République Algérienne Démocratique et Populaire

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABOU BEKR BELKAID DE TLEMCEN
FACULTE DE TECHNOLOGIE
DEPARTEMENT DE GENIE ELECTRIQUE ET ELECTRONIQUE

MANUFACTURING ENGINEERING
MELT LABORATORY of TLEMCEN

MEMOIRE

Présenté pour l'obtention du diplôme

Master en Génie Industriel

Intitulé :

Aide à la décision pour la résolution du problème d'ordonnancement Job shop sous contrainte de ressources consommables

Présenté par :

Mr. ABDELAZIZ Adlen
Mr. MESSAI Abdelaziz

Soutenue le 17 Juin 2017 devant le Jury :

Mr. HASSAM Ahmed.	**MCB**	**Président/ Université ABBT**
Mr. BENKROUF Mohamed	**MCB**	**Examinateur/ Université ABBT**
Mr. MALIKI Fouad	**MAA**	**Examinateur/ EPST/Tlemcen**
Mr. HADRI Abdelkader.	**MAA**	**Encadrant/ Université ABBT**
Mr. MKEDDER Med Amine	**Ingénieur de Recherche**	**Co-encadrant/ Université ABBT**

Année Universitaire : 2016- 2017

Remerciements :

Louanges à Dieu le tout puissant, grâce à qui nous avons pu faire tout ce travail.

Nos remerciements s'adressent à toutes les personnes qui, d'une manière ou d'une autre nous ont permis de mener à son terme ce mémoire et plus particulièrement :

A notre encadrant, Mr Hadri Abdelkader qui nous a grandement aidées à définir avec circonspection le sujet de notre travail, et qui par ses conseils et idées, nous a permis d'améliorer la qualité du contenu. Son aide technique a été primordiale pour avancer dans notre projet.

A notre Co-Encadrant Mr Mkedder Med Amine pour ses précieux conseils et remarques qui nous ont beaucoup aidés pour l'avancement de notre travail, et également pour son soutien morale tout au long de notre cursus.

Nos vifs remerciements vont aux membres de Jury, pour avoir accepté de juger notre travail.

A nos enseignants, à qui nous devons notre formation, qu'ils trouvent ici l'expression de nos sentiments les plus respectueux et de notre profonde gratitude pour leur gentillesse, leur disponibilité et leur soutien.

A nos chers parents, nos frères et sœurs et nos familles qui nous ont toujours soutenus le long de notre formation.

A tous nos amis, pour leur soutien morale et aide précieuse.

A Noureddine pour son aide précieuse et pour le temps qu'il a bien voulu nous accorder.

Dédicace :

Moi A.Adlen, je dédie ce travail à tous ceux qui ont cru en moi et m'ont soutenue :

A mes très chers parents, mes deux sœurs Sabah et Assia, Mon frere Ghoulem, que j'aime énormément,

A mon beaux frere K.Hocine, et ma belle Sœur M.Ines,

A mes petites nieces, Malak, Razane, Jana et Mes neveux Mehdi et Rassim à qui je souhaite de longues vies.

A tous mes cousins et cousines.

A toute ma famille Abdelaziz.

A mes amis : Aziz, Moncef, yakoub, Fico, Omar, Adel, Didou, Mouad, Sidali, Mono, Dahmoun, Brahim, Lamine, Ilyes, Hamza, Alae, Sidou, Bouftou, Titouni, Houdayfa, blacko, Bouchra et Naoelle, que je n'oublierais jamais.

Moi M.Abdelaziz, je dédie ce modeste travail à :

Mes très chers parents pour leur soutien et leur noble sacrifice afin de me voir réussir.

Mes frères « Oussama », « Abdelssalam », « Abdalwadoud », et mes sœurs « maimouna » et « chaima », qui j'aime beaucoup,

A toute ma famille, ma belle-famille

Tous ceux qui ont supporté mon absence durant toutes ces années

Tous ceux qui ont contribué de près ou de loin à l'aboutissement de cette thèse

A tous mes amis

Tous ce qui m'aime

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Chaque entité économique (entreprise industrielle, entreprise du bâtiment, administrations, sous-traitants…) doit posséder une certaine cohérence technique et économique pour la réalisation du produit et/ou service. Elle a pour but la satisfaction du client tout en respectant le cahier des charges, les délais, et les coûts.

