single machine group scheduling under decreasing linear deterioration. J. Appl. Math. Comput. 24, 283–293. https://doi.org/10.1007/BF02832317

Wang, J.-B., 2007. Single-machine scheduling problems with the effects of learning and deterioration. Omega 35, 397–402. https://doi.org/10.1016/j.omega.2005.07.008

Wang, J.-B., Lin, L., Shan, F., 2008. Single-machine group scheduling problems with deteriorating jobs. Int. J. Adv. Manuf. Technol. 39, 808–812. https://doi.org/10.1007/s00170-007-1255-x

Wang, J.-B., Wang, M.-Z., Ji, P., 2012. Scheduling jobs with processing times dependent on position, starting time, and allotted resource. Asia-Pac. J. Oper. Res. 29, 1250030. https://doi.org/10.1142/S0217595912500303

Wang, S., Wang, X., Yu, J., Ma, S., Liu, M., 2018a. Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. J. Clean. Prod. 193, 424–440. https://doi.org/10.1016/j.jclepro.2018.05.056

Wang, S., Wang, X., Yu, J., Ma, S., Liu, M., 2018b. Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. J. Clean. Prod. 193, 424–440. https://doi.org/10.1016/j.jclepro.2018.05.056

Wang, X.-R., Wang, J.-J., 2013. Single-machine scheduling with convex resource dependent processing times and deteriorating jobs. Appl. Math. Model. 37, 2388–2393. https://doi.org/10.1016/j.apm.2012.05.025

Wang, X.-Y., Wang, J.-J., 2013. Single-machine due date assignment problem with deteriorating jobs and resource-dependent processing times. Int. J. Adv. Manuf. Technol. 67, 255–260. https://doi.org/10.1007/s00170-013-4771-x

Wei, C.-M., Wang, J.-B., Ji, P., 2012. Single-machine scheduling with time-and-resource-dependent processing times. Appl. Math. Model. 36, 792–798. https://doi.org/10.1016/j.apm.2011.07.005

Widmer, M., 2001. Les metaheuristiques: des outils performants pour les problemes industriels. Presented at the 3eme conférence francophone de MOdélisation et SIMulation "Conception, Analyse et Gestion des Systèmes Industriels, Troyes (France).

Wu, C.-C., Lee, W.-C., 2003a. Scheduling linear deteriorating jobs to minimize makespan with an availability constraint on a single machine. Inf. Process. Lett. 87, 89–93. https://doi.org/10.1016/S0020-0190(03)00262-X

Wu, C.-C., Lee, W.-C., 2003b. Scheduling linear deteriorating jobs to minimize makespan with an availability constraint on a single machine. Inf. Process. Lett. 87, 89–93. https://doi.org/10.1016/S0020-0190(03)00262-X

Xie, J., 1996. Polynomial algorithms for single machine scheduling problems with financial constraints. Oper. Reseaech Lett. 39–42.

Yang, D.-L., Cheng, T.C.E., Yang, S.-J., 2014. Parallel-machine scheduling with controllable processing times and rate-modifying activities to minimise total cost involving total completion time and job compressions. Int. J. Prod. Res. 52, 1133–1141. https://doi.org/10.1080/00207543.2013.841330

Yang, S.-J., Yang, D.-L., 2013. Note on "A unique integer mathematical model for scheduling deteriorating jobs with rate-modifying activities on a single machine." Int. J. Adv. Manuf. Technol. 64, 1759–1764. https://doi.org/10.1007/s00170-012-4139-7

Yang, S.-J., Yang, D.-L., 2010. Minimizing the makespan on single-machine scheduling with aging effect and variable maintenance activities. Omega 38, 528–533. https://doi.org/10.1016/j.omega.2010.01.003

Yin, N., Kang, L., Sun, T.-C., Yue, C., Wang, X.-R., 2014. Unrelated parallel machines scheduling with deteriorating jobs and resource dependent processing times. Appl. Math. Model. 38, 4747–4755. https://doi.org/10.1016/j.apm.2014.03.022

Yin, Y., Wu, W.-H., Cheng, T.C.E., Wu, C.-C., 2015. Single-machine scheduling with time-dependent and position-dependent deteriorating jobs. Int. J. Comput. Integr. Manuf. 28, 781–790. https://doi.org/10.1080/0951192X.2014.900872

Zäpfel, G., Braune, R., Bögl, M., 2010. Metaheuristic Search Concepts. Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-11343-7

