RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ ABOU BEKR BELKAID DE TLEMCEN

FACULTÉ DE TECHNOLOGIE

D'ÉPARTEMENT DE GÉNIE ELECTRIQUE ET ELECTRONIQUE

MANUFACTURING ENGINEERING MELT LABORATORY of TLEMCEN

MÉMOIRE DE MASTER GENIE INDUSTRIEL

OPTIONS : CHAINE LOGISTIQUE

INGÉNIERIE DE LA PRODUCTION

Utilisation de l'heuristique Shifting Bottleneck pour résoudre le problème d'ordonnancement de type Job Shop

Réalisé par: **Selougha Samir**

Halassa Zakaria

Soutenu le 01 juillet 2019 devant le jury:

Président	BELKAID Fayçal	MCA	UABB Tlemcen
Examinateur	KHEDIM Amaria	MAA	UABB Tlemcen
Examinateur	TRIKI Lamia	MCA	UABB Tlemcen
Examinateur	SEKKAL Djazia Nadjat	Doctorante	UABB Tlemcen
Encadrant	HADRI Abdelkader	MAA	UABB Tlemcen

Année académique: 2018-2019

Dédicaces

Je dédie ce mémoire

A mon cher père

Et ma très chère mère

Pour l'éducation et le grand amour dont ils m'ont entouré depuis ma naissance.

Et pour leurs patiences et leurs sacrifices.

A mes chers frères

A mes chères sœurs

A tous mes proches : mes tantes, mes cousines Anis et ….. Et surtout qui m'aider pendant mon travail, mes voisins et toute la famille HALASSA.

A tous ceux qui m'aiment;

A tous mes ami (e) s;

A tous ceux que j'aime ….;

A tout mes collègues de promo Master II en Génie industriel (2018/2019) ;

HALASSA ZAKARIA

Remerciements

En premier lieu, Nous remercions avant tout Allah, tout puissant pour la volonté, la santé et la patience qu'il nous a données durant toutes ces longues années d'études afin que nous puissions arriver à ce stade.

Nous tenons à remercier Mr HADRI Abdelkader pour l'élaboration de ce travail et pour leurs conseils et leur disponibilité pendant toute la durée de la réalisation de ce mémoire.

Egalement, nous remercions l'ensemble des professeurs et enseignants de département de génie électrique et électronique de l'université de Tlemcen qui ont participé à ma formation pendant ma formation universitaire. Et à tous ce qui ont enseigné moi au long de ma vie scolaire.

Enfin, nous n'oublions pas de remercier nos amies et nos collègues et à toutes les personnes qui auront contribué de près ou de loin à l'élaboration de ce mémoire.

Dédicaces

Je dédie ce mémoire

A ma très chère mère

Et ma sœur

ET mon grand père

Pour l'éducation et le grand amour dont ils m'ont entouré depuis ma naissance.

Et pour leurs patiences et leurs sacrifices.

A mes chers frères ;

A mes chères sœurs ;

A tous mes proches : mes tantes, mes cousines et ….. Et toute personne m'aider pendant ce travail, mes voisins et toute les familles SELOUGHA et SEMASSEL.

A tous ceux qui m'aiment;

A tous mes ami (e) s;

A tous ceux que j'aime …;

A tous mes collègues de promo Master II en Génie industriel (2018/2019) ;

SELOUGHA SAMIR

SOMMAIRE

Liste des figures

Liste des tableaux

INTRODUCTION GENERALE

Introduction générale

Dans l'environnement des entreprises industrielles particulièrement qui sont caractérisées par le changement, la complexité, la concurrence intense. Il est exigé qu'elles résistent toujours aux défis auxquels cet environnement est confronté aussi rapidement que nécessaire, dans les délais indiqués et avec une grande efficacité. Ceci est à l'aide des méthodes technologiques modernes dans les processus de la production et de la commercialisation, et de l'utilisation des stratégies compétitives modernes. La gestion et l'ordonnancement de production sont les méthodes les plus importantes pour renforcer et améliorer leurs compétences de concurrence et le développement des produits de haute qualité et au plus bas coût possible de manière efficace. Ceci leur permet d'augmenter la compétitivité de ses entreprises.

Les avantages de l'ordonnancement de la production par rapport à la productivité des entreprises sont reconnus. Un plan solide et efficace maximise la production en assurant la disponibilité des matériaux, de l'équipement et des ressources humaines au bon moment et au bon endroit. Parmi eux, la réduction des coûts et des dépenses grâce à l'élimination des pertes de temps, et l'optimisation de l'utilisation de l'équipement et augmentation de la capacité etc.

Puisqu'il existe une grande diversité de systèmes de production dans l'organisation des ateliers industrielle, on classe ces ateliers selon l'ordre de machines et le routage des jobs ; En effet, on trouve dans la littérature les problèmes d'ordonnancement les plus étudiés qui sont: machines parallèles, flow-shop, job-shop, Open-shop etc.

Parmi les problèmes d'ordonnancement les plus difficiles et les plus étudiés, nous avons le problème d'ordonnancement d'ateliers de type Job Shop. Ce problème consiste à trouver une séquence optimale pour l'exécution de n jobs sur m machines afin d'optimiser une fonction objective (minimisation de la durée totale d'achèvement de toutes les tâches, minimisation du retard total des tâches, minimisation de la charge des machines, etc.).

L'aspect le plus difficile du problème du Job Shop est de trouver et d'obtenir la solution optimale de manière précise, ce qui est presque impossible lorsqu'on augmente le nombre des machines et le nombre des jobs. Cela a poussé les chercheurs en ordonnancement à développer des approches et des méthodes approximatives pour trouver une bonne solution dans un temps raisonnable. Parmi ces dernières, il y a les heuristiques et les méta-heuristiques.

L’avantage de ces méthodes, c’est qu’elles donnent un bon compromis entre le coût de résolution et la qualité des solutions dans le problème d’ordonnancement de Job Shop.

Dans ce travail, nous allons résoudre un problème d’ordonnancement d’atelier de type Job shop sans contrainte dont l’objectif de notre travail est d'étudier et appliquer une des méthodes heuristiques dédiées aux problèmes d'ordonnancement de Job Shop c’est l’heuristique « Shifting bottleneck », qui a été proposée par Adams en 1988.

Ce mémoire est organisé dans l'ordre suivant :

Dans le premier chapitre, nous nous intéressons par les généralités sur les problèmes d’ordonnancement dans les systèmes de production et les caractéristiques nécessaires de ces problèmes.

Dans le deuxième chapitre, nous présentons les formulations des problèmes d’ordonnancement d’atelier de type job shop et ses méthodes de résolution qui peuvent être des approches exactes ou des approches approximatives.

Dans le troisième chapitre, nous présentons l’heuristique « Shifting bottleneck » pour résoudre le problème d'ordonnancement d'un système de production de type Job-Shop afin de minimiser le makespan et donner une synthèse des résultats obtenus par l’application de cette heuristique (Shifting bottleneck) pour la résolution d’un ensemble de problèmes de Job shop à travers des tests que nous avons faits.

CHAPITRE I

L'ordonnancement des activités de production

CHAPITRE I

L'ordonnancement des activités de production

1. Introduction

L'objectif principal d'une entreprise industrielle est d'assurer des bénéfices sur les biens et les services qu'elle vend à ses clients. En effet, l'atteinte de cet objectif dépend de son capacité à satisfaire sa clientèle qui devient de plus en plus exigeante vu la grande concurrence qui règne sur le marché industriel.

Pour ce faire, l'entreprise doit avoir recours à une bonne organisation de la production en s'appuyant sur un certain nombre d'outils tel que l'ordonnancement, qui est l'un des parties principales de la planification.

L'objectif de ce chapitre est de donner quelques généralités sur les problèmes d'ordonnancement dans les systèmes de production. Une description sur les systèmes de production et la gestion de ces derniers est alors présentée dans la première section. Dans la deuxième section nous allons présenter les caractéristiques des problèmes d'ordonnancement. La troisième section est dédiée à la démonstration des différents types de problèmes, leurs modélisations et leurs représentations.

2. Les systèmes de production

2.1. Définitions

Un système de production : est un système artificiel composé d'unités organisées qui interagissent et interfèrent dans le but de produire des biens ou des services. Un système de production est soumis à une charge qui définit l'ensemble des biens ou des services que le système doit réaliser [1].

La charge d'un système de production (Jobs) : c'est le travail que le système doit réaliser et elle est généralement constituée d'un ou plusieurs jobs. Chaque job subit un processus définition la réalisation d'un produit, de la matière première jusqu'au produit fini (gamme de fabrication).

Gamme de fabrication : Une gamme décrit la suite des opérations qu'un job doit réaliser. Une gamme correspond donc à un type particulier de produits. Par laquelle on peut définir les opérations à réaliser en donnant, leurs durées, la ou les machines qui sont concernées et même aussi les ressources à consommer.

Limites de systèmes de production Comme tout système de la vie réelle, les systèmes de production sont finis. Ils ne peuvent donc pas écouler une charge illimitée de travail. Ce sont des ressources présentes en quantités limitées: des machines, des opérateurs, des matières premières, des places en stocks.

Décomposition de système de production : Un système de production peut se décomposer en trois sous-systèmes : le système physique, le système de décision et le système d'information [2] [3] .

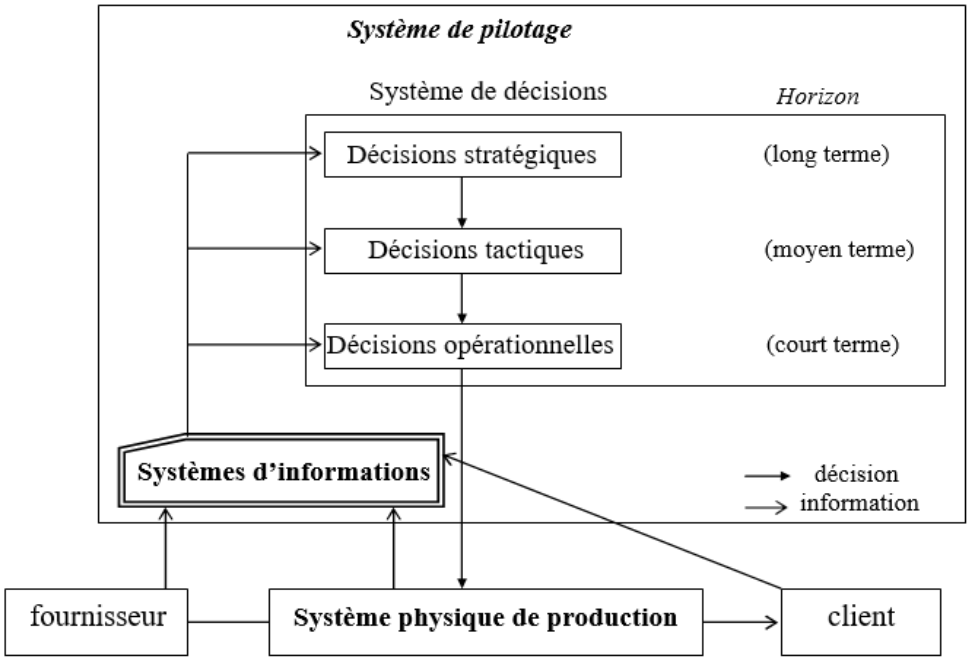

Figure 1: *Décomposition de système de production [4] [2]*

- ✓ **Le système physique de production:** est dédié à la Transformation des matières premières ou des composants en produits finis. Il est constitué de ressources humaines et physiques.
- ✓ **Le système de décision:** Contrôle le système physique de production. Il en coordonne et organise les activités en prenant des décisions basées sur les données transmises par le système d'information.

- ✓ **Le système d'information:** Intervient à plusieurs niveaux: à l'interface entre les systèmes de décision et de production; à l'intérieur du système de décision, pour la gestion des informations utilisées lors de prises de décisions ; et à l'intérieur du système physique de production. Son rôle est de collecter, stocker et transmettre des informations de différents types [4].

2.2. La gestion de production

Définitions : La gestion de production est un ensemble de processus qui permet de mener à bien la fabrication de produits à partir d'un ensemble de données et de prévisions [5].

[6] Définit la gestion de production comme étant « la fonction qui permet de réaliser les opérations de production en respectant les conditions de qualité, délai, coûts qui résultent des objectifs de l'entreprise.

En fait, la gestion de production s'occupe d'un ensemble de problèmes liés à la production tels que la gestion des données, la planification, le contrôle (suivi) de la production, la gestion des stocks, la prévision, l'ordonnancement etc. [7] .

*Les niveaux de la gestion de la production :*En général, les niveaux hiérarchiques de la gestion de production couramment retenus sont en nombre de trois : stratégique, tactique et opérationnel [8] :

- ✓ **Le niveau stratégique:** Il s'agit de la formulation de la politique à long terme de l'entreprise (à un horizon de plus de deux ans). Elle porte essentiellement sur la gestion des ressources durables, afin que celles-ci soient en mesure d'assurer la pérennité de l'entreprise.
- ✓ **Le niveau tactique:** Il s'agit de décisions à m1oyen terme. Elles assurent la liaison entre le niveau stratégique et le niveau opérationnel. L'objectif est de produire au moindre coût pour satisfaire la demande prévisible, en s'inscrivant dans le cadre fixé par le plan stratégique de l'entreprise.
- ✓ **Le niveau opérationnel:** Il s'agit des décisions à court et à très court terme. C'est une gestion quotidienne pour faire face à la demande au jour le jour, dans le respect des décisions tactiques. Parmi les décisions opérationnelles, on trouve : la gestion des stocks, la gestion de la main d'œuvre, la gestion des équipements [6].

2.3. Rôle de l'ordonnancement en gestion de production

L'ordonnancement en gestion de production correspond aux concepts et techniques qui règlent les problèmes de priorités et l'organisation des flux de production. Dans un environnement de production complexe, l'ordonnancement peut devenir un problème extrêmement difficile à résoudre [9].

L'ordonnancement est un aspect important de la gestion de production. La connaissance des méthodes d'ordonnancement permet d'éviter des pertes de performance importantes. Mais d'un point de vue général, l'optimisation de l'environnement de production apporte plus de valeur ajoutée à l'entreprise que la recherche d'une méthode d'ordonnancement optimal.

La place de l'ordonnancement varie entre le niveau tactique et le niveau opérationnel. IL s'occupe de la réalisation des décisions venant de niveaux supérieurs.

3. L'ordonnancement de la production

Plusieurs définitions différentes de l'ordonnancement ont été proposées dans la littérature et dans ces définitions, on peut remarquer deux éléments communs ; l'affectation des de ressources aux tâches et la recherche d'optimiser certains critères (objectif). Parmi les définitions les plus rencontrées:

« L'ordonnancement est une branche de la recherche opérationnelle et de la gestion de la production qui vise à améliorer l'efficacité d'une entreprise en termes de coûts de production et de délais de livraison. Les problèmes d'ordonnancement sont présents dans tous les secteurs d'activités de l'économie, depuis l'industrie manufacturière » [10].

« Ordonnancer le fonctionnement d'un système industriel de production consiste à gérer l'allocation des ressources au cours du temps, tout en optimisant au mieux un ensemble de critères » [11].

« L'ordonnancement est la programmation dans le temps de l'exécution des tâches en leur attribuant les ressources nécessaires, matérielles ou humaines de manière à satisfaire un ou plusieurs critères préalablement définis, tout en respectant les contraintes de réalisation » [12].

« Le problème d'ordonnancement consiste à déterminer quelles opérations doivent être exécutées, et à assigner des dates et des ressources à ces opérations de façon à ce que les tâches soient, dans la mesure du possible, accomplies en temps utile, au moindre coût et dans les meilleures conditions » [5].

4. Formulation d'un problème d'ordonnancement

Le problème d'ordonnancement est parmi les problèmes les plus difficiles à résoudre et elle se trouve presque dans tous les domaines ; informatique, services, production …etc. Dans tous ces domaines, un problème d'ordonnancement se définit par quatre éléments fondamentaux qui sont les tâches, les ressources, les contraintes et les critères.

4.1. Les tâches

« Une tâche est une entité élémentaire de travail localisée dans le temps par une date de débute une date de fin d'exécution et qui consomme des ressources avec des quantités déterminées. Un coût (ou poids) est attribué à une tâche pour estimer sa priorité, son degré d'urgence, ou son coût d'immobilisation dans le système » [12]

Deux types de tâches sont rencontrés :

- *Les tâches morcelables* (préemptives) qui peuvent être exécutées en plusieurs fois, facilitant ainsi la résolution de certains problèmes.
- *Les tâches non morcelables* (indivisibles) qui doivent être exécutées en une seule fois et ne sont interrompues qu'une fois terminées.

Une tâche généralement est caractérisé par:

- Durée d'exécution de l'opération
- La date de disponibilité de la tâche (date de début au plus tôt) .
- La date de début au plus tard de la tâche.
- La date de fin au plus tôt.
- La date de fin au plus tard de la tâche.

4.2. Les ressources:

Une ressource est un moyen technique ou humain, destiné à être utilisé pour la réalisation d'une tâche et disponible en quantité limitée [12].

Une ressource est renouvelable si après chaque utilisation elle reste disponible dans la même quantité et dans le même état de fonctionnement qu'avant l'utilisation. (Les hommes, les machines, l'espace, l'équipement en général, etc.). Lorsque la quantité d'une ressource diminue au fur et à mesure de son utilisation, on parle d'une ressource consommable (matières premières, l'énergie, budget, etc.)

Les ressources disjonctives (ou non partageables) sont des ressources qui peuvent être allouées à une selle tâche à la fois (machine-outil, robot manipulateur). Les ressources cumulatives (ou partageables) peuvent être utilisées par plusieurs tâches simultanément (équipe d'ouvriers, poste de travail).

Dans notre travail, on s'intéresse uniquement par les ressources renouvelables et disjonctives qui sont les machines.

4.3. Les contraintes

Suivant la disponibilité des ressources et suivant l'évolution temporelle, deux types de contraintes peuvent être distingués: contraintes de ressources et contraintes temporelles [13].

- ✓ **Les contraintes de ressources**:

Plusieurs types de contraintes peuvent être induits par la nature des ressources. A titre d'exemple, la capacité limitée d'une ressource implique un certain nombre, à ne pas dépasser, de tâches à exécuter sur cette ressource.