Des systèmes de production de plus en plus complexes sont mis en œuvre dans le milieu industriel, afin de satisfaire au mieux, qualitativement et/ou quantitativement, et dans un contexte économique donné, une demande provenant d'un marché incertain.

La gestion de ces systèmes pose de nombreuses problèmes dans des domaines variés : gestion de production, marketing, gestion des ressources humaines, environnement, brevets, … Pour résoudre ces problèmes, des techniques issues de discipline très différentes sont employées.

Les problèmes d'ordonnancement apparaissent dans tous les domaines de l'économie : l'informatique (tâches : jobs ; ressources : processeurs : ou mémoire…), la construction (suivi de projet), l'industrie (problèmes d'ateliers, gestion de production), l'administration (emplois du temps).

Dans le cas des problèmes d'atelier, une tâche est une opération, une ressource peut être une machine ou un composant (matière première) dans notre cas, et chaque opération nécessite pour sa réalisation une machine. Dans le modèle de base de l'ordonnancement d'atelier.

Une classification peut s'opérer selon l'ordre d'utilisation des machines pour fabriquer un produit (gamme de fabrication). On rencontre des ateliers à cheminement unique (flow-shop), des ateliers à cheminement multiples (job-shop) et (open-shop) qui est une gamme spécifique, les dates de début des opérations (i, j) constituent les inconnues du problème et leur détermination définit l'ordonnancement. Les contraintes sont de type disjonctif ; le choix d'une séquence sur une machine résout donc les conflits d'utilisation de la machine.

Les méthodes d'ordonnancement des tâches permettent d'avoir une représentation graphique d'une réalisation, qui permet le positionnement relatif des opérations dans le temps.

Les approches les plus connues pour résoudre les problèmes NP-difficiles (dont le job-shop) sont les méthodes constructives, procédures par évaluation et séparation, la programmation linéaire et dynamique, les algorithmes approchés (heuristiques) et méta heuristiques.

Notre travail consistera donc à essayer de résoudre un problème d'ordonnancement d'atelier du type job-shop focalisé sur le système pédagogique ICIM qui se trouve dans le laboratoire Productique MELT de Faculté de technologie, Université de Tlemcen. Auquel nous ajoutons la contrainte de la ressource, notre apport consiste en l'ajout de la contrainte de disponibilité des ressources (Matière Première) dans la mesure où notre ressource est consommable, et donc chaque opération (job) nécessite un certain nombre x de ressources sur la machine d'assemblage, qui possède un stock illimité en MP.

Les méthodes heuristiques offrent l'avantage de ne parcourir qu'une fraction de l'espace de recherche pour parvenir à une solution acceptable ou proche de l'optimum. Dans le cadre de cette étude, nous nous sommes intéressés aux différentes heuristiques (règles de priorités : SPT, LPT, SRC…),

La plus grande partie du travail est l'implantation de la fonction qui permet de calculer le Cmax (makespan) en tenant compte des ressources consommables sous le logiciel Matlab.

Pour l'application de cette fonction, nous nous limitons aux heuristiques SPT, LPT, SRC, LRC. L'objectif essentiel de cette étude concerne l'évaluation des performances de ces heuristiques sur la résolution de notre problème, et de définir quel est la règle qui nous donne les meilleurs résultats en matière de temps d'achèvement.

Notre but sera de trouver le meilleur ordonnancement des jobs afin de minimiser la durée totale de production (Makespan ; Cmax).

Le travail que nous allons réaliser porte sur l'étude théorique, la modélisation et la simulation d'un système de production ICIM, qui est focalisé sur plusieurs stations pédagogique enchainés entre eux (station de la qualité, station d'assemblage, station Tour et Fraiseuse) qui se trouve dans le laboratoire MELT faculté de technologie. Sa mission principale est de faire des expérimentations didactiques pour les étudiants du Génie industriel et de la recherche pour les chercheurs scientifiques du Laboratoire MELT.

Ce manuscrit est structuré en 3 chapitres.

Dans un premier chapitre, nous allons proposer quelques définitions et généralités sur l'ordonnancement, les systèmes de production et les problèmes qui y existent, ainsi qu'un petit aperçue historique et les différents types et classes qui existent. Pour terminer, les objectifs attendus en utilisant une méthode d'ordonnancement et une conclusion.