Zhang, L., Deng, Q., Gong, G., Han, W., 2019. A new unrelated parallel machine scheduling problem with tool changes to minimise the total energy consumption. Int. J. Prod. Res. 1–20. https://doi.org/10.1080/00207543.2019.1685708

Zhang, X., Liao, L., Zhang, W., Cheng, T.C.E., Tan, Y., Ji, M., 2018. Single-machine group scheduling with new models of position-dependent processing times. Comput. Ind. Eng. 117, 1–5. https://doi.org/10.1016/j.cie.2018.01.002

Zhao, C., Tang, H., 2010. Single machine scheduling with general job-dependent aging effect and maintenance activities to minimize makespan. Appl. Math. Model. 34, 837–841. https://doi.org/10.1016/j.apm.2009.07.002

Résumé

Dans la majorité des systèmes manufacturiers, plusieurs types de ressources interviennent lors de l'exécution d'une tâche. Cependant cette dernière affirmation est très peu considérée dans les travaux existants dans la littérature. Pour combler cette lacune, nous investiguons dans cette thèse les problèmes d'ordonnancement à machines parallèles identique avec effet de détérioration et plusieurs types de ressources. Le premier axe de notre étude repose sur la considération de plusieurs ressources consommables. Quant au deuxième axe, plusieurs ressources renouvelables sont prises en compte, à savoir les ressources humaines qualifiées et ressources partagées. Deux modèles mathématiques sont proposés pour la résolution de ces deux problèmes. Ces modèles donnent des solutions exactes, mais vue la complexité des problèmes d'ordonnancement avec contraintes de ressources, l'efficacité des modèles mathématiques est limitée aux petites et moyennes instances. Pour cette raison, des méthodes de résolution approximatives sont développées. Ces dernières sont basées sur le recuit simulé, mono et multi objectif et l'approche de décomposition.

Mots clés: ordonnancement, machines parallèles, détérioration, ressources consommables, recuit simulé, makespan, TEC.

Abstract

In the majority of manufacturing systems, several types of resources can be necessary to executing jobs. However, in the literature, most of studies consider only one type of resource at a time. In this context, we study in this thesis two scheduling problems under deterioration and resources constraints; the first problem is a parallel machine scheduling problem with deterioration effect and several consumable resources, the second is a parallel machine scheduling problem with deterioration effect, skilled humans resource and scarce resources. To study these two problems, we propose two mathematical models. However, due to the complexity of the problems, the mathematical model solves only small instances of the problem. The reason for which we develop approximates methods to efficiently solve the problem. These approaches are based on simulated annealing, multi objectif simulated annealing and decomposition approach.

Keywords: scheduling, parallel machine, deterioration effect, consumable resources, simulated annealing, makespan, TEC.

ملخص

تهدف دراسة مشاكل الجدولة إلى التخطيط المسبق لتخصيص العمليات لمختلف المشغلين وتخصيص الموارد للعمليات ، مع مراعاة قيود النظام. في غالبية أنظمة التصنيع ، يتم استخدام عدة أنواع من الموارد في تنفيذ المهمة. ومع ذلك ، نادرًا ما يتم النظر في هذا البيان الأخير في الأعمال الموجودة في الأدبيات. لسد هذه الفجوة ، نتحرى في هذه الأطروحة عن مشاكل جدولة الآلات المتوازية مع تأثير التدهور وأنواع متعددة من الموارد. يعتمد المحور الأول لدراستنا على النظر في نوعين من الموارد الاستهلاكية. أما المحور الثاني فيؤخذ في الاعتبار نوعان من الموارد المتجددة وهما الموارد البشرية المؤهلة والموارد المشتركة. تم اقتراح نموذجين رياضيين لحل المشكلتين. هذه توفر الحلول المثلى بالضبط. ولكن نظرًا للتعقيد الكبير لمشكلات جدولة الآلة المتوازية مع قيود الموارد ، فإن فعالية النماذج الرياضية تقتصر على الحالات الصغيرة والمتوسطة. لهذا السبب ، تم تطوير طرق الدقة التقريبية. هذه تعتمد على التلدين المحاكى ، التلدين المحاكى متعدد الأغراض ونهج التحلل

الكلمات المفتاحية: جدولة، الآلات المتوازية ، تأثير التدهور ، الموارد المستهلكة ، الموارد المتجددة ، خوارزمية محاكاة التصلب ، الطاقة المستهلكة الإجمالية