Les contraintes relatives aux ressources disjonctives, induisant une contrainte de réalisation des tâches sur des intervalles temporels disjoints pour une même ressource, ou cumulatives impliquant la limitation du nombre de tâches à réaliser en parallèle.

- ✓ **Les contraintes temporelles**:

Elles représentent des restrictions sur les valeurs que peuvent prendre certaines variables temporelles d'ordonnancement. Ces contraintes peuvent être:

- Des contraintes de dates butoirs, certaines tâches doivent être achevées avant une date préalablement fixée.
- Des contraintes de précédence, une tâche *i* doit précéder la tâche *j*.
- Des contraintes de dates au plus tôt, liées à l'indisponibilité de certains facteurs nécessaires pour commencer l'exécution des tâches.

4.4. Les critères

Un critère correspond à des exigences qualitatives et quantitatives à satisfaire permettant d'évaluer la qualité de l'ordonnancement établi. L'objectif d'un problème d'ordonnancement consiste à optimiser un ou plusieurs critères ou à trouver une solution admissible qui respecte les contraintes du problème**.** On peut classer les critères en critères réguliers et en critères irréguliers [14].

Les critères réguliers : Ils sont dits réguliers car ils constituent des fonctions croissantes des dates d'achèvement des opérations. Nous citons, à titre d'exemple :

- La minimisation des dates d'achèvement des actions.
- La minimisation du maximum des dates d'achèvement des actions.
- La minimisation de la moyenne des dates d'achèvement des actions.
- La minimisation des retards sur les dates d'achèvement des actions.
- La minimisation du maximum des retards sur les dates d'achèvement des actions.
- La minimisation de la moyenne des retards sur les dates d'achèvement des actions.
- La minimisation du temps du cycle (dans le cas d'un ordonnancement cyclique).

Les critères irréguliers : Ils ne sont pas des fonctions monotones des dates de fin d'exécution des opérations. Parmi ces types de critères, on cite:

- La minimisation des encours.
- La minimisation du coût du stockage des matières premières.
- L'équilibrage des charges des machines.

5. Les différents types de problèmes d'ordonnancement

5.1. Notation des problèmes (trois champs α, β, γ)

Plusieurs notations ont été proposée dans la littérature [15] Pour représenter un problème d'ordonnancement d'une manière simple. Cette notation consiste à représenter le problème sous forme de trois champsα, β, γ.

5.1.1. Le champ α

Ce champ représente l'environnement machines possibles, permet identifier le type de problème ou d'atelier.

5.1.2. Le champ β

Ce champ spécifie les caractéristiques des tâches et les contraints d'ordonnancement, parmi plusieurs des contraints on se trouve:

- *Dates de disponibilité :* disponibilité des taches.
- *Préemption (prmp):* les opérations peuvent être exécutées par morceaux.
- *Précédence (prec):* certaines taches doivent être exécutées avant d'autres.
- *Pannes (brkdwn):* les ressources (machines) ne sont pas disponibles en permanence.
- *Permutation (prmu):* il n'y a pas de permutation possible entre les taches dans un flow shop.

- *Re-circulation (recrc):* les taches peuvent passer plus d'une fois par la même machine (environnement Jm).
- *Temps de changement* **(s_{jk}):** dépendant de la séquence**.**
- *Restriction sur les machines (Mj):* la tâche ne peut pas être effectuée par n'importe quelle machine (environnement Pm).

5.1.3. Le champ γ

Ce champ représente la fonction objective ou le critère à optimiser.

Les problèmes sont caractérisés par le nombre de machines dans l'atelier et leur disposition, des nombres d'opération composant les jobs, et des ordres de leurs passages sur les machines.

5.2. Problèmes à une machine

Dans ce problèmes l'ensemble des tâches à réaliser est fait par une seule machine. Les tâches alors sont composées d'une seule opération qui nécessite la même machine.

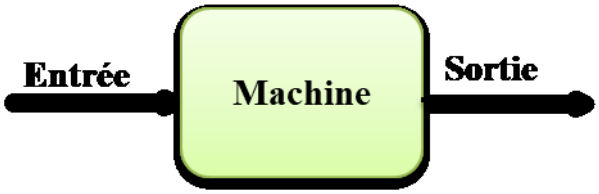

Figure 2: *Problèmes à une machine*

5.3. Problèmes à machines parallèles

Le problème d'ordonnancement à machines parallèles est une généralisation du problème à une machine et un cas particulier de problème multi machines. Chaque job est constitué d'une seule opération et chaque opération peut être réalisée par n'importe laquelle des machines, disposées en parallèle, mais n'en nécessite qu'une seule.

- Si toutes les machines ont la même vitesse d'exécution des tâches, les machines sont appelées « **identiques** » et le problème noté (**P**).
- Si les machines sont différentes par leur vitesse d'exécution et la vitesse de chaque machine est constante et ne dépend pas de l'ensemble des tâches, elles sont dites « **uniformes** ». (**Q**).
- Si les vitesses d'exécution des machines dépendent des tâches et sont différentes alors elles sont dites « **quelconques** » ou différentes (**R**).

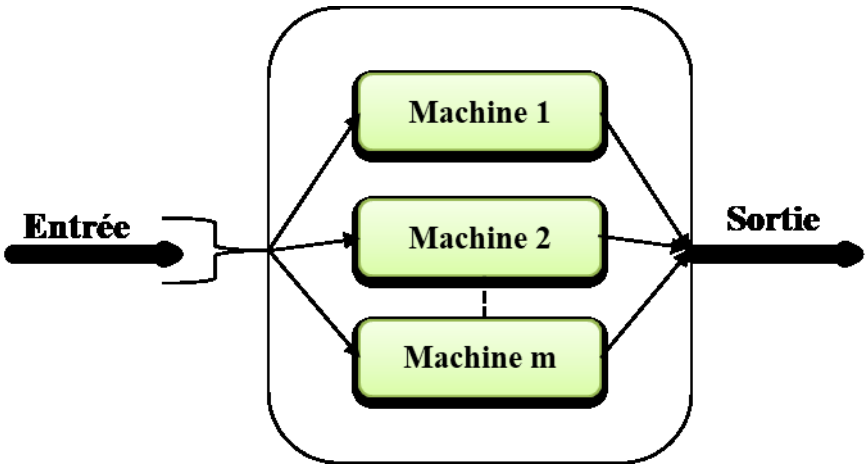

Figure 3: *Problèmes à machines parallèles*

5.4. Problèmes de Flow shop(Fm)

Dans ce problème les machines sont disposées en série et les jobs à réaliser sont composés de plusieurs opérations et visitent toutes les machines selon une gamme opératoire (gamme de fabrication) unique.

- M machines en série.
- Chaque job passe par toutes les machines.
- Le routage est identique pour tous les jobs.
- On note en général (**F**)

Figure 4:*Problèmes de Flow shop*

5.4.1. Flow shop hybride (Flexible Flow shop, FFc):

L'atelier est organisé en étages constitués d'un ensemble de machines en parallèle. Cependant, une opération(tache) n'en nécessite qu'une seule pour son exécution. Les différents jobs à réaliser doivent passer sur tous les étages dans le même ordre. Ceci revient donc à trouver pour chaque job la machine exécutant l'opération associée chaque étage, ainsi que les dates d'exécution des différentes opérations.

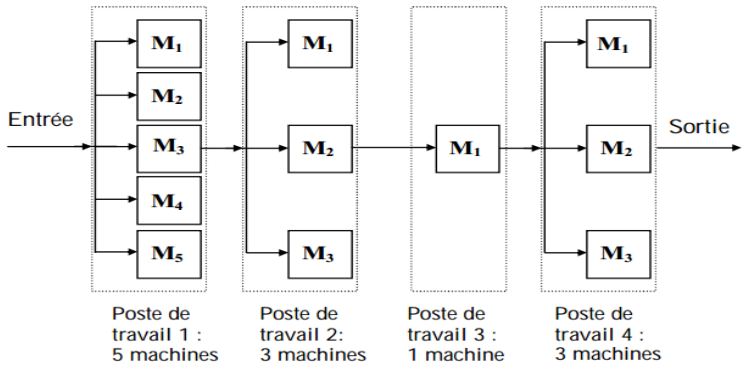

Figure 5 : *Exemples de Flow Shop simple et hybride*

- S stations de travail en série, chaque station est composée de machines en parallèle.
- Chaque job passe par toutes les stations.
- Le routage est identique pour tous les jobs.
- Au niveau de chaque station la tache peut être exécutée par n'importe quelle machine.

5.5. Problèmes de job shop(Jm)

C'est une généralisation de celui du flow shop, la gamme de fabrication est linéaire et fixée à l'avance mais chaque tâche possède son propre mode de passage sur les machines.

- ✓ Le nombre d'opération n'est pas forcément le même pour tous les jobs.
- ✓ Chaque job a son propre ordre de passage sur les machines.
- ✓ On note en général (Jm)

Parmi les autres caractéristiques d'un problème d'ordonnancement dans un atelier à cheminements multiples:

- Le nombre de solutions possibles est de l'ordre de $(n!)^m$, où n est le nombre de tâches à effectuer et m le nombre de machines. Notons qu'une tâche veut dire la même chose qu'un travail.

➢ Le problème est considéré parmi les problèmes les plus difficiles à traiter.

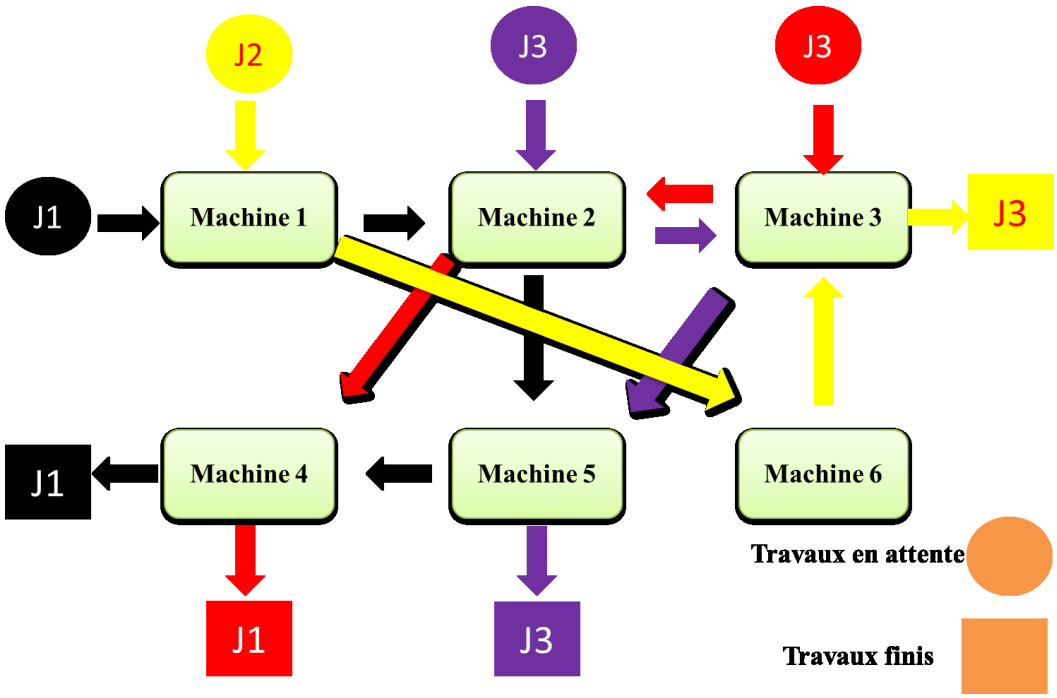

Figure 6: *Exemple Problèmes de job shop*

5.5.1. Job shop flexible (FJc)

La différence est que pour le problème du job shop flexible, chaque opération peut être effectuée par une seule machine dans un ensemble de machines. Le problème est ainsi de déterminer à la fois une affectation et un séquencement des opérations sur les machines en fonction de l'objectif à atteindre. Le problème du job shop flexible est pertinent dans au moins deux types de systèmes de production. Dans les systèmes manufacturiers flexibles, les machines peuvent effectuer différents types d'opérations. Le deuxième type consiste en des ateliers avec des pools de machines parallèles, où les machines d'un pool peuvent effectuer uniquement un seul type d'opérations.

5.6. Problèmes de Open Shop(Om)

Dans ces problèmes, le cheminement des jobs est multiple, mais à la différence du job shop, les jobs ne possèdent pas de gammes. L'ordre de passage des opérations est ainsi quelconque (la gamme de fabrication libre). Ce paramètre est déterminé lors de l'ordonnancement. L'open-shop n'est pas très étudié dans la littérature, ceci étant dû au fait qu'il n'est pas courant dans les entreprises, on note en général (O_m)

6. Modélisation et représentation des problèmes d'ordonnancement

6.1. Les méthodes mathématiques

Dans le cas des méthodes mathématiques, les données, les contraintes et la fonction d'évaluation des critères sont écrites sous forme d'équations et d'inéquations mathématiques. Ces méthodes, couramment utilisées, ont l'avantage d'être simples et directement exploitables par les algorithmes de résolution [12].

6.2. Le diagramme de Gantt

Le diagramme de Gantt, appelé aussi diagramme à barres, du nom de son développeur Henry Gantt, Il est devenu le moyen le plus simple et célèbre pour représenter une solution d'un ordonnancement. Sur ce graphique, on visualise l'enchaînement des opérations et les ressources les exécutant. Dans un ordonnancement sur machines parallèles identiques, l'axe des abscisses représente le temps et sur l'axe des ordonnées apparaissent les machines. Sur chaque ligne horizontale, on met l'ordonnancement des tâches sur cette machine. Chaque tâche est représentée par une barre. La longueur de cette barre est proportionnelle à sa durée. A la fin, on obtient sur le diagramme l'enchaînement de opérations sur chacune des machines, avec les dates de début et de fin de chaque tâche.

Exemple: représente un ordonnancement de 11 jobs sur 3 machines parallèles identiques

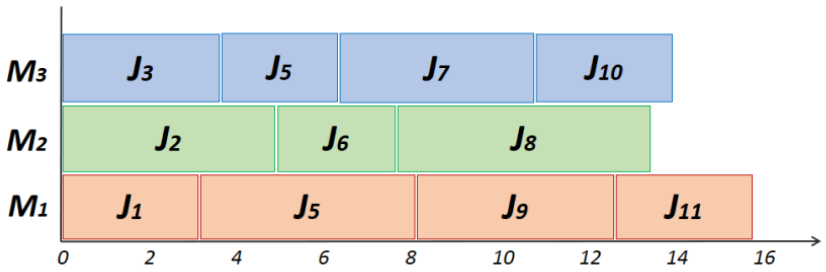

Figure 7:*Exemple Diagrammes de Gantt de 11 Jobs sur 3 machines.*

6.3. Graphe Potentiel-Tâches (disjonctif)

Un graphe de précédence est constitué d'un ensemble de nœuds et de deux types d'arcs. Les nœuds représentent les tâches à réaliser. Les arcs conjonctifs values de la durée de réalisation de la tâche d'où l'arc sort, représentent les contraintes de précédence. Les arcs disjonctifs à deux sens représentent les contraintes de ressources. Pour représenter le début et la fin de l'ordonnancement, deux nœuds fictifs sont rajoutés au graphe [16].

La figure 8 représente un problème d'ordonnancement composé de trois ressources notées A, B et C et 8 tâches numérotées de 1 à 8. Les tâches 1, 5 et 6 doivent se réaliser sur la même ressource, pareil pour les tâches 2, 4 et 7 et 3 et 8. Le début et la n de l'ordonnancement sont représentés respectivement par les nœuds fictifs "s" et "p". Le fait de fixer un sens à chaque arc disjonctif réalise un ordonnancement [16].

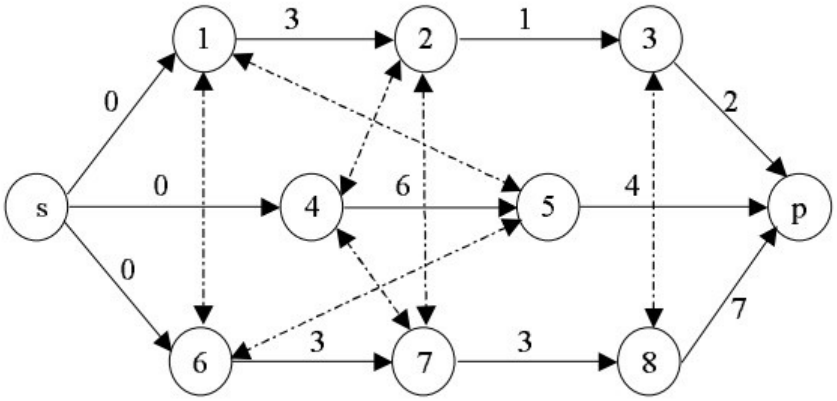

Figure 8: *Graphe Potentiel-Tâches (disjonctif)*

7. Les classes d'ordonnancement

Avant de développer les méthodes de résolution, nous définissons les différentes classes de l'ordonnancement. L'espace des ordonnancements peut être partagé en sous classes en fonction de l'affectation des tâches dans le temps.

7.1. Ordonnancement admissible (acceptable)

Un ordonnancement est dit admissible ou acceptable si l'ordre d'affectation des tâches respecte les contraintes du problème**.**

Exemple: Le problème de Job Shop 3-tâches 3-machines

J1	$M_1(4)$	$M_2(3\ )$	$M_3(3)$
J2	$M_1(1\ )$	$M_3(5)$	$M_2(3)$
J3	$M_2(2\ )$	$M_1(4)$	$M_3(1)$

Tableau 1: *Problème de Job Shop 3 tâches 3 machines*

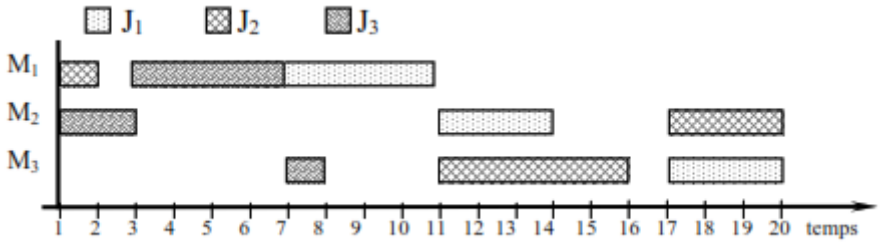

Figure 9:*Exemple d'un ordonnancement admissible*

Dans certains cas, des décalages à gauche sur certaines opérations sont nécessaires. Selon que l'ordre des opérations reste inchangé ou non, on distingue deux cas [10]:

- ✓ On parle de décalage à gauche "*local*", lorsqu'on avance le début d'une opération sans remettre en cause l'ordre relatif entre les autres opérations.
- ✓ On parle de décalage à gauche "*global*", lorsqu'on avance le début d'une opération en modifiant l'ordre relatif entre au moins deux opérations.

7.2. Ordonnancement semi-actif

Un ordonnancement est dit semi-actif, si aucun décalage à gauche local n'est possible. On ne peut exécuter aucune opération plus tôt, sans altérer l'ordre relatif au moins de deux opérations. Toutes les opérations sont calées, soit sur l'opération qui précède dans la gamme, soit sur l'opération qui précède sur la machine utilisée. A titre d'exemple, l'ordonnancement présenté sur la figure 10 est semi-actif, aucun décalage local n'est possible.

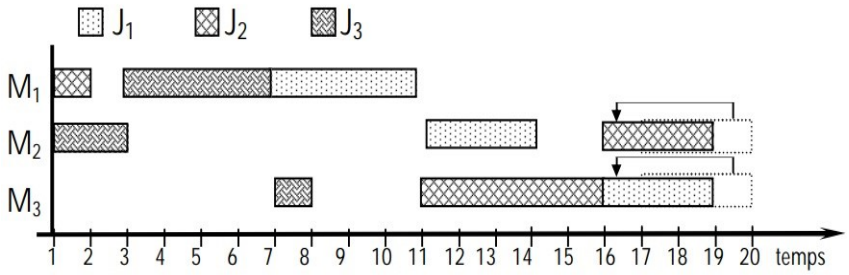

Figure 10:*Décalage à gauche local de O1,3 et O2,2 dans l'ordonnancement de la figure 9, conduit à un ordonnancement semi-actif.*

7.3. Ordonnancement actif

Un ordonnancement est actif si au un décalage ou permutation de es tâches n'est possible [10]. Au une tâche ne peut commencer plus tôt sans reporter le début d'une autre.

L'ordonnancement présenté sur la figure 11 est actif puisque aucun décalage à gauche, local ou global ne peut être effectué.

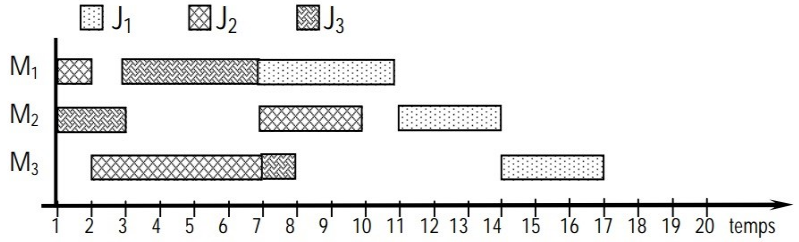

Figure 11: *Exemple d'ordonnancement actif.*

7.4. Ordonnancements sans délai

Un ordonnancement est dit sans délai ou sans retard, si et seulement si aucune opération n'est mise en attente alors qu'une machine est disponible pour l'exécuter [17].

Ainsi, l'ordonnancement illustré sur la figure 11, bien qu'il soit actif, il n'appartient pas à cette classe (*i.e.* il est avec retard). En effet, l'opération $O_{1,1}$ qui attend la machine M_1 est mise en attente, alors que M_1 est oisive à $t = 1$. On peut le transformer sans retard comme cela est illustré sur la figure 12 :

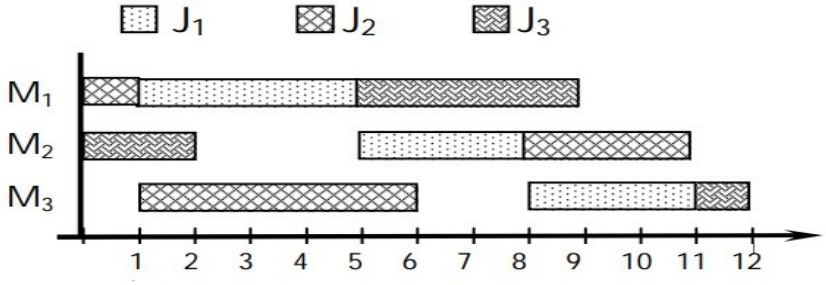

Figure 12: *Exemple Ordonnancement sans délai.*

A noter que la transformation à un ordonnancement sans délai peut mener à une solution plus mauvaise du point de vue de makespan [12].

La figure 13 représente les relations d'inclusion des classes d'ordonnancements vues précédemment.

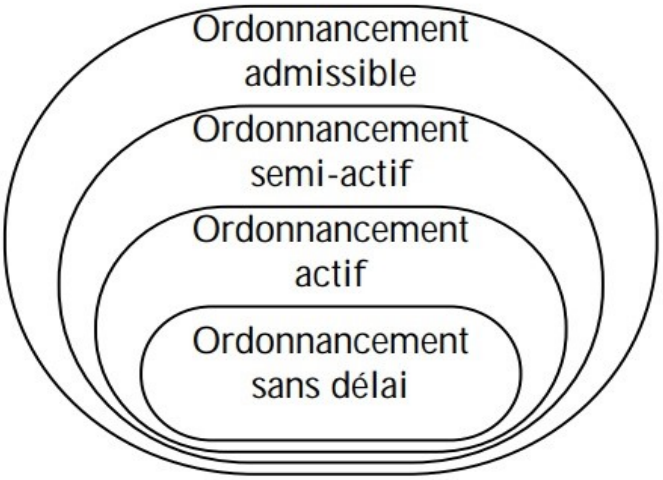

Figure 13: *Relations d'inclusion entre les différentes classes d'ordonnancements [10] .*

8. Conclusion

Au travers ce chapitre nous avons présenté les systèmes de production et la gestion de production, le rôle de l'ordonnancement au sein de ces systèmes et présenté les concepts et les bases fondamentales du problème d'ordonnancement et la caractérisation du problème d'ordonnancement d'atelier en présentant sa définition, et en précisant notamment les différents éléments qui le déterminent, à savoir : les tâches, les ressources, les contraintes et les critères d'optimisation ; ainsi que les approches de classification qui s'appuient sur un ensemble de paramètres portant sur différentes caractéristiques du problème dont l'organisation d'ateliers Job Shop, Flow Shop, Open Shop) décrites par la suite, constitue un champ important de la classification.

CHAPITRE II

Le problème d'ordonnancement de Job Shop

CHAPITRE II

Le problème d'ordonnancement de Job Shop

1. Introduction

Le problème d'ordonnancement de Job Shop est un cas particulier du problème général d'ordonnancement. C'est l'un des problèmes de la théorie de l'ordonnancement et de l'optimisation combinatoire les plus traités [18].

Dans ce chapitre, on va présenter les termes théoriques et pratiques importants liés au problème, en se concentrant sur sa spécificité en terme de contraintes, difficulté, modélisation et méthodes de résolution. Ainsi, la deuxième section est consacrée à la description du problème : les données, les contraintes et les critères d'évaluation. La troisième section les approches importantes de présentation du problème. La complexité du problème sont données à la quatrième section. Dans la cinquième section, nous discutons un résumé des méthodes de résolution connues en citer les importantes méthodes envisagées pour la résolution du problème.

2. Présentation du problème de Job Shop

Le problème de Job Shop constitue l'un des problèmes d'ordonnancement d'atelier les plus étudiés. Il existe de nombreuses variations autour du problème de Job Shop. Nous donnons ici la formulation la plus générale possible du problème de Job Shop, tout en précisant ensuite les restrictions qui caractérisent le cas du Job Shop traité dans ce mémoire.

Le problème d'ordonnancement de Job Shop consiste réaliser un ensemble de n job sur un ensemble de m ressources (machines) en cherchant d'atteindre certains objectifs. Chaque job J_i est composé d'une suite de n_i opérations devant être exécutées sur les différentes ressources selon un ordre préalablement défini. Par ailleurs, un ensemble de contraintes concernant les ressources et les tâches doivent être respectées.

Donc, la détermination du problème de Job Shop m × n, constitué de n Job et m machines, se fait en définissant les données, les contraintes et les objectifs du problème.

2.1. Les données

Les données du problème sont [19] [5]:

- **Un ensemble *M* de *m* machines :** Une machine est notée M_k avec $k = 1, m$. Chaque machine ne peut effectuer qu'un seul type d'opérations. Le nombre total d'opérations exécutées par cette machine est noté m_k.
- **Un ensemble *J* de *n* tâches :** Une tâche est notée J_i avec $i = 1, n$. Chaque tâche est composée d'une gamme opératoire, *i.e.* une séquence linéaire fixée de n_i opérations. Cette séquence ne dépend que de la tâche, et peut varier d'une tâche à l'autre.

Cet ensemble d'opérations d'une tâche $J_i \in J$, est défini par :

$$O_i = \{O_{i,1} \bullet \dots \bullet O_{i,k}\}, \text{ tel que : } O_i \subset O ;$$

Où • est l'opérateur de précédence. Il définit un ordre total sur l'ensemble des opérations de la même gamme.

- L'opération $O_{i,j}$ est la $j^{ème}$ opération dans la gamme opératoire de J_i. Elle se caractérise par :
  - La machine sur laquelle elle s'exécute : M_k . Le fait que la machine demandée par l'opération $O_{i,j}$ soit M_k s'écrit : $R(O_{i,j}) = M_k$.
  - Le temps opératoire $p_{i,j}$ qui correspond à la durée de $O_{i,j}$ sur M_k.

Le nombre total des opérations dans l'atelier est noté : $n_o = \sum_{i=1}^{n} n_i$.

2.2. Les contraintes

Cet atelier, introduit précédemment, est soumis à des contraintes qui touchent à la fois les possibilités d'utilisation des machines et les liens qui peuvent exister entre les opérations.

En outre, les contraintes diffèrent d'une formulation l'autre (selon le type du Job Shop). Mais, nous ne retiendrons ici que le cas du Job Shop simple, dont les contraintes sont les suivantes :

- Les machines sont indépendantes les unes des autres (pas d'utilisation d'outil Commun, par exemple).
- Chaque machine est disponible pendant toute la période de l'ordonnancement.

- Une machine ne peut exécuter qu'une seule opération à un instant donné.
- Les jobs sont indépendants les uns des autres. En particulier, il n'existe aucun Ordre de priorité attaché aux jobs.
- Une opération ne peut être en état d'exécution que sur une seule machine a la

  Fois. Deux opérations du même job ne peuvent être exécutées simultanément.
- Une opération en cours d'exécution ne peut pas être interrompue (pas de préemption).
- Les opérations sont autorisées d'attendre les ressources autant qu'il faut. Il n'y a pas de dates d'échéance.
- Seulement le temps d'exécution proprement dit, est pris en compte. Les temps de transport d'une machine l'autre, de préparation, etc. ne sont pas considérés.

2.3. Les objectifs

L'objectif du problème d'ordonnancement est de fixer les dates de début des opérations. Pour cela, il faut déterminer l'ordre de passage de l'ensemble des tâches sur chaque machine, en respectant les contraintes du problème. Le but est ensuite, de minimiser une fonction objective, pour trouver la ou les meilleure(s) solution(s). Cette fonction objective s'appelle aussi dans ce contexte : critère de performance, critère d'évaluation ou encore objectif de l'ordonnancement [20].

Le résultat d'un ordonnancement, généralement représenté sur un diagramme de Gantt, est fonction des dates de début calculées des opérations [18].

Dans notre cas, et pour tous les problèmes traités, nous prendrons comme objectif, la durée totale de l'ordonnancement **Cmax**, puisque d'un côté, c'est un critère simple et est le plus utilisé dans la littérature ; et d'un autre côté, notre travail n'exige pas des critères trop compliqués, un critère simple suffit à la réalisation de l'étude.

3. Modélisation graphique du problème de Job Shop

Traiter un problème d'une telle complexité que celle du Job Shop nécessite une modélisation claire et précise. Il existe diverses manières de modéliser ce problème, et qui conduisent à choisir un mode de description adéquat.

Si la modélisation par graphe disjonctif est la plus classique et la plus répandue, les modélisations en programmation mathématique sont les plus anciennes. Il existe également, des modélisations algébriques, polyédrales, par les graphes potentiels-tâches, par les réseaux de Pétri, des modélisations basées sur UML etc.

3.1. Modélisation par graphe disjonctif

La modélisation sous forme d'un graphe disjonctif est très utilisée pour représenter les problèmes d'ordonnancement et en particulier les problèmes de Job Shop. Présentée pour la première fois par B. Roy et B. Sussmann en 1964, cette formulation a été à l'origine des premières méthodes de résolution du problème [18] [21].

Un problème de Job Shop peut alors se modéliser par un graphe disjonctif G (X, $C \cup D$) où :

- X : est l'ensemble des *sommets* : chaque opération correspond à un sommet, avec deux opérations fictives S (source) et P (puits) désignant le début et la fin de l'ordonnancement.
- C : est l'ensemble des arcs *conjonctifs* représentant les contraintes d'enchaînement des opérations d'une même tâche (gammes opératoires). Pour un arc (ij,ik) de la partie conjonctive, on a :

$$t_{ij} - t_{ik} \geq p_{i,k} \; \forall \; (ij,ik) \in C$$

On aura donc un arc entre tous les sommets (i, j), $(i, j+1)$ pour $i = 1 \ldots n$ *et* $j = 1 \ldots n_i - 1$; (n : nombre de tâches, n_i : nombre d'opérations de la tâche i). De plus, il existe des arcs entre S et tous les sommets $(i,1)$, et d'autres arcs entre les sommets (i, n_i) et le puits P.

- D: l'ensemble des arcs disjonctifs associés aux conflits d'utilisation d'une machine. Pour un arc (ij, ik) de la partie disjonctive, on a [12] :

$$t_{jk} - t_{ik} \geq p_{ik} \text{ ou } t_{ik} - t_{jk} \geq p_{jk} \; \forall \; (ik,jk) \in D$$

Que l'on modélise par un arc et un arc retour entre les sommets (t_{jk}, t_{ik}) représentant deux opérations utilisant la même machine.

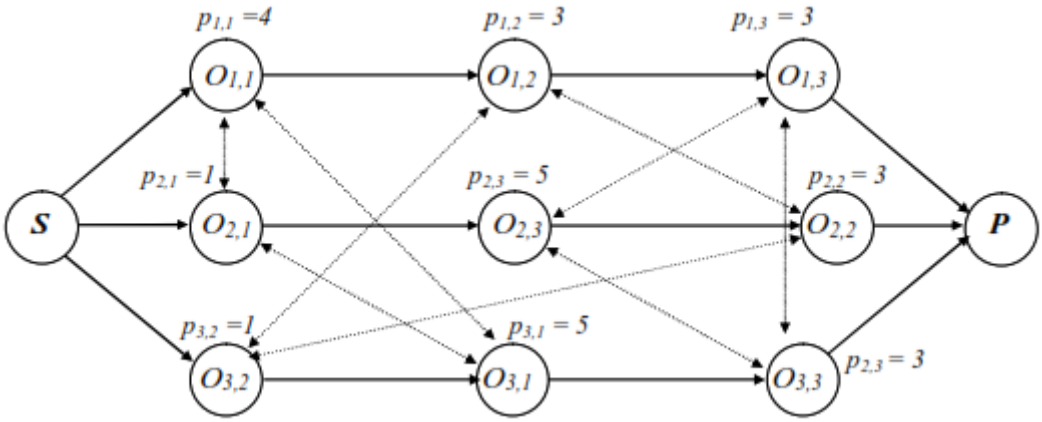

Figure 14: *Le graphe disjonctif du problème de Job Shop (3x3)*

L'ensemble des paires de disjonction associées à une même machine forment une clique de disjonction.

Le graphe disjonctif sous cette forme, modélise un problème d'ordonnancement. Pour construire un ordonnancement admissible sur ce graphe, il suffit de choisir pour chaque paire d'arcs, l'arc qui déterminera l'ordre de passage des deux opérations sur la machine ; c'est à dire *arbitrer* chacune des disjonctions. L'arbitrage doit être complet (toutes les disjonctions sont arbitrées), et compatible (le graphe est sans circuits) [18] [22].

Pour un ordonnancement donné, certaines notions bien évidentes sur le graphe disjonctif sont intéressantes à évoquer :

- ***L'opération critique :***

  Soit un ordonnancement actif, une opération est dite *critique*, si elle provoque nécessairement l'augmentation du makespan de l'ordonnancement, lorsqu'elle est retardée (alors que l'ordre d'exécution des opérations défini par l'ordonnancement ne change pas) [23].

- ***Le chemin critique :***

  Le chemin critique est une suite d'opérations critiques liées par des relations de précédence. La longueur d'un chemin critique est égale à la somme des durées des opérations qui le composent. Pour un ordonnancement donné, il peut exister plus d'un chemin critique.

- ***Le bloc critique*** *:*

Le bloc critique est une succession d'opérations critiques s'exécutant sur la même machine [22]. Tout chemin critique peut se décomposer en *b* blocs critiques. Pour un problème *n*-tâches *m*-machines, *b* est compris entre 1 et *n* : $1 \leq b \leq n$.

3.2. Représentation de l'ordonnancement par le diagramme de Gantt

La représentation la plus courante pour l'ordonnancement est le *diagramme de Gantt*. Celui- ci représente une opération par un segment ou une barre horizontale, dont la longueur est proportionnelle à sa durée opératoire.

Donc, sur ce diagramme sont indiqués, selon une échelle temporelle : l'occupation des machines par les différentes tâches, les temps morts et les éventuelles indisponibilités des machines dues aux changements entre produits [24].

Deux types de diagramme de Gantt sont utilisés : Gantt ressources et Gantt tâches (voir la figure 15) [12] :

- Le diagramme de Gantt ressources, est composé d'une ligne horizontale pour chaque ressource (machine). Sur cette ligne, sont visualisées les périodes d'exécution des différentes opérations en séquence et les périodes de l'oisiveté de la ressource.
- Le diagramme de Gantt tâche, permet de visualiser les séquences des opérations des tâches, en représentant chaque tâche par une ligne sur laquelle sont visibles, les périodes d'exécution des opérations et les périodes où la tâche est en attente des ressources.

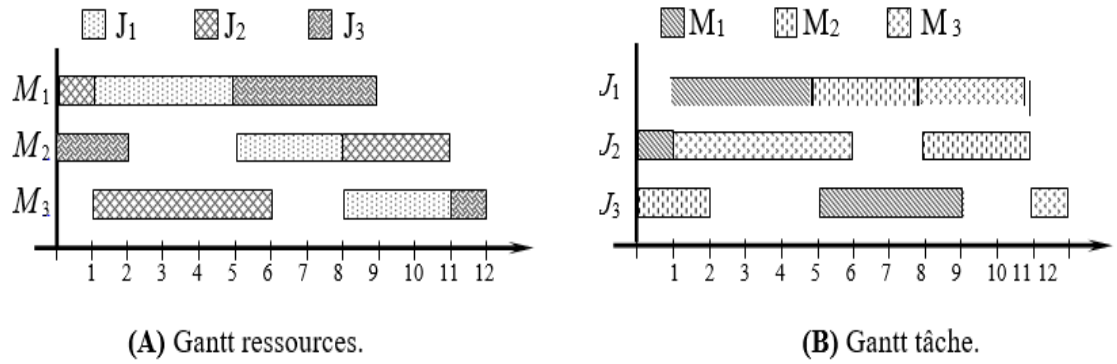

Figure 15: *Diagrammes de Gantt (problème de Job Shop du 3x3.*

4. Complexité du problème de Job Shop

La théorie de complexité permet de classer les problèmes en deux classes P et NP. La classe P regroupe les problèmes résolubles par des algorithmes polynomiaux. Un algorithme est dit polynomial, lorsque son temps d'exécution est borné par $O(p(x))$ où p est un polynôme et x est la longueur d'entrée d'une instance du problème [19]. Les algorithmes dont la complexité ne peut pas être bornée polynomialement sont qualifiés d'exponentiels et correspondent à la classe NP [19]. Les problèmes NP-complets ou NP-difficiles, représentent une large classe parmi la classe NP.

4.1. L'importance de l'espace de solutions

Le problème de Job Shop est classé par la théorie de complexité parmi les problèmes NP-difficiles au sens fort [25]. Afin de donner une idée sur l'importance de l'espace de solutions pour une instance du problème, rappelons-nous qu'il existe $(n!)^m$ différentes solutions pour un problème de n tâches à réaliser sur m machines. Ainsi, pour un problème de taille de 10×10, il existe 39594×10^{65} solutions différentes (par comparaison, l'âge de la Terre ne dépasse pas 10^{18} secondes). Sachant que ce nombre ne comptabilise pas les ordonnancements semi-actifs. Enumérer toutes ces possibilités pour arriver à la solution optimale n'est pas une chose réaliste.

Il est à noter que ce nombre évolue plus vite en fonction de J qu'en fonction de M. Autrement dit, l'ajout d'une tâche a beaucoup plus d'impact que l'ajout d'une machine. Par exemple :

- Pour un problème de 4 tâches et 5 machines (4×5) : $(n!)^m = 7962624$;
- Alors que, pour un problème de 5 tâches et 4 machines (5×4) : $(n!)^m = 207360000$.

La difficulté du Job Shop est fonction du nombre de tâches, du nombre de machines, du nombre d'opérations par tâche, et de la durée des opérations.

Le tableau 2.1, suivant [23] résume les cas où le problème peut être résolu en temps polynomial et les cas où il devient NP.

- n : nombre de tâches,
- m: nombre de machine,
- n_j : nombre d'opération par tâche,
- d(o) : durée de l'opération.

<table>
<tr><td>P</td><td colspan="2">NP</td></tr>
<tr><td>$m = 2, \forall j\ n_j \leq 2$</td><td>$m = 2, \forall j\ n_j \leq 3\ [\exists\ k, n_k = 3]$</td><td rowspan="2">$M = 2$
autre
cas</td></tr>
<tr><td>$m = 2, \forall o\ d(o) = 1$</td><td>$m = 2, \forall o\ d(o) \leq 2\ [\exists\ k, d(k) = 3]$</td></tr>
<tr><td rowspan="2"></td><td>$m = 3, \forall j\ n_j \leq 2$</td><td rowspan="2">$m \geq 2$</td></tr>
<tr><td>$m = 3, \forall o\ d(o) = 1$</td></tr>
<tr><td>$n = 2$</td><td colspan="2">$n \geq 3$</td></tr>
<tr><td>$C_{max} \leq 3$</td><td colspan="2">$C_{max} \geq 3$</td></tr>
</table>

Tableau 2: *Complexité du Job Shop [23].*

4.2. Difficulté relative des instances

La taille des instances que l'on traite est très importante dans la détermination de leur difficulté. Une classification des problèmes en fonction du : nombre de job, nombre d'opérations par job et nombre de machines est proposé par B. Penz [18].

Cette classification, même moins rigoureuse [18], permettra de repérer facilement la taille d'un problème donné, en déterminant trois classes :

- Problèmes de petite taille : Problèmes de moins de 20 tâches, moins de 10 machines et moins de 10 opérations par tâche.
- Problèmes de taille moyenne : Problèmes entre 20 et 50 tâches, de 5 à 15 machines et de 5 à 15 opérations par tâche.
- Problèmes de grande taille : Problèmes de plus de 50 tâches, de plus de 5 machines et de plus 5 opérations par machine.

La taille n'est pas le seul déterminant de la difficulté, en effet d'autres paramètres influent sur cette difficulté des problèmes. Ainsi, il est démontré que le rapport $\frac{J}{M}$ détermine bien la difficulté de l'instance. Ce rapport interprète la conjecture de Taillard, dont l'énoncé est le suivant [23] :

« Les instances telle que $n \approx m$ sont plus difficiles à résoudre que celles pour lesquelles $n >> m$; et la frontière entre les instances *faciles* et les instances *difficiles* se situe aux environs de $n = 5m$ ».

5. Méthodes de résolution du problème de Job Shop

La théorie de complexité permet de classer les problèmes en deux classes P et NP. La classe P regroupe les problèmes résolubles par des algorithmes polynomiaux. Un algorithme est dit polynomial, lorsque son temps d'exécution est borné par O (p(x)) où p est un polynôme et x est la longueur d'entrée d'une instance du problème [19]. Les algorithmes dont la complexité ne peut pas être bornée polynomiale- ment sont qualifiés d'exponentiels et correspondent la classe NP (Non -déterministe Polynomial) [19]. Les problèmes NP-complets ou NP-difficiles, représentent une large classe parmi la classe NP. Le problème de job shop appartient à cette classe. Pour un temps de calcul donné, il est possible de résoudre optimalement des problèmes de petite taille, alors qu'il devient difficile de trouver une bonne solution admissible pour un grand problème.

5.1. Les différentes méthodes de résolution

Une présentation de la totalité des méthodes et de leurs variantes apparaît une tâche difficile au vu du nombre d'approches proposées. Nous présentons sur le tableau 2.1 les grandes orientations des méthodes, en essayant de repérer les travaux de recherche essentiels. Ce tableau est synthétisé à partir des travaux de Duvivier [23] [21] [26] [18]. Pour les Méta-heuristiques, nous essayons de circonscrire avec plus de détails - sans prétendre toutefois être exhaustif - l'important des travaux effectués sur ces méthodes dans un paragraphe à part.

Méthodes de résolution			**Principaux auteurs**
Exactes	Méthodes pour les problèmes polynomiaux	Le problème à deux machines	Johnson (54), Akers (56), Jackson (56), Szwarc (60).
		Le problème à deux tâches	Akers (56), Brucker (88, 94), Brucker & Jurisch (93) Kravchenko & Sotskov (96), Kubiak & Timkovsky (96), Timkovsky (97), Williamson & al. (97), Brucker & al. (97).
	La programmation mathématique		Bowman (59), Wagner (59), Manne (60), Balas (65), Dyer & Wolsey (90), Van Den Akker (94).
	Le "branch and bound"		Cook (71), Lageweg & al. (77) Pinson (88, 95).

Approximatives	Les méthodes de relaxation		Fisher (73, 76) Fisher & al. (75), Van De Velde (91), Hoitomt & al. (93), Della Croce & al. (93) Hoogeveen & Van De Velde (95).
	Les méthodes de décomposition		Ashour (67) Kanzedal (83), Chu et al. (92) Krüger et al. (95).
	Les heuristiques constructives	L'approche par opérations	Rowe & Jackson (56), Giffler & Thompson (60), Fisher & Thompson (63), Crowston et al. (63), Jeremiah & al. (64), Gere (66), Moore (68) Panwalkar & Iskander (77), Blackstone & al. (82) Viviers (83), Lawrence (84), Haupt (89), Chang & al. (96), Sabuncuoglu & Bayiz (97).
		L'approche par machines (Shifting Bottleneck)	Adams & al. (88), Applegate & Cook (91), Dauzère-Pérès & Lasserre (93), Balas & al. (95), Dauzère-Pérès (95), Holtsclaw & Uzsoy (96), Demirkol & al. (97), Balas & Vazacopoulos (98).
		L'approche par tâches	Penz (94),
	Les Métaheuristiques	Algorithmes Génétiques	Davis (85), Falkenauer & Bouffouix (91), Nakano & Yamada (91, 92), Tamaki & Nishikawa (92), Davidor & al. (93), Fang & al. (93), Mattfeld & al. (94), Della Croce & al. (95), Kobayashi & al. (95), Norman and Bean (97), Bierwirth (95), Bierwirth & al. (96), Cheng & al. (96), Shi (97).
		Recherche Locale Génétique	Aarts & al. (91, 94), Pesch (93), Della Croce & al. (94), Dorndorf & Pesch (95), Mattfeld (96), Yamada & Nakano (95, 96).
		Recuit Simulé	Matsuo & al. (88), Van Laarhooven & al. (88, 92), Aarts & al. (91, 94), Yamada & al. (94), Sadeh & Nakakuki (96), Yamada & Nakano (95, 96), Sadeh & al. (97), Kolonko (98), Aarts & al. (91, 94), Storer & al. (92), Aarts & al. (91, 94).
Approximatives (suite)	Les Métaheuristiques	Recherche Tabou	Taillard (89, 94), DellAmico & Trubian (93), Hara (95), Barnes & Chambers (95), Sun & al. (95), Nowicki & Smutnicki (96), Ten Eikelder & al. (97), Thomsen (97), Jain & Meeran (98, 98), Jain & al. (98).

Approximatives (suite)	Autres méthodes	Satisfaction de Contraintes	Erschler & al. (76), Fox & Sycara (90) Fox & Sadeh (90), Caseau & Laburthe (94,95), Nuijten & Aarts (96), Harvey & Ginsberg (95), Sadeh & al. (95), Baptiste & Le Pape (95), Baptiste & al. (95), Pesch & Tetzlaff (96), Sadeh & Fox (96).
		Réseaux de neurones	Foo & Takefuji (88), Zhou & al. (90, 91), Van Hulle (91), Hanada & Ohnishi (93), Chang & Nam (93), Lo & Bavarian (93), Gang & Shuchun (94) Willems & Rooda (94), Satake & al. (94), Foo & al. (94, 95), Sabuncuoglu & Gurgun (96), Dagli & al. (91), Watanabe & al. (93), Cedimoglu (93), Sim & al. (94), Kim & al. (95).
		Systèmes experts	Alexander (87), Kusiak & Chen (88), Biegel & Wink (89), Charalambous & Hindi (91), Shakhlevich & al. (96), Sotskov (96).

Tableau 3: *Principales méthodes de résolution du problème de Job Shop [7].*

5.2. Les Métaheuristiques

Dès l'avènement des Métaheuristiques comme approche de résolution importante des problèmes difficiles offrant un bon compromis entre le coût de résolution et la qualité des solutions, le problème d'ordonnancement de Job Shop constitue l'un des domaines d'application de ces méthodes intensivement investigué. Bien qu'il existe une multitude de Métaheuristiques, trois ont en trouvé une place importante : Les Algorithmes Génétiques, Le Recuit Simulé et la Recherche Tabou.

5.2.1. Les Algorithmes Génétiques

Les Algorithmes Génétiques AG, sont largement appliqués aux problèmes d'ordonnancement de Job Shop. Leur application requiert les adapter au problème, en définissant adéquatement la représentation et les opérateurs.

Plusieurs approches sont proposées pour représenter le problème de Job Shop avec cette méthode. Cheng & al. (1996) [26]en recensent neuf qui sont:

- *Operation based representation* : représentation basée sur l'opération.
- *Job based representation* : représentation basée sur la tâche.
- *Job pair relation based representation* : représentation basée sur relations de paire de tâches.

- *Completion time based representation* : représentation basée sur le temps d'achèvement.
- *Random keys representation* : représentation basée sur des clés aléatoires.
- *Preference list based representation* : représentation basée sur des listes de préférences.
- *Priority rule based representation* : représentation basée sur les règles de priorité.
- *Disjunctive graph based representation* : représentation basée sur le graphe disjonctif.
- *Machine based representation* : représentation basée sur la machine.

Les cinq premières approches sont des représentations directes du problème, tandis que les autres sont indirectes, leur implantation nécessite la mise en place de décodeur.

Parmi les premiers travaux exploitant les Algorithmes Génétiques pour résoudre ce problème, figure celui de Davis (1985) [27] [28], qui emploie une technique de codage basée sur des listes d'opérations en ordre préféré pour chaque machine. Cette stratégie fut étendue par Falkenauer & Bouffoui [29], en employant des listes de préférence de tâches. Della Croce & al. [30] Ont adopté aussi cette stratégie. L'un des travaux les plus important se basant sur la même représentation est réalisé par Kobayashi & al. [31] Où un chromosome est une chaîne de symboles de longueur *n* et chaque symbole identifie une opération qui s'exécutera sur une machine donnée. Ce schéma exige un croisement spécifique comme SXX développé par Kobayashi & al [31], ou LOX de Falkenauer & al [32], ou encore JOX développé par Yamamura & al [33].

Yamada & Nakano (1991), utilisent un codage binaire basé sur les relations de précédence entre les opérations s'exécutant sur la même machine. Le codage dans ce cas est connu par "codage basé sur le temps d'achèvement des opérations" [34].

Tamaki & Nishikawa utilisent une représentation indirecte basée sur le graphe disjonctif. Un chromosome consiste en une chaîne binaire correspondant à une liste ordonnée de préférences de nœuds relatives au graphe disjonctif [26].

Un autre codage proposé par Holsapple & al [35] Repose sur des listes de tâches : l'ordonnancement porte sur toutes les opérations d'une tâche donnée avant de passer à la suivante.

Dans la représentation basée sur les règles de priorité, utilisée par Dorndorf & al [36], le chromosome est une suite de gènes désignant chacun une règle de priorité. Les auteurs utilisent

un constructeur d'ordonnancement basé sur l'algorithme de Giffle & Thompson. Ces règles sont très nombreuses, Sauer en décompte plus d'une centaine [37].

Bierwirth [38] crée un AG dont les chromosomes représentent des permutations de tâches. Dans un travail similaire, Bierwirth & al [39] Analysent trois opérateurs de croisement, qui préservent respectivement l'ordre : relative, de position et absolue des permutations d'opérations. Un schéma proche est employé par Fang & al. L'avantage de ce schéma est qu'il n'exige qu'un simple constructeur ; de plus, plusieurs croisements peuvent s'appliquer : GOX proposé par Oliver & al [37], PPX proposé par Bierwirth & al [39] [40].

Norman & Bean proposent l'approche de clefs aléatoires, chaque gène (clé aléatoire) du chromosome consiste en deux parties : un entier de {1, 2..., *m*} représentant la machine allouée, et une fraction générée aléatoirement de {0,1} dans un ordre croissant, détermine l'ordre des opérations sur chaque machine [26]

Shi a appliqué une technique de croisement qui divise arbitrairement le potentiel génétique en deux parties, à partir desquelles sont générés les descendants [26].

5.2.2. La Recherche Tabou

La Recherche Tabou (RT) issue des travaux de Glover, est une variante du paradigme de la recherche locale. La Recherche Tabou est une sorte de procédure de recherche déterministe, orientée en s'aidant d'une mémoire qui garde l'historique des solutions, et qui interdit le retour à des solutions récemment visitées. La fonction de voisinage est la composante primordiale de la Recherche Tabou qui détermine amplement l'efficacité de recherche.

Une contribution remarquable en fonction de voisinage est provenue de Van Laarhoven & al [41], qui consiste à inverser l'ordre d'exécution sur la même machine de deux opérations critiques adjacentes. De plus, ils démontrent la propriété qu'il existe forcément une séquence de mouvements qui mènent à une solution optimale.

Laguna & al (1991, 1993) présentent un des travaux précoces concernant l'application de cette méthode à l'ordonnancement des Job Shop. Ils ont construit trois stratégies de Recherche Tabou reposant sur des mouvements simples. Barnes & Laguna, suggèrent six composantes qui assurent la fiabilité de la méthode aux problèmes d'ordonnancement [26]. Dans leur méthode de Recherche Tabou, Barnes & Chambers (1995) utilisent la liste des meilleures solutions trouvées comme point de redémarrage de la recherche [26].

Dans sa notable contribution, Taillard [42]incorpore une technique qui accélère la recherche en ne mettant à jour que les dates de début de certaines opérations. Cette technique ne permettant pas le calcul exact du makespan des mouvements est considérée comme une méthode d'estimation rapide mais relative.

Dell'Amico & Trubian [43] incorporent à leur algorithme de RT un nouveau générateur de solution initiale. De plus, deux voisinages sont formulés : le premier considère le reversement de jusqu'à trois arcs impliquant : *i*, *MP*[*i*] et *MP*[*MP*[*i*]] (resp. *i*, *MS*[*i*] et *MS*[*MS*[*i*]]) (où *MP*[*i*] et *MS*[*i*] dénote resp. L'opération qui précède (suit) *i* sur la même machine). Si le reversement de l'un de ces arcs est tabou, alors le mouvement entier est interdit. Le deuxième voisinage repose sur le block critique. En vue d'accélérer la recherche, les auteurs emploient l'estimation précédente de Taillard.

Les meilleures résultats fournis par une simple Recherche Tabou sont obtenus par Nowicki & Smutnicki (1996), qui ont appliqué un voisinage plus restreint visant à inverser les deux premières ou les deux dernières opérations de chaque bloc critique non dernier (resp. Non premier) [44] [45] .

Thomsen 1997 a pu améliorer la résolution de plusieurs benchmarks difficiles par l'hybridation de la RT et la méthode par séparation et évaluation. Citons aussi parmi plusieurs autres contributions à la résolution du problème par la Recherche Tabou : Hurink & al. (1994), Brucker & Kr•mer (1995), Dauzère-Pérès & Paulli (1997), Brucker & Neyer (1998), Baar & al. (1997) [26].

5.2.3. Le Recuit Simulé

Le Recuit Simulé RS est une autre Métaheuristique basée sur la recherche locale. Il est proposé par Kirkpatrick & al. (1983) et Cerny (1985), mais ces origines remontent aux travaux de Metropolis & al. (1953). Son application au problème d'ordonnancement de Job Shop marque plusieurs travaux :

En se basant sur leur première fonction de voisinage, Van Laarhooven & al(1992) [41] Construisent un Recuit Simulé générique ne demandant pas une vue profonde sur la structure du problème.

Un autre voisinage important est défini par Matsuo & al (1988), qui ont prouvé que le reversement de deux opérations critiques ne conduit plus à une amélioration immédiate du

makespan, si l'opération qui les précède et celle qui les suit sont également critiques, ce voisinage plus restreint, est incorporé à un Recuit Simulé et utilisé par ces auteurs [46].

Le majeur inconvénient rencontré lors de l'application de la méthode de Recuit Simulé, est le temps d'exécution : à titre d'exemple certaines instances de *swv* (voir figure 2.1) exigent plus de 375 millions itérations, qui se déroulent en plus de 44 heures sur les ordinateurs de l'époque. A cause de ces exigences, Johnson & al. (1989) proposent assister la méthode par une table de recherche "table lookup" comportant des approximations de la probabilité d'acceptation. Szu & Hartely (1987) proposent à leur tour un Recuit Simulé Rapide RSR, qui permet occasionnellement des grands sauts afin d'accélérer la [26].

Yamada & al (1994) ont formulé une méthode appelée "Block Critique Recuit Simulé" utilisant une fonction de voisinage dérivée du block critique. La température initiale et finale sont définies en fonction d'un taux d'acceptation, et des ré-intensifications sont appliquées pour améliorer la recherche. Les auteurs [47] ré-implémentent leur BCRS avec un générateur d'ordonnancements de Giffler & Thompson. Krishna & al. (Kris et al. 1995). Un autre algorithme de Recuit Simulé connu par *Focused Simulated Annealing* FSA, procédant au gonflement du coût des solutions inefficaces, est proposé par Sadeh & Nakakuki [48].

Kolonko [49] de son part, a montré que le Recuit Simulé appliqué au Job Shop n'est plus un processus convergent, et le voisinage standard du problème n'est pas symétrique. Par conséquent, il a présenté une méthode hybride combinant le Recuit Simulé et un Algorithme Génétique.

6. Les heuristiques

Les heuristiques sont des méthodes empiriques basées sur des règles simplifiées pour optimiser un ou plusieurs critères. Le principe général de ces méthodes est d'intégrer des stratégies de décision pour construire une solution proche de l'optimum, tout en essayant de l'obtenir en un temps de calcul raisonnable [14] . On distingue :

- FIFO (First In First Out) : la première tâche qui vient est la première tâche ordonnancée.

- SPT (Shortest Processing Time) : la tâche ayant le temps opératoire le plus court est traitée en premier lieu.

- LPT (Longest Processing Time) : la tâche ayant le temps opératoire le plus important est ordonnancée en premier lieu.

- EDD (Earliest Due Date) : la tâche ayant la date due la plus petite est la plus prioritaire.

- SRPT (Shortest Remaining Processing Time) : cette règle, servant à lancer la tâche ayant la plus courte durée de travail restant à exécuter, est très utilisée pour minimiser les encours et dans le cas des problèmes d'ordonnancement préemptifs.

- ST (Slack Time) : à chaque point de décision, l'opération ayant la plus petite marge temporelle est prioritaire. Faute de disponibilité des ressources de production, cette marge peut devenir négative.

6.1. Shifting Bottleneck

Le shifting bottleneck, décrit dans (Adams et al 1988) [50], est une heuristique très connue et efficace pour le problème de job shop avec comme objectif le makespan.

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

7. Conclusion

L'ordonnancement d'ateliers de type Job Shop est considéré comme un problème NP-Difficile. Malgré sa modélisation simple et sa définition précise des données (tâches et machines), des contraintes et des objectifs, la résolution du problème est une tâche difficile, la classant comme l'un des problèmes NP- Difficiles dans un sens fort. Cela en fait un terrain d'essai parfait pour la mise à l'essai d'une multitude de méthodes, qui compris de nouvelles approches de résolution, telles que les heuristiques, qui sont le sujet de notre application.

CHAPITRE III

Utilisation de l'heuristique Shifting bottleneck

CHAPITRE III

1. Introduction

L'objectif de notre travail est d'étudier et améliorer une des méthodes heuristiques dédiées aux problèmes d'ordonnancement de Job Shop c'est l'heuristique shifting bottleneck. De ce fait, ce chapitre est consacré à la présentation de notre travail effectué qui consiste premièrement à résoudre un problème d'ordonnancement de type job shop dans sa version classique (sans autres contraintes) en utilisant et l'heuristiques shifting bottleneck et deuxièmement à proposer des améliorations possibles à cette heuristique. Les résultats obtenus après la simulation des deux approches sur plusieurs exemples sont ensuite comparés avec des résultats obtenus en utilisant une méthode exacte.

2. Problème à étudier

Le problème considéré dans ce travail est un problème de job shop, qui consiste à ordonnancer un ensemble de *n* jobs J= {1 2 3…n} à être exécutés sans interruption sur m machines M= {1 2 3 ...m}. Les jobs sont présents à l'instant initiale. Chaque job contient une ou plusieurs opérations qui sont rangées selon des gammes bien définies. L'objectif est de minimiser le temps total d'exécution. Selon la notation de Graham [15] ce problème peut être notée comme suit : **J/ /C_{max}.**

Dans ce qui suit nous allons donner les notations des données utilisées dans le cadre de notre problème :

- ✓ **M** = {M1, M2…Mm} : Ensemble de toute les machines disponibles
- ✓ **n**: Nombre totale de jobs.
- ✓ **m** :Nombre totale de machines.
- ✓ i : indice du i éme job.
- ✓ j : Indice de la j éme opération du job j i
- ✓ O_{ij} :j éme Opération du job j i
- ✓ P_{ij} : Durée d'exécution de l'opération Oij.
- ✓ C_{ij} : Date de fin de l'opération Oij.

- ✓ r_{ij}: date de disponibilité (release date) du job J.
- ✓ $\mathbf{d_{ij}}$: date de fin souhaitée (due date) du job J.
- ✓ $\mathbf{C_{max}}$=max {Ci, i = 1…n} : Makespan ou date de fin de tous les jobs

3. L'heuristique Shifting Bottleneck

L'heuristique Shifting Bottleneck est une procédure souvent utilisée pour minimiser le makespan dans un atelier de type job shop pour lequel les ressources de traitement sont critiques. Cette heuristique est destinée à minimiser l'effet de goulet d'étranglement (bottleneck) en ordonnançant d'abord les ressources qui causent potentiellement le plus de retard.

3.1. État de l'art

L'heuristique originale du shifting bottleneck a été développé pour le problème (J_m// C_{max}) [50]. Plus tard, L'heuristique du shifting bottleneck d'Adams et al. (1988) a été étendu à d'autres mesures comme le total weighted tardiness (W_jT_j) [51]. Et le retard maximal (L_{max}) (Demirkol et al., 1997), (Demirkol et Uzsoy 2000), (Ovacik et Uzsoy 1997) [52] , [53], [54].L'heuristique appliqué aux job shop avec mesure de performance Lmax est décrite par Demirkol et al [52], où différents types de procédures sophistiquées de solution secondaire sont étudiés. Il s'avère que l'utilisation d'approches exactes comme branch and bound conduit à une amélioration significative par rapport aux procédures de solution de sous-système basées sur la répartition. Ovacik et Uzsoy ont rapporté dans [54] que l'heuristique de shifting bottleneck appliquée à la zone de test des installations de fabrication de plaquettes de semi-conducteurs conduit à de meilleurs résultats par rapport aux approches de dispatching pures uniquement si la recherche de voisinage est incluse.

Mason et al [55] présentent une shifting bottleneck modifié pour les ateliers complexes du job shop. Une méthode de simulation permettant d'évaluer les performances de l'heuristique dans un environnement dynamique est développé par Mohnch et al [56] .Les résultats présentés in [57] montrent que le shifting bottleneck modifié avec des procédures de solution de sous-problème de dispatching surpasse clairement les approches de dispatching pures pour les ateliers job shop complexes dans un environnement dynamique.

Brandimarte(2000) [58] Décrivent une méthode conceptuelle pour résoudre les problèmes d'ordonnancement par décomposition en sous-problèmes qui sont plus importants que les groupes d'outils dans l'approche initiale du shifting bottleneck.

3.2. Procédure de shifting bottleneck (SB)

La procédure standard commence avec deux ensembles : l'ensemble des ressources de traitement qui sont déjà séquencées M_0 (initialement cet ensemble est vide), et l'ensemble total des ressources de traitement M. Un graphe disjonctif du problème associé est construit. Plusieurs itérations vont être effectuées pour trouver le meilleur séquencèrent des jobs dans chaque machine. A chaque itération, une ressource est choisie et ajoutée dans M_0. Les opérations sur cette ressource sont séquencées et le graphe est mis à jour en intégrant l'arbitrage des disjonctions correspondantes sous forme d'arcs conjonctifs.

En effet la procédure de cette méthode est déroulée en quatre étapes principales:

- **Etape 01 :** Modélisation par un graphe disjonctif.

  Dans cette étape nous modélisons le problème par un graphe disjonctif, où chaque ligne représente un job et les nœuds (i, j) représentent les opérations correspondantes à chaque job.

  Pour bien expliquer on donne un exemple de problème d'ordonnancement $j//C_{max}$ avec 3 machines et 3 jobs suivant :

	M_1		**M_2**		**M_3**	
	P	O	P	O	P	O
J1	7	1	10	3	8	2
J2	4	2	6	1	12	3
J3	8	1	8	2	7	3

Tableau 4: *Exemple de problème d'ordonnancement avec 3 machines et 3 jobs*

La modélisons de ce problème par le graphe suivant :

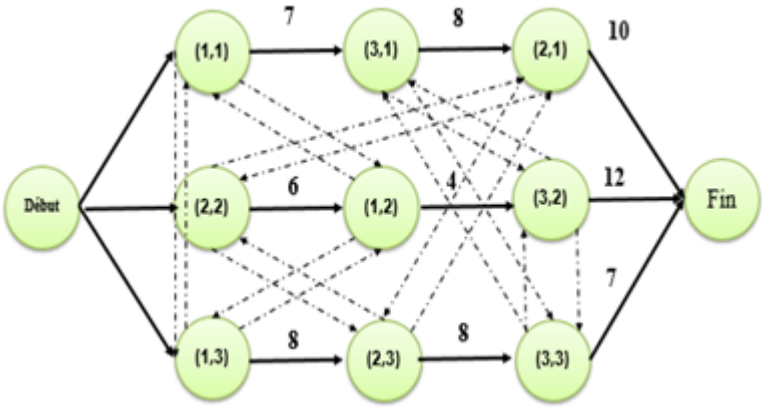

Figure 16: *Modélisation par un graphe disjonctif*

- **Etape 02:** déterminer la ressource qui cause le plus de blocage (Machine goulot).

Dans cette étape on va sélectionner la machine avec le plus grand temps de traitement total d'opération comme machine goulot. C'est-à-dire calculer la somme de processing time $\sum_j \boldsymbol{P_{ij}}$ de tous les jobs qui passe par la machine.

Dans l'exemple précédent pour trouver la première machine goulot il faut calculer $\sum_j \boldsymbol{P_{ij}}$

- Pour M_1 ..... $\sum_j \boldsymbol{P_1}$ = 7+4+8 = **19**
- Pour M_2 ..... $\sum_j \boldsymbol{P_2}$ = 10+6+8 = **24**
- Pour M_3 ..... $\sum_j \boldsymbol{P_3}$ = 8+12+7= **27**

Donc la première machine goulot est la machine $\boldsymbol{M_3}$ car elle possède le plus grand temps d'exécution **27** et le séquencement des machines goulots selon l'ordre **M_3-M_2-M_1.**

- **Etape 03** : résoudre le problème 1/rj/L_{max} et trouver le meilleur séquencement

Dans cette étape en premier temps il faut calculer la date de disponibilité rj (release date) et date de fin souhaitée dj (due date) qui sont déterminées par le plus long chemin depuis le nœud de départ jusqu'au nœud opératoire associé et pour la résolution de problème 1/rj/L_{max} on utilise une procédure de Branch and Bound pour trouver le plus petit retard L .

Pour mieux comprendre et détailler on va revenir à l'exemple précédent les dates de disponibilité rj des 3 jobs sont 7 pour J_1 (car il doit s'exécuter sur la machine M_1 avant de passer à la machine M_3), 10 pour J_2 (car il va passer par la machine M_2 et M_1 puis M_3) et 16 pour J_3 (car il doit s'exécuter sur M_1 et M_2 puis terminer par M_3).

Les dates de fin sont calculées à partir du C_{max} correspondant au plus long chemin

dans ce graphe dans notre exemple le C_{max} est égal 25, donc les date de fin dj= (C_{max} - la longueur de chemin du nœud considérée jusqu'à la fin + le temps de traitement de l'opération considérée). Alors dans l'exemple seront respectivement 15, 25 et 25 pour les jobs J_1, J_2et J_3.

	J_1	J_2	J_3
p_j	8	12	7
r_j	7	10	16
d_j	15	25	25

Tableau 5: *Temps opératoires, la date de disponibilité et date de fin pour J1, J2 et J3*

On passe à la résolution de problème 1/rj/L_{max} par l'utilisation Branch and Bound dans le premier temps commence par J_1 entant que premier job de la séquence et en basant sur la règle preemptive EDD (earliest due date) pour ordonner les reste des Jobs qui donne le séquencement suivant J_1- J_2- J_3 après il faut calculer retard L_{max} en fais comme ça pour tous les jobs, Ensuite faire une comparaison entre les L_{max} et en choisi le plus bas, le Job qui donne L_{max} fixé comme le premier qui passe sur la machine (dans notre exemple J_1 c'est le premier qui passe sur M_3) et fait tous les étapes précédent sur les reste Job. Après l'application de branch and bound dans l'exemple précédent on trouver la séquence qui minimise Lmax sur la machine M_3 est : J_1- J_2- J_3.

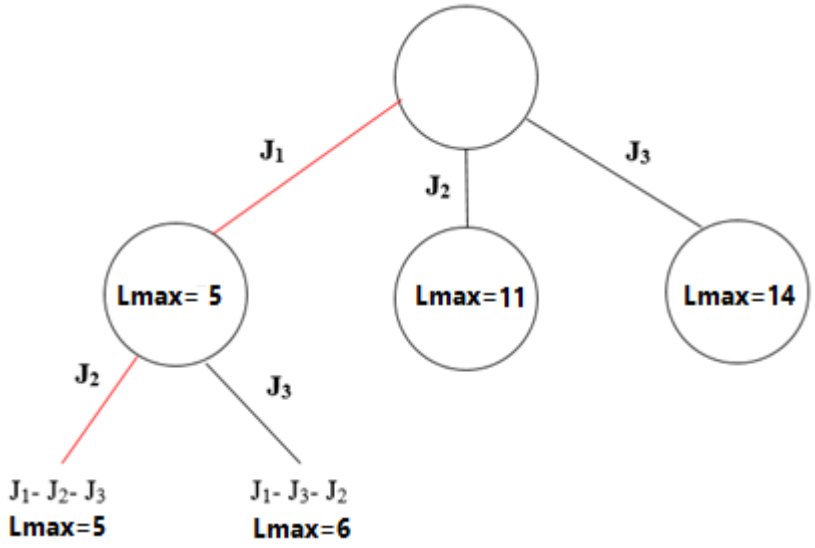

Figure 17: :*Arbre de séparation et d'évaluation*

- **Étape 04** : Mise à jour du graphe.

Après que nous avons trouvé le meilleur séquencement sur la machine goulot on va faire une mise à jour sur le graphe qui établit dans la première étape en plaçant des arcs qui correspondent l'ordre optimale sur la machine goulot.

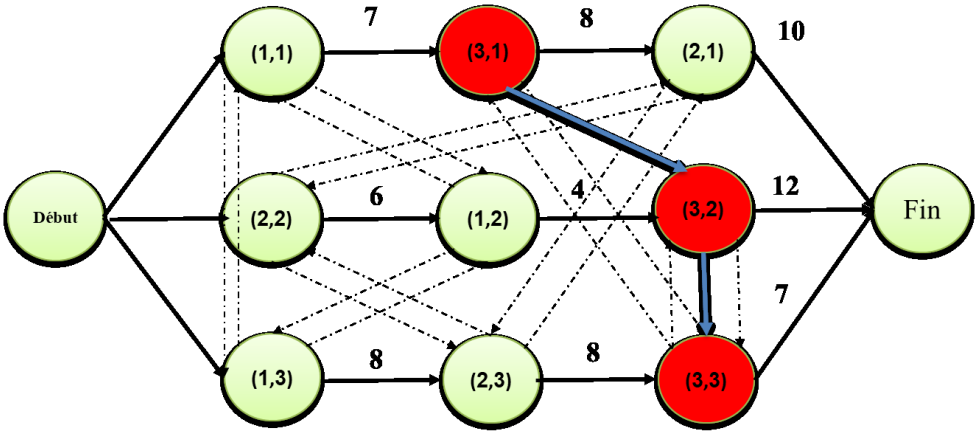

Figure 18: *Graphe disjonctif avec mise a jour.*

Les quatre étapes doivent être effectuées jusqu'à fin des machines. Pour bien comprendre on peut représenter l'heuristique par l'organigramme suivant :

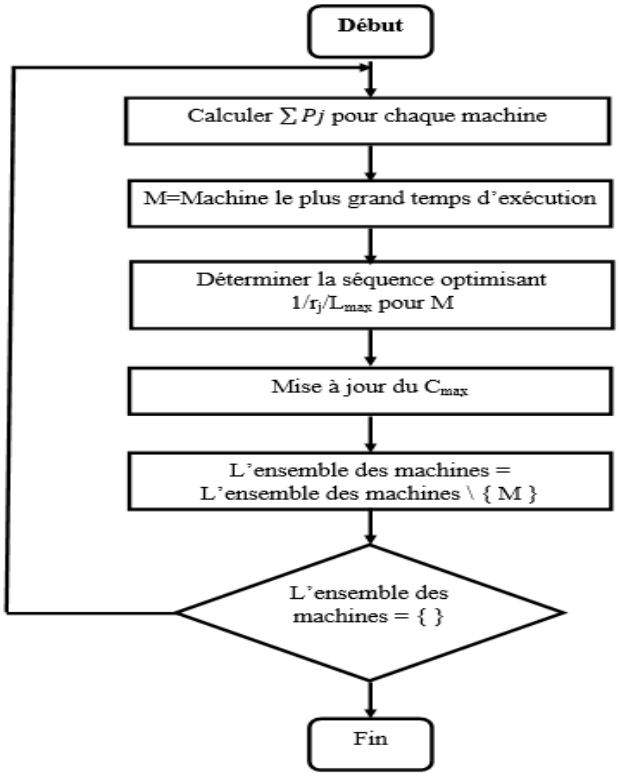

Figure 19: *Heuristique du Shifting Bottleneck*

4. Shifting modifiée bottleneck (SB EDD)

Après une étude approfondie sur l'heuristique SB présentée dans plusieurs travaux de la littérature nous avons constaté que cette méthode n'est pas adaptée à notre problème, plus précisément dans l'étape 3 dont la plupart des chercheurs font le recours à la méthode EDD préemptive pour résoudre le problème d'une machine avec Lmax (comme objectif). En effet cette méthode (EDD préemptive) n'est applicable que si la préemption des tâches est autorisée ce qui n'est pas le cas dans notre problème.

En plus, et dans la même étape, nous avons proposé d'utiliser la méthode CPM pour calculer les valeurs de release date (rj) et due date (dj) afin d'assurer la qualité des solutions trouvées. En appuyant sur ces deux points importants notre modification proposée pour améliorer et adapter SB a été alors représentée par deux contributions

- **Utilisation de la règle EDD**

La règle EDD (Earliest Due Date) consiste à séquencer les tâches dans l'ordre croissant des dates de fin au plus tard. Cette règle permet de résoudre d'une manière optimale les problèmes 1| |Lmax.

Dans notre travaille on a utilisé cette règle dans l'étape 3 pour ordonner les jobs sur la machine goulot afin de résoudre le problème $1/rj/L_{max}$.

- **Utilisation de la méthode CPM**

Notre contribution dans cette partie consiste à remplacer la méthode utilisée dans l'étape 3 de l'heuristique SB pour le calcul de deux valeurs rj et dj, par une méthode exacte utilisée dans l'ordonnancement des projets c'est la méthode du chemin critique (Critical Path Method ou CPM)

La CPM est une technique de gestion de projet qui consiste à identifier étape par étape la planification des tâches de projet et à identifier les chemins critiques et non critiques dans le but de prévenir le délai et les goulots d'étranglement existants dans ce projet.

Cette méthode est une technique de schématisation d'un ensemble d'activités (un réseau d'activités) sous forme de diagramme fléché ou les boîtes (ou nœuds) représentent les activités et les flèches représentent les relations logiques entre les activités.

La figure 20 représente un exemple d'un réseau CPM représentant la planification des tâches d'un projet.

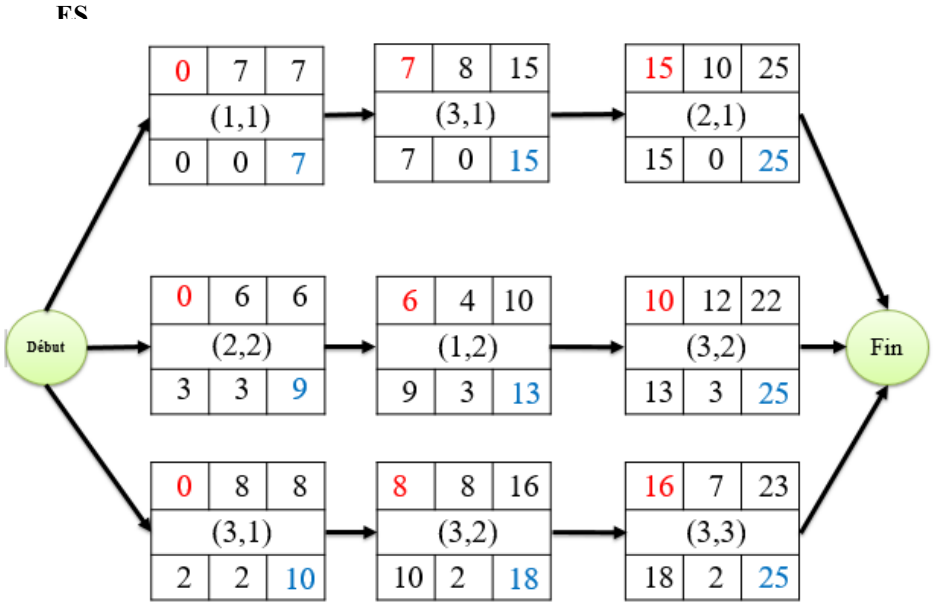

Figure 20: *Diagramme de CPM.*

En effet notre contribution consiste à changer le graph disjonctif du problème de job shop par un réseau CPM à base duquel nous allons calculer les valeurs de rj et dj. Alors dans ce diagramme la valeur de rj de chaque opération est égal à la date de début au plus tôt **E**arly **S**tart (**ES**), la date avant laquelle une activité ne peut pas être débuté. La valeur de dj correspond à la date de fin au plus tard **L**aste **F**inish (LF) c'est la date à laquelle une tâche peut terminer sans que la date de fin projet ne se décale pas.

5. Méthode exacte (Mixed integer linear programming)

Généralement il existe trois formulations mathématique normalisées de MILP pour les problèmes d'ordonnancement à savoir time indexed formulation, rank-based formulation et disjonctif formulation. D'après l'étude de Pan [59] qui fait une comparaison théorique entre ces trois modèles pour un problème job shop JSP, il a montré que le modèle disjonctif est le plus efficace puisqu'il comporte le moins de variables binaires. De ce fait nous avons choisi d'utiliser cette dernière formulation pour faire la comparaison dans notre travail.

Notre modèle disjonctif est inspéré du travail de Manne [60], dont les variables de décision et le modèle sont de dénommés comme suit :

- x_{ij} est date de début du travail j sur la machine i
- z_{ijk} est égal à 1 si la tâche j précède la tâche k sur la machine i.
- σ_h^J s'appelle la h-ième opération du travail j
- σ_m^j est la dernière opération de job j

$$\begin{aligned}
\min \quad & C_{max} && && (1)\\
\text{s.t.} \quad & x_{ij} \geq 0, && \forall j \in J, i \in M && (2)\\
& x_{\sigma_h^j,j} \geq x_{\sigma_{h-1}^j,j} + p_{\sigma_{h-1}^j,j}, && \forall j \in J, h = 2,\ldots,m && (3)\\
& x_{ij} \geq x_{ik} + p_{ik} - V \cdot z_{ijk}, && \forall j,k \in J, j < k, i \in M && (4)\\
& x_{ik} \geq x_{ij} + p_{ij} - V \cdot (1 - z_{ijk}), && \forall j,k \in J, j < k, i \in M && (5)\\
& C_{max} \geq x_{\sigma_m^j,j} + p_{\sigma_m^j,j}, && \forall j \in J && (6)\\
& z_{ijk} \in \{0,1\}, && \forall j,k \in J, i \in M && (7)
\end{aligned}$$

Figure 21 : *Le modèle disjonctif Manne [60]*

La fonction objective est énoncée en (1). La contrainte (2) garantit que la date de début de chaque Job est supérieure ou égale à 0. La contrainte (3) est la contrainte de priorité. Elle garantit que toutes les opérations d'un Job sont exécutées dans l'ordre donné. Les contraintes disjonctives (4) et (5) font en sorte qu'il n'est pas possible de programmer deux Job sur la même machine en même temps V doit être attribué à une valeur suffisamment importante pour assurer l'exactitude des points (4) et (5).

Dans ce modèle, on attribue, $V = \sum_{j\in J} \sum_{i\in M} pij$ puisque le temps de réalisation de toute opération ne peut pas dépasser la somme des temps de traitement de toutes les opérations. La contrainte (6) garantit que le makespan est la plus longue durée de réalisation de la dernière opération de tous les Job.

Nous notons que les solveurs MILP modernes permettent une modélisation directe des contraintes disjonctives avec les solveurs des contraintes spécifiques. Par exemple, les contraintes (4) et (5) peuvent être modélisées avec les contraintes « indicatrices » dans CPLEX.

6. Simulation et résultats

Pour évaluer les performances de la méthode proposée nous avons effectué plusieurs applications de cette méthode sur plusieurs exemples pour des problèmes de tailles différentes. Dans ce cadre nous avons créé un programme sous Matlab qui permet de calculer le Cmax et trouver la meilleure solution. Ces résultats sont ensuite comparés avec des résultats obtenus par les deux méthodes, la méthode Shifting Bottleneck avec énumération complète (simulée par logiciel Matlab) et la méthode exacte (MILP) développée par la programmation mathématique dont nous avons utilisé le solveur Cplex.

En effet nous avons étudié dix exemples différents pour chaque problème (comme le montre le tableau 7). Ces exemples sont générés d'une manière aléatoire à travers un sous-programme qui définit aléatoirement les gammes opératoires et le temps d'exécution des jobs dans chaque machine (ces temps sont pris entre 0 et 30).

Le tableau ci-dessous présente les résultats obtenus du Makespan (Cmax) et du temps de calcul (CPU) pour le problème 6 jobs 7 machines en utilisant les trois méthodes.

Problème 6×7	**MILP**		**SB EDD**		**SB**	
	C_{max}	**CPU (s)**	**C_{max}**	**CPU (s)**	**C_{max}**	**CPU (s)**
1	166	0,13	195	0,110814	203	0,233426
2	167	0,11	176	0,095333	180	0,241836
3	186	0,13	223	0,099785	190	0,237297
4	162	0,06	199	0,099885	176	0,239698
5	170	0,16	206	0,097482	186	0,233463
6	150	0,11	211	0,099772	202	0,23911
7	143	0,14	214	0,103018	183	0,237931
8	164	0,14	193	0,101097	173	0,230874
9	166	0,13	230	0,105637	187	0,24317
10	153	0,17	197	0,104051	178	0,245169
Moyenne	162,7	0,128	204,4	0,101687	185,8	0,238197

Tableau 6: *Exemple de résultats obtenus par simulation du problème 6 jobs et 7 machines.*

Dans les parties qui suivent nous allons présenter la comparaison que nous avons fait entre les trois méthodes, à travers différentes instances. Cette comparaison a été faite sur deux points importants; le Cmax représentant la meilleure solution trouvée et le CPU qui définit le temps de calcul consommé pour trouver cette meilleure solution.

En plus nous avons calculé le taux de rapprochement des solutions des deux heuristiques **SB EDD** et **SB** à la solution optimale obtenue par **MILP** (L'équation 03).

$$Taux = \frac{Cmax(\text{MILP})\dots}{Cmax\ (heuristique)} \times 100 \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3)$$

Partie 01

Pour cette première partie nous avons choisi de traiter huit problèmes de trois jobs. Chaque résultat montré ici (de chaque factor) représente la moyenne des résultats de dix exemples du même problème.

Problème J ×M	MILP		SB EDD			SB		
	C_{max}	CPU (s)	C_{max}	CPU (s)	Taux (%)	C_{max}	CPU (s)	taux(%)
3×3	75,1	0,027143	92,7	0,080085	81,01402	84	0,094183	89,40476
3×4	85,2	0,028889	110,1	0,08192	77,3842	99,7	0,094556	85,45637
3×5	92,9	0,041	108,1	0,087186	85,93895	99,8	0,099187	93,08617
3×6	116,6	0,026	129,4	0,088887	90,10819	121,9	0,099728	95,65217
3×7	137,9	0,022	156,8	0,091767	87,94643	148,1	0,108149	93,11276
3×8	153,9	0,025	170,1	0,094837	90,47619	165	0,105606	93,27273
3×9	166,5	0,023	184,9	0,096865	90,04867	180,8	0,110223	92,09071
3×10	181,2	0,029	201	0,101367	90,14925	198,7	0,117057	91,19275

Tableau 7: *Résultats obtenus pour les problèmes de 3 Jobs*

D'après le tableau 7 il est remarqué d'une part, que les résultats obtenus lors de l'application de l'algorithme (SB EDD) sont très proches aux solutions optimales ce qui est justifié par le taux de rapprochement (Taux) élevé. Dans l'autre part que les temps d'exécution écoulé en utilisant la méthode (SB EDD) sont meilleur par rapport aux seuls donnés par les autres méthodes (MILP et SB).

Partie 02

Pour la deuxième partie nous avons traité huit problèmes de six jobs. En basant toujours sur les mêmes factor de comparaison, nous avons varié l'ensemble des machines de 3 à 10 comme il est montré dans le tableau suivant :

Problème J ×M	**M I P**		**SB EDD**			**SB**		
	C_{max}	**CPU (s)**	**C_{max}**	**CPU (s)**	**Taux (%)**	**C_{max}**	**CPU (s)**	**taux(%)**
6×3	101	0,0925	125,5	0,082994	80,47809	115	0,144522	87,82609
6×4	134,8	0,103	160,1	0,087299	84,19738	155,5	0,170994	86,6881
6×5	129	0,153	144,7	0,091774	89,14997	146,2	0,192964	88,23529
6×6	150,8	0,134	186,4	0,092908	80,90129	172,9	0,212211	87,21805
6×7	162,7	0,128	204,4	0,101687	79,59883	185,8	0,238197	87,56728
6×8	177,9	0,171	214	0,106839	83,13084	204,5	0,266758	86,99267
6×9	197	0,162	240,3	0,097157	81,98086	231,4	0,291627	85,13397
6×10	216,9	0,24	256,3	0,127215	84,62739	251,7	0,324607	86,17402

Tableau 8 : *Résultats obtenus pour les problèmes de 6 Jobs*

Il est bien distingué à travers les résultats du tableau 8 que l'application de l'algorithme (SB EDD) a donnée des meilleures solutions en termes de temps d'exécution par rapport aux autres méthodes.

Partie 03

Dans cette partie, huit problèmes de dix jobs sont donc choisi pour le traitement. Le tableau suivant montre les résultats obtenus après l'utilisation des trois méthodes sur huit problèmes.

Problème J ×M	MILP		SB EDD			SB		
	C_{max}	CPU (s)	C_{max}	CPU (s)	Taux (%)	C_{max}	CPU (s)	taux(%)
10×3	181,2	38,709	191,8	0,084462	94,47341	///	Sup à 1 heure	///
10×4	195	36,08	218	0,087614	89,44954	///	Sup à 1 heure	///
10×5	181,4	11,521	243,7	0,096453	74,43578	///	Sup à 1 heure	///
10×6	197,8	15,253	252,5	0,104753	78,33663	///	Sup à 1 heure	///
10×7	205,4	3,573	263,8	0,115404	77,86202	///	Sup à 1 heure	///
10×8	197,8	15,253	252,5	0,104753	78,33663	///	Sup à 1 heure	///
10×9	205,4	3,573	263,8	0,115404	77,86202	///	Sup à 1 heure	///
10×10	211,1	4,03	305,3	0,130954	69,1451	///	Sup à 1 heure	///
11×11	///	Sup à 1 heure	384	0,187222	///	///	Sup à 1 heure	///
10×15	///	Sup à 1 heure	434	0,383118	///	///	Sup à 1 heure	///

Tableau 9:*Résultats obtenus pour les problèmes de 10 Jobs*

D'après la dernière simulation, nous avons constaté que les deux méthodes MILP et SB ne sont pas capables de trouver une solution quand les nombres des machines et des jobs sont très élevés en revanche, la méthode SB EDD a pu résoudre les problèmes en donnant des résultats acceptables dans un temps raisonnable (<= 40ms).

Représentation graphique des résultats du tableau :

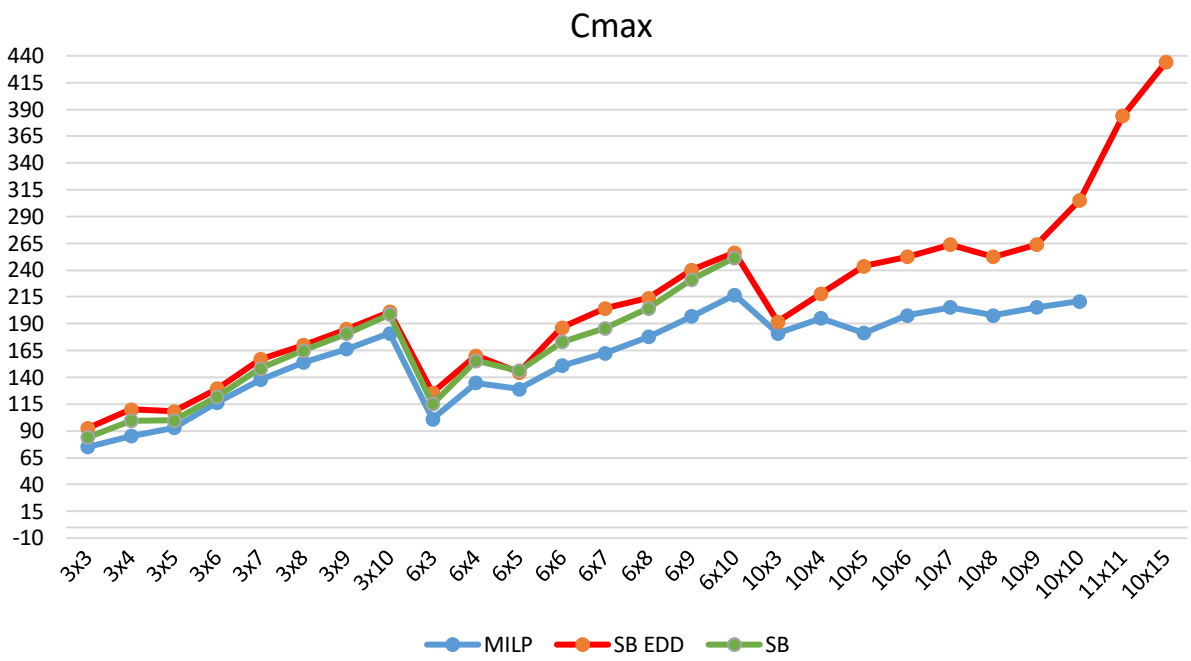

Figure 22: *Représentation graphique des résultats de Cmax*

Ce que vous voyez sur ces graphes de lignes c'est la valeur de Cmax de notre méthode (SB EDD) qui est en rouge, par rapport aux deux autres méthodes (le modèle mathématique en bleu et le SB en vert).On remarque lorsqu'on augmente le nombre job et le nombre des machines, notre méthode a des valeurs de Cmax proches de ceux de la méthode originale; Et à partir 10 job et 10 machines on remarque la méthode SB ne donne pas les résultats, et à partir 11 job et 11 machines on remarque MILP ne donne pas les résultats.

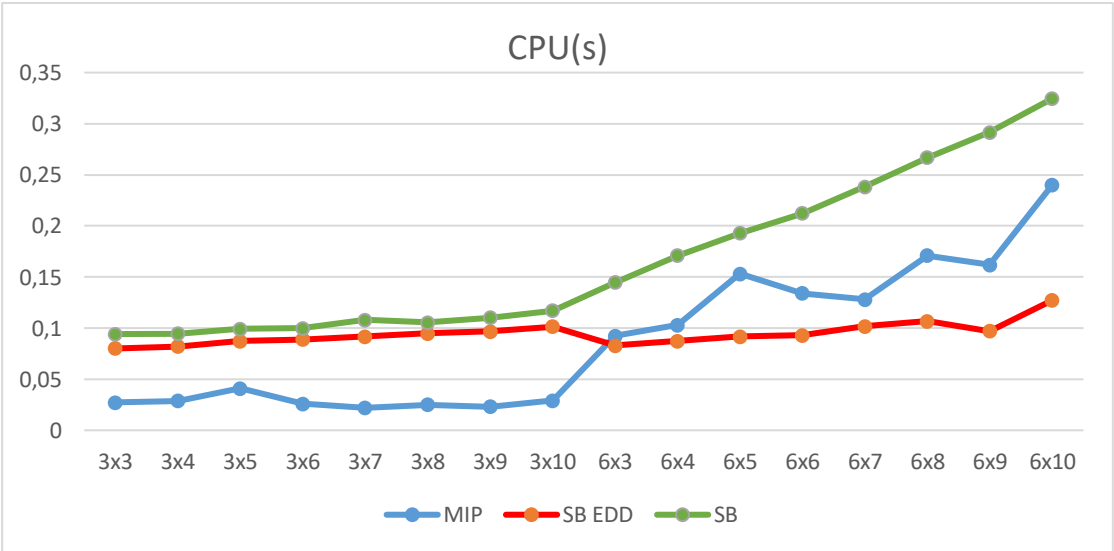

Figure 23 : *Représentation graphique des résultats du CPU(s)*

Maintenant, nous avons changé le critère d'évaluation en calculant le taux CPU(s) utilisé Ce graphe représente les temps d'exécution, on remarque que notre méthode s'exécute en des temps meilleurs par rapport à la méthode SB (originale).

7. Conclusion

Ce chapitre est composé de trois section. La première section est la description du problème à étudier et les étapes de l'heuristique shifting bottleneck. La deuxième section présente l'heuristique proposé pour résoudre notre problème et le modèle mathématique du job shop. La troisième section présente les résultats que nous avons obtenus après la simulation sur plusieurs exemples.

Nous avons expliqué les étapes de l'heuristique shifting bottleneck et modifié cette heuristique et on a appliqué cette heuristique sur plusieurs exemples avec des nombres de machines et des jobs différents et tester son efficacité et sa performance et Comparer les résultats obtenus avec les résultats de modèle mathématique.

CONCLUSION GENERALE

Conclusion générale

Dans ce mémoire, nous avons appliqué une des méthodes heuristiques afin de résoudre un des problèmes d'ordonnancement concernant les ateliers de type Job-shop, et de leur trouver une solution satisfaisante. Ce type de problèmes d'ordonnancement existe dans un grand nombre d'atelier dans différents secteurs de l'industrie.

Les ateliers de type Job-shop sont étudiés d'une façon extrêmement intense par les chercheurs de l'ordonnancement vu qu'ils présentent des problèmes complexes ; en fait, ce type de problème d'ordonnancement est classé parmi les problèmes NP-difficile, qui veut dire qu'en augmentant le nombre de machines et de jobs, on aura une explosion de nombres de solutions possibles. Ceci rend les méthodes exactes obsolètes, où leur application devient non réaliste si on veut obtenir des résultats optimaux. C'est pour cela qu'on utilise des méthodes approchées comme les heuristiques et les méta-heuristiques.

Dans ce travail, nous avons appliqué la méthode « Shifting Bottleneck » modifiée, où nous avons proposé d'ordonnancer les tâches de la machine goulot en utilisant la règle EDD au lieu de résoudre le problème 1|rj|Lmax.

Nous avons simulé notre méthode proposée en utilisant le logiciel Matlab et Cplex, sur plusieurs exemples de tests de nombre de machines et de jobs différents. Les résultats obtenus par l'application de notre méthode, ont été comparées avec les résultats de deux autres méthodes (la méthode shifting SB et une méthode exacte MIP) dont l'objectif est de tester son efficacité et sa performance.

A travers ces résultats, nous avons remarqué que notre algorithme de la méthode proposée présente des résultats satisfaisants avec un temps d'exécution le plus optimal par rapport aux autres méthodes même si le problème est de grande taille (grand nombre de machines et de jobs).

Pour enrichir ce travail, nous pouvons finalement ajouter des perspectives telles que :

- Pour des travaux futurs sur ce thème, on peut utiliser une méta-heuristiques dans l'étape 3 pour trouver des solutions plus fiables (Recuit simulé…).
- Traiter le Problème avec deux machines goulots.

REFERENCES BIBLIOGRAPHIQUES

Références bibliographiques

[1] A. Caumond, «Le problème de jobshop avec contraintes: modélisation et optimisation,» Université Blaise Pascal-Clermont-Ferrand II, 2006.

[2] A. Letouzey, «Ordonnancement interactif basé sur des indicateurs: Applications à la gestion de commandes incertaines et à l'affectation des opérateurs,» Institut National Polytechnique , Toulouse, 2001.

[3] G. Javel, Organisation et Gestion de la Production, 3ème éd., Paris: DUNOD, 2004.

[4] F. Fontanili, «Intégration d'outils de simulation et d'optimisation pour le pilotage d'une ligne d'assemblage multiproduit à transfert asynchrone,» Université Paris XIII, Paris , 1999.

[5] . J. P. Vacher, «Un système adaptatif par agents avec utilisation des algorithmes génétiques multi-objectifs : Application à l'ordonnancement d'atelier de type job-shop N×M,» 2000.

[6] F. Blondel, Gestion de la production, 3ème éd., Paris: DUNOD, 2004.

[7] M. Kebabla, «Utilisation des stratégies Métaheuristiques pour l'rdonnancement des ateliers de type Job Shop,» Université de Batna, 2008.

[8] V. Giard , Gestion de la Production, 2ème éd., Paris: Economica, 1988.

[9] Etudier, «Etudier,» 24 mars 2012. [En ligne]. Available: https://www.etudier.com/dissertations/l'Ordonnancement-De-La-Production/360496.html. [Accès le 15 mars 2019].

[10] M. L. Pinedo, « Scheduling : Theory, Algorithms and systems », New Jersey: Prentice-Hall, 1955.

[11] . F. A. Rodammer et . K. P. White, «« A recent survey of production scheduling »,» *IEEE Transaction on Systems, Man and Cybernetics,* n° %106, pp. 841-851, 1988.

[12] P. Lopez et P. Esquirol, «L'ordonnancement,» Economica, Paris , 1999.

[13] J. Carlier et P. Chretienne, Problèmes d'ordonnancement, Modélisation, Complexité, Algorithmes, Paris: Edition Masson, 1988.

[14] I. Kacem, «Ordonnancement multicritère des job-shops flexibles : formulation, bornes inférieures et approche évolutionniste coopérative,» Université des sciences et techniques de Lille1, Lille, 2003.

[15] Graham, L. Ronald, L. L. Eugene et L. Jan Karel, «Optimization and approximation in deterministic sequencing and scheduling: a survey.,» *Annals of Discrete Mathematics 5,* p. 287–326, 1979.

[16] Y. Harrath, «Contribution à l'ordonnancement conjoint de la production et de la maintenance: application au cas d'un job shop,» L'UFR des Sciences et Techniques de l'Université de Franche-Comté, Besançon, 2003.

[17] V. T'kindt et . J.-C. Billaut, « Multicriteria scheduling: theory, models and algorithms,» Springer Science & Business Media, Berlin , 2006.

[18] B. Penz , «Constructions agrégatives d'ordonnancements pour des Job-Shops statiques, dynamiques et réactifs,» 1994.

[19] S. French, «Sequencing and Scheduling: An Introduction to the Mathematics of the Job-shop,» Wiley, New York, 1982.

[20] D. Duvivier, P. Preux, C. Fonlupt, D. Robilliard et E.-G. Talbi, «The fitness function and its impact on Local Search Methods,» chez *IEEE Systems*, San Diego, USA, 1998.

[21] A. S. Jain et M. Sheik , «A multi-level hybrid framework for the deterministic job-shop scheduling problem,» University of Dundee, Scotland, 1998.

[22] J. Hurink et S. Knust, «Tabu search algorithms for job-shop problems with a single transport robot,» *European journal of operational research 162,* pp. 99-111, 2005.

[23] D. Duvivier, «Etude de l'hybridation des méta-heuristiques, Application à un problème d'ordonnancement de type jobshop,» Université du Littoral Côte d'Opale, Calais, 2000.

[24] W. H. Jeffrey , W. f. john, d. milind et K. Mark, «Handbook Of Production Scheduling,» Springer, NY, 2006.

[25] W. Brinkkötter et . P. Brucker, «solving open benchmark for the job shop,» *Journal of Scheduling 4,* pp. 53-64, 2001.

[26] S. Meeran et A. S. Jain, «deterministic job shop scheduling past present and future,» *European journal of operational research 113,* n° %12 , pp. 390-434, 1999.

[27] B. Christian, «A Generalized Permutation Approach to Job-Shop Scheduling with Genetic Algorithms,» *Operations-Research-Spektrum 17,* pp. 87-92, 1995.

[28] B. Sharma et X. Yao, «Characterizing genetic algorithm approaches to job shop scheduling,» Loughborough University, In UK Workshop on Computational Intelligence, 2004.

[29] . E. Falkenauer et S. Bouffouix, «A genetic algorithm for job shop,» chez *1991 IEEE International Conference on Robotics and Automation*, 1991.

[30] C. Della , Federico, . T. Roberto et Giuseppe, «A genetic algorithm for the job shop problem,» *Computers & Operations Research 22,* pp. 15-24, 1995.

[31] Kobayashi, S., I. & Yamamura, M. et Ono, «An Efficient Genetic Algorithm for Job Shop Scheduling Problems,» chez *Proceedings of the Sixth International Conference on Genetic Algorithms*, San Francisco, 1995.

[32] E. Falkenauer et . S. Bouffouix , «A Genetic Algorithm for the Job-Shop,» chez *Proceedings of the IEEE International Conference on Robotics and Automation*, California, 1991.

[33] M. Yamamura, Ono, Isao et K. Shigenobu, «A genetic algorithm for job-shop scheduling problems using job-based order crossover,» chez *In Proceedings of IEEE International Conference on evolutionary computation*, 1996.

[34] T. Yamada et . R. Nakano, «Genetic algorithms for job-shop scheduling problems,» chez *Proceedings of modern heuristic for decision support* , 1997.

[35] Holsapple, W. Clyde , S. J. Varghese et Ramakrishn, «A genetics-based hybrid scheduler for generating static schedules in flexible manufacturing contexts,» chez *IEEE Transactions on Systems*, 1993.

[36] Dorndorf et . E. Pesch, «Evolution based learning in a job shop scheduling environment,» *Computers & Operations Research 22,* pp. 25-40, 1995.

[37] Sharma et X. Yao, «Characterizing genetic algorithm approaches to job shop scheduling,» In UK Workshop on Computational Intelligence, Loughborough University, 2004.

[38] C. Bierwirth, «A generalized permutation approach to job shop scheduling with genetic algorithms,» *Operations-Research-Spektrum,* pp. 87-92, 1995.

[39] Bierwirth, Christian, Herber, C. Dirk et Mattfeld, «On permutation representations for scheduling problems,» chez *In International Conference on Parallel Problem Solving from Nature*, Berlin, 1996.

[40] D. C. Mattfeld et C. Bierwirth, «Minimizing job tardiness: Priority rules vs. adaptive scheduling,» chez *In Adaptive Computing in Design and Manufacture*, London, 1998.

[41] P. J. M. van Laarhoven, H. L. A. Emile , Lenstra et Jan Karel , «Job Shop Scheduling by Simulated Annealing,» *Operations research 40,* pp. 113-125, 1992.

[42] . E. D. Taillard, «Parallel taboo search techniques for the job shop scheduling problem,» *ORSA journal on Computing 6,* pp. 108-117, 1994.

[43] M. Dell'Amico et . M. Trubian, «Applying tabu search to the job-shop scheduling problem,» *nnals of Operations research 41,* n° %13, pp. 231-252, 1993.

[44] Jain, Anant Singh, Balasubramanian , Rangaswamy, Meeran et Sheik , «New and "stronger" job-shop neighbourhoods: a focus on the method of Nowicki and Smutnicki (1996).,» *Journal of Heuristics 6,* n° %14, pp. 457-480, 2000.

[45] Watson, Jean-Paul, Adele E. , Howe et W. Darrell , «Deconstructing Nowicki and Smutnicki's i-TSAB tabu search algorithm for the job-shop scheduling problem,» *Computers & Operations Research 33,* n° %19, pp. 2623-2644, 2006.

[46] K. Schmidt, «Using tabu-search to solve the job-shop scheduling problem with sequence dependent setup times,» Brown University, 2001.

[47] R. Nakano et T. Yamada, «Job-shop scheduling by simulated annealing combined with deterministic local search,» chez *In Meta-Heuristics*, Boston, 1996.

[48] N. M. Sadeh et Y. Nakakuk, «Focused simulated annealing search: An application to job shop scheduling,» *Annals of Operations Research 63,* n° %11, pp. 77-103, 1996.

[49] M. Kolonko, «Some new results on simulated annealing applied to the job shop scheduling problem,» *European Journal of Operational Research 113,* n° %11, pp. 123-136, 1999.

[50] Adams, Joseph, Balas, Egon et D. Zawack, «The shifting bottleneck procedure for job shop scheduling,» *Management science 34,* n° %11, pp. 391-401, 1988.

[51] M. Pinedo et . M. Singer, «A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop.,» *Naval Research Logistics (NRL) 46,* n° %11, pp. 1-17, 1999.

[52] Demirkol, Ebru, M. Sanjay et . R. Uzsoy, «A computational study of shifting bottleneck procedures for shop scheduling problems.,» *Journal of Heuristics 3,* n° %12, pp. 111-137, 1997.

[53] . E. Demirkol et R. Uzsoy, «Decomposition methods for reentrant flow shops with sequence-dependent setup times.,» *Journal of scheduling 3,* n° %13, pp. 155-177, 2000.

[54] I. M. Ovacik et R. Uzsoy, Decomposition methods for complex factory scheduling problems, Kluwer Academic , 1997.

[55] Mason, J. Scott , W. F. John et C. W. Matthew, «A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops.,» *Journal of Scheduling 5,* n° %13, pp. 247-262, 2002.

[56] Mönch, Lars, . R. Oliver et . R. Sturm, «A simulation framework for the performance assessment of shop-floor control systems.,» *Simulation 79,* n° %13, pp. 163-170, 2003.

[57] . L. Mönch et O. Rose, «Shifting-Bottleneck-Heuristik für komplexe Produktionssysteme: softwaretechnische Realisierung und Leistungsbewertung.,» *DSOR Beiträge zur Wirtschaftsinformatik 2,* pp. 145-159, 2OO4.

[58] Brandimarte, Paolo, M. Rigodanza et L. Roero, «Conceptual modeling of an object-oriented scheduling architecture based on the shifting bottleneck procedure.,» *Iie Transactions 32,* pp. 921-929, 2000.

[59] C.-H. Pan, «A study of integer programming formulations for scheduling problems.,» *International Journal of Systems Science 28,* n° %11, pp. 33-41, 1997.

[60] A. S. Manne, «On the job-shop scheduling problem,» *Operations Research 8,* n° %12, pp. 219-223, 1960.

Annexes

Les résultats obtenus par les trois méthodes :

Problème 3× 3	**MILP**		**SB EDD**		**SB**	
	C_{max}	**CPU (s)**	**C_{max}**	**CPU (s)**	**C_{max}**	**CPU (s)**
1	69	0,09	98	0,080165	83	0,095702
2	75	0,02	92	0,079603	75	0,096931
3	57	0,02	81	0,081496	60	0,098806
4	69	0,02	98	0,077188	83	0,091036
5	72	0,05	84	0,080742	84	0,093094
6	77	0,02	96	0,0839	89	0,090439
7	88	0,01	103	0,080536	96	0,09588
8	88	0,05	100	0,079726	102	0,089402
9	77	0,05	87	0,080731	80	0,099653
10	79	0,02	88	0,076761	88	0,090889
Moyenne	75,1	0,027143	92,7	0,080085	84	0,094183

Problème 6× 3	**MILP**		**SB EDD**		**SB**	
	C_{max}	**CPU (s)**	**C_{max}**	**CPU (s)**	**C_{max}**	**CPU (s)**
1	108	0,09	128	0,079628	128	0,140549
2	102	0,08	109	0,086983	112	0,149949
3	107	0,08	122	0,078637	119	0,138752
4	92	0,08	148	0,079055	112	0,142287
5	103	0,11	115	0,080021	104	0,143695
6	122	0,08	147	0,088787	148	0,152146
7	95	0,09	97	0,082178	103	0,145954
8	102	0,11	154	0,087475	129	0,142196
9	76	0,14	108	0,080793	90	0,141842
10	103	0,05	127	0,086379	105	0,147845
Moyenne	101	0,0925	125,5	0,082994	115	0,144522

Problème 3× 4	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	89	0,05	141	0,079848	127	0,09019
2	68	0,01	91	0,078137	86	0,092531
3	104	0,03	125	0,082656	125	0,098357
4	65	0,02	82	0,083985	74	0,092466
5	87	0,03	105	0,080111	89	0,098149
6	74	0,03	84	0,085328	78	0,097082
7	88	0,02	115	0,08282	107	0,09314
8	93	0,08	106	0,080324	103	0,096078
9	78	0,02	125	0,07983	86	0,090819
10	106	0,02	127	0,086158	122	0,096747
Moyenne	85,2	0,028889	110,1	0,08192	99,7	0,094556

Problème 6× 4	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	160	0,13	210	0,086682	167	0,173543
2	124	0,13	125	0,088463	133	0,160338
3	130	0,09	150	0,085829	162	0,172414
4	161	0,09	180	0,084087	171	0,164254
5	127	0,06	156	0,083978	158	0,169393
6	136	0,11	158	0,08621	170	0,174143
7	125	0,13	158	0,093402	131	0,179752
8	147	0,09	175	0,089496	161	0,169432
9	124	0,09	164	0,086704	166	0,177382
10	114	0,11	125	0,088142	136	0,169284
Moyenne	134,8	0,103	160,1	0,087299	155,5	0,170994

Problème 3× 5	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	83	0,03	94	0,083187	89	0,095705
2	81	0,03	89	0,083463	89	0,104253
3	107	0,03	117	0,088634	111	0,097204
4	112	0,01	130	0,090063	112	0,103202
5	93	0,05	99	0,089057	99	0,105395
6	83	0,02	109	0,08485	93	0,098222
7	106	0,05	138	0,086097	106	0,102521
8	101	0,02	121	0,088553	106	0,093871
9	85	0,08	92	0,086355	102	0,099582
10	78	0,09	92	0,091603	91	0,091913
Moyenne	92,9	0,041	108,1	0,087186	99,8	0,099187

Problème 6× 5	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	124	0,16	126	0,090387	133	0,185306
2	149	0,19	183	0,097347	175	0,191136
3	141	0,08	135	0,085974	136	0,203511
4	113	0,09	117	0,095029	129	0,199559
5	122	0,28	144	0,091869	137	0,194908
6	128	0,2	144	0,09356	148	0,193281
7	86	0,14	127	0,090773	115	0,17866
8	154	0,19	154	0,089487	156	0,182082
9	151	0,11	163	0,091473	182	0,198896
10	122	0,09	154	0,091845	151	0,202303
Moyenne	129	0,153	144,7	0,091774	146,2	0,192964

Problème 3× 6	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	156	0,05	171	0,095402	171	0,10902
2	117	0,03	132	0,089048	119	0,099769
3	131	0,03	157	0,09193	142	0,107322
4	104	0,03	118	0,093706	122	0,099209
5	109	0,02	105	0,087069	105	0,094814
6	96	0,02	102	0,085511	88	0,099685
7	109	0,03	119	0,089506	109	0,098869
8	115	0,01	145	0,083389	123	0,093243
9	126	0,03	138	0,089401	133	0,095291
10	103	0,01	107	0,083911	107	0,100058
Moyenne	116,6	0,026	129,4	0,088887	121,9	0,099728

Problème 6× 6	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	160	0,16	188	0,099173	177	0,220405
2	138	0,11	182	0,089615	164	0,21774
3	174	0,13	221	0,09304	205	0,210136
4	133	0,16	178	0,090642	164	0,215789
5	169	0,17	222	0,094143	200	0,207564
6	144	0,09	186	0,092101	168	0,211456
7	150	0,13	157	0,093722	152	0,206427
8	143	0,13	175	0,091023	167	0,215199
9	148	0,13	158	0,092873	166	0,203058
10	149	0,13	197	0,092748	166	0,214333
Moyenne	150,8	0,134	186,4	0,092908	172,9	0,212211

Problème 3× 7	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	155	0,03	169	0,096255	169	0,112185
2	141	0,03	164	0,090205	151	0,100563
3	131	0,03	154	0,09512	148	0,104441
4	137	0,02	143	0,087548	138	0,120466
5	115	0,02	156	0,090897	136	0,10269
6	139	0,02	145	0,087534	137	0,118613
7	146	0,01	140	0,094122	148	0,103408
8	128	0,02	164	0,090633	146	0,107131
9	138	0,01	179	0,097573	149	0,107649
10	149	0,03	154	0,087783	159	0,104342
Moyenne	137,9	0,022	156,8	0,091767	148,1	0,108149

Problème 6× 7	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	166	0,13	195	0,110814	203	0,233426
2	167	0,11	176	0,095333	180	0,241836
3	186	0,13	223	0,099785	190	0,237297
4	162	0,06	199	0,099885	176	0,239698
5	170	0,16	206	0,097482	186	0,233463
6	150	0,11	211	0,099772	202	0,23911
7	143	0,14	214	0,103018	183	0,237931
8	164	0,14	193	0,101097	173	0,230874
9	166	0,13	230	0,105637	187	0,24317
10	153	0,17	197	0,104051	178	0,245169
Moyenne	162,7	0,128	204,4	0,101687	185,8	0,238197

Problème 3× 8	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	152	0,03	166	0,088391	167	0,101513
2	167	0,03	166	0,093245	166	0,104104
3	157	0,01	178	0,091426	167	0,103104
4	121	0,02	141	0,090676	137	0,103139
5	151	0,03	157	0,097187	157	0,105962
6	185	0,03	186	0,100973	197	0,109954
7	175	0,02	208	0,10936	178	0,107346
8	131	0,02	145	0,091109	151	0,108611
9	167	0,03	215	0,091802	191	0,10479
10	133	0,03	139	0,094202	139	0,107537
Moyenne	153,9	0,025	170,1	0,094837	165	0,105606

Problème 6× 8	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	183	0,27	211	0,110038	204	0,261535
2	186	0,17	232	0,112616	212	0,2732
3	189	0,16	272	0,107824	205	0,279169
4	168	0,16	200	0,104553	176	0,260168
5	192	0,17	231	0,110593	206	0,261184
6	164	0,16	198	0,104751	203	0,276278
7	173	0,14	181	0,105201	183	**0,266306**
8	188	0,17	210	0,105245	188	0,262098
9	166	0,17	213	0,107987	254	0,26424
10	170	0,14	192	0,09958	214	0,263403
Moyenne	177,9	0,171	214	0,106839	204,5	0,266758

Problème 3× 9	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	135	0,02	169	0,09415	163	0,108283
2	180	0,02	210	0,092716	205	0,116212
3	177	0,01	202	0,09923	181	0,108612
4	181	0,03	188	0,099697	185	0,110141
5	165	0,03	179	0,095305	179	0,103463
6	159	0,02	164	0,096037	165	0,104018
7	179	0,03	207	0,097586	205	0,110311
8	165	0,03	169	0,095462	171	0,111894
9	145	0,01	160	0,101817	162	0,109248
10	179	0,03	201	0,096645	192	0,120048
Moyenne	166,5	0,023	184,9	0,096865	180,8	0,110223

Problème 6× 9	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	186	0,23	248	0,099471	238	0,28395
2	191	0,08	230	0,095928	220	0,286799
3	210	0,17	237	0,09925	227	0,290195
4	199	0,14	217	0,096846	224	0,277881
5	224	0,14	295	0,096416	242	0,280511
6	185	0,3	233	0,096427	212	0,29853
7	183	0,11	242	0,097832	229	0,29758
8	203	0,16	258	0,096972	246	0,313883
9	192	0,13	211	0,096212	222	0,285666
10	197	0,16	232	0,09622	254	0,301279
Moyenne	197	0,162	240,3	0,097157	231,4	0,291627

Problème 3× 10	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	185	0,02	223	0,109169	218	0,115221
2	175	0,03	176	0,094185	186	0,113071
3	190	0,03	220	0,106978	225	0,112983
4	162	0,01	191	0,094091	177	0,11824
5	160	0,03	173	0,098589	162	0,113193
6	167	0,02	185	0,099424	183	0,124475
7	187	0,02	198	0,108488	187	0,114304
8	209	0,03	226	0,099986	217	0,117736
9	197	0,05	207	0,100623	209	0,117261
10	180	0,05	211	0,102136	223	0,124087
Moyenne	181,2	0,029	201	0,101367	198,7	0,117057

Problème 6× 10	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	195	0,19	227	0,126732	238	0,310272
2	221	0,55	251	0,12485	253	0,348242
3	211	0,2	274	0,132454	244	0,314791
4	230	0,17	277	0,12236	262	0,322117
5	207	0,22	237	0,133177	256	0,319048
6	244	0,22	290	0,125663	268	0,320307
7	191	0,17	247	0,130365	207	0,320489
8	205	0,23	243	0,126927	245	0,310235
9	242	0,25	267	0,123214	289	0,355
10	223	0,2	250	0,126412	255	0,325572
Moyenne	216,9	0,24	256,3	0,127215	251,7	0,324607

Problème 10× 3	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	176	39,83	187	0,086291	///	Sup à 1 heure
2	193	32,28	214	0,084986	///	Sup à 1 heure
3	150	9,72	159	0,08916	///	Sup à 1 heure
4	190	91,08	197	0,083143	///	Sup à 1 heure
5	174	38,13	175	0,082363	///	Sup à 1 heure
6	157	8,74	185	0,082713	///	Sup à 1 heure
7	220	116,8	223	0,082893	///	Sup à 1 heure
8	172	12,86	172	0,087621	///	Sup à 1 heure
9	201	28,39	201	0,080826	///	Sup à 1 heure
10	179	9,26	205	0,084624	///	Sup à 1 heure
Moyenne	181,2	38,709	191,8	0,084462	///	Sup à 1 heure

Problème 10× 4	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	195	36,08	218	0,087614	///	Sup à 1 heure
2	186	8,52	203	0,08712	///	Sup à 1 heure
3	142	1,95	182	0,090118	///	Sup à 1 heure
4	197	38,31	229	0,089688	///	Sup à 1 heure
5	188	4,02	208	0,092414	///	Sup à 1 heure
6	197	30,72	248	0,090318	///	Sup à 1 heure
7	180	16,3	212	0,091093	///	Sup à 1 heure
8	171	5,89	192	0,091012	///	Sup à 1 heure
9	198	6,76	200	0,088165	///	Sup à 1 heure
10	189	13,3	210	0,091289	///	Sup à 1 heure
Moyenne	195	36,08	218	0,087614	///	Sup à 1 heure

Problème 10× 5	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	193	8,39	266	0,092732	///	Sup à 1 heure
2	203	33,25	205	0,094639	///	Sup à 1 heure
3	161	2,89	232	0,099249	///	Sup à 1 heure
4	190	18,03	277	0,095994	///	Sup à 1 heure
5	186	5,03	285	0,096813	///	Sup à 1 heure
6	162	1,23	270	0,100861	///	Sup à 1 heure
7	177	2,08	215	0,093432	///	Sup à 1 heure
8	166	4,55	216	0,093507	///	Sup à 1 heure
9	161	3,92	204	0,09947	///	Sup à 1 heure
10	215	35,84	267	0,097835	///	Sup à 1 heure
Moyenne	181,4	11,521	243,7	0,096453	///	Sup à 1 heure

Problème 10× 6	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	200	3,49	274	0,103823	///	Sup à 1 heure
2	205	3,58	263	0,102502	///	Sup à 1 heure
3	203	16,16	277	0,106873	///	Sup à 1 heure
4	188	2,56	224	0,104401	///	Sup à 1 heure
5	219	11,95	275	0,103024	///	Sup à 1 heure
6	193	2,08	279	0,10725	///	Sup à 1 heure
7	210	3,73	289	0,10834	///	Sup à 1 heure
8	198	3,63	234	0,102111	///	Sup à 1 heure
9	181	102,1	202	0,105878	///	Sup à 1 heure
10	181	3,25	208	0,103332	///	Sup à 1 heure
Moyenne	197,8	15,253	252,5	0,104753	///	Sup à 1 heure

Problème 10× 7	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	204	5,16	266	0,110004	///	Sup à 1 heure
2	208	5,16	259	0,115673	///	Sup à 1 heure
3	204	2,17	276	0,115137	///	Sup à 1 heure
4	211	3,69	281	0,119022	///	Sup à 1 heure
5	180	3,08	247	0,115407	///	Sup à 1 heure
6	189	8,97	231	0,117829	///	Sup à 1 heure
7	184	1,03	233	0,11534	///	Sup à 1 heure
8	227	2,51	348	0,11793	///	Sup à 1 heure
9	241	2,01	252	0,112503	///	Sup à 1 heure
10	206	1,95	245	0,115198	///	Sup à 1 heure
Moyenne	205,4	3,573	263,8	0,115404	///	Sup à 1 heure

Problème 10× 8	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	228	3	335	0,128464	///	Sup à 1 heure
2	217	2,8	272	0,128066	///	Sup à 1 heure
3	223	5,31	337	0,135025	///	Sup à 1 heure
4	198	4,39	247	0,12664	///	Sup à 1 heure
5	186	1,88	261	0,132484	///	Sup à 1 heure
6	229	9,92	281	0,127511	///	Sup à 1 heure
7	208	2,17	334	0,130474	///	Sup à 1 heure
8	197	2,39	347	0,139554	///	Sup à 1 heure
9	219	5,56	362	0,13184	///	Sup à 1 heure
10	206	2,88	277	0,129482	///	Sup à 1 heure
Moyenne	211,1	4,03	305,3	0,130954	///	Sup à 1 heure

Problème 10× 9	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	227	1,76	309	0,146134	///	Sup à 1 heure
2	254	12,64	309	0,144007	///	Sup à 1 heure
3	221	2,11	303	0,143134	///	Sup à 1 heure
4	224	9,86	330	0,148806	///	Sup à 1 heure
5	238	4,45	269	0,14249	///	Sup à 1 heure
6	245	6,61	303	0,143043	///	Sup à 1 heure
7	245	4,25	409	0,150495	///	Sup à 1 heure
8	227	4,01	345	0,149345	///	Sup à 1 heure
9	259	24,3	349	0,144687	///	Sup à 1 heure
10	235	1,78	368	0,148845	///	Sup à 1 heure
Moyenne	237,5	7,177	329,4	0,146099	///	Sup à 1 heure

Problème 10× 10	MILP		SB EDD		SB	
	C_{max}	CPU (s)	C_{max}	CPU (s)	C_{max}	CPU (s)
1	250	9,5	293	0,162297	///	Sup à 1 heure
2	270	6,59	348	0,163168	///	Sup à 1 heure
3	212	3,3	275	0,16875	///	Sup à 1 heure
4	237	2,48	329	0,162527	///	Sup à 1 heure
5	238	2,31	289	0,166497	///	Sup à 1 heure
6	236	2,25	333	0,167262	///	Sup à 1 heure
7	227	2,17	322	0,16409	///	Sup à 1 heure
8	239	3,25	320	0,160501	///	Sup à 1 heure
9	217	1,91	300	0,173851	///	Sup à 1 heure
10	276	6,63	360	0,153768	///	Sup à 1 heure
Moyenne	240,2	4,039	316,9	0,164271	///	Sup à 1 heure

Résumé

L'aspect le plus difficile du problème d'ordonnancement Job Shop est de trouver et d'obtenir la solution optimale de manière précise, ce qui est presque impossible lorsqu'on augmente le nombre des machines et le nombre des jobs. Cela a poussé les chercheurs à développer des approches et des méthodes approximatives pour trouver une bonne solution dans un temps raisonnable. Le but de ce travail est d'étudier et appliquer une des heuristiques dédiées à ce type de problème c'est l'heuristique Shifiting bottleneck. Cette heuristique vise à atteindre et à rapprocher les résultats du modèle mathématique de job shop qui optimiser la durée totale d'achèvement Cmax.

Mots clés: *Ordonnancement Job Shop, heuristiques, Shifiting bottleneck.*

Abstract

The most difficult aspect of the Job Shop scheduling problem is to find and obtain the optimal solution precisely, which is almost impossible when increasing the number of machines and the number of jobs. This prompted researchers to develop approaches and approximate methods to find a good solution in a reasonable time. The purpose of this work is to study and apply one of the heuristics dedicated to this type of problem is the heuristic Shifiting bottleneck. This heuristic aims to reach and reconcile the results of the job shop mathematical model that optimize the total completion time Cmax.

Keywords: *Job Shop scheduling, heuristics, Shifiting bottleneck.*

ملخص

يتمثل الجانب الأكثر صعوبة في مشكلة جدولة Job Shop في إيجاد الحل الأمثل والحصول عليه بدقة، وهو أمر مستحيل تقريبًا عند زيادة عدد الأجهزة وعدد الوظائف. هذا ما دفع الباحثين إلى تطوير أساليب وأساليب تقريبية لإيجاد حل جيد في وقت معقول. الغرض من هذا العمل هو دراسة وتطبيق أحد الاستدلال المكرس لهذا النوع من المشكلات هو عنق الزجاجة Shifiting bottleneck. يهدف هذا الاستدلال إلى الوصول إلى نتائج النموذج الرياضي لمحل العمل والتوفيق بينها والتي تعمل على تحسين إجمالي وقت الانتهاء Cmax.

الكلمات المفتاحية: *جدولة جوب شوب، الاستدلال، تحويل عنق الزجاجة.*