Pour démystifier notre problème, on va générer un deuxième chapitre, qui sera particulier à l'ordonnancement et au problème du job shop, ou on va citer quelques définitions, ainsi que les problèmes du job shop et leurs types.

Au deuxième chapitre nous nous intéressons particulièrement aux problèmes d'ordonnancement dans les ateliers job shop ainsi qu'aux méthodes de résolution. La première partie comportera des généralités et terminologies liés au job shop ainsi qu'une description mathématique. La deuxième partie est consacrée aux différentes approches de modélisation graphique du job shop et aux différentes méthodes pour la résolution de problèmes d'ordonnancement, puis nous terminerons avec un état de l'art sur le job shop et le job shop sous contrainte de ressources consommables pour mieux intégrer le chapitre 3.

Quant au troisième chapitre, il est consacré au problème étudié, celui du job shop sous contrainte de ressources consommables. Ou figurera une définition formelle du problème ainsi qu'une formulation mathématique simplifié. Nous allons aussi définir notre approche qui consiste en la génération d'un algorithme sous Matlab permettant de calculer le Cmax en tenant en compte la contrainte de ressources, et l'application d'heuristiques pour comparer et traduire les résultats.

Nous terminerons ce manuscrit avec une conclusion générale et perspectives pour d'éventuels travaux futurs.

CHAPITRE I : ORDONNANCEMENT : GENERALITE, SPECIFICITES, COMPLEXITE

I. **Introduction**

II. Définition et généralités sur l'ordonnancement

1. Formulation d'un ordonnancement
2. Aperçu historique sur l'ordonnancement
3. Les types d'ordonnancement

III. Les systèmes de production

1. Définition des systèmes de production
2. Caractéristiques d'un système de production
3. Classification des systèmes de production
4. Les caractéristiques d'ordonnancement dans un système de production
5. Les différents types d'ateliers

IV. Classification des problèmes d'ordonnancement

1. Problème à une machine
2. Problème multi-machines à machines parallèles
3. Problème d'atelier multi-machines à cheminement unique (Flow shop)
4. Problème d'atelier multi-machines à cheminement multiple (Job shop)
5. Problème d'atelier multi-machines à cheminement libres (Open shop)

V. Complexité des problèmes d'ordonnancement

VI. Les objectifs à atteindre lorsqu'on utilise l'ordonnancement

1. Les objectifs liés au temps
2. Les objectifs liés aux ressources
3. Les objectifs liés au coût

VII. Conclusion

Chapitre I : Ordonnancement : Généralité, spécificités, complexité

I. Introduction

Ce premier chapitre est consacré aux connaissances fondamentales sur l'ordonnancement. Nous souhaitons bien que le présent mémoire puisse être parfaitement compris par tout spécialiste d'optimisation combinatoire ou de RO. C'est pourquoi nous avons estimé nécessaire de présenter rapidement dans ce chapitre les définitions et les concepts principaux liés à l'environnement de notre travail.

La majorité des problèmes d'ordonnancement se ramènent à des problèmes d'optimisation. C'est pourquoi on s'intéresse aux techniques d'optimisation pour les problèmes d'ordonnancement. L'optimisation est utilisée pour obtenir des performances étendues ou pour rechercher des performances optimales. Elle est également utilisée pour trouver une solution de bonne qualité.

Pour les chercheurs, l'ordonnancement a été l'un des problèmes les plus complexes étudiés. Cette dissertation se concentre sur le problème de job-shop qui est un cas particulier des problèmes d'ordonnancement.

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.

Dans les systèmes de production, un problème d'ordonnancement se définit comme étant la localisation dans le temps et l'espace la réalisation d'un ensemble de **tâches**, compte tenu des **contraintes** temporelles et des contraintes de ressources.

Les ateliers de production sont des cellules productrices. Ils exécutent les tâches et assurent la transformation des matières premières en produits finis.

II. Définition et généralités sur l'ordonnancement

1. Formulation d'un ordonnancement

1.1. *Définition d'un ordonnancement*

Plusieurs définitions de l'ordonnancement ont été proposés par nombreux auteurs, nous pouvons citer :

- Selon Carlier et Chretienne [9] : « O*rdonnancer c'est programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution … les différentes données d'un problème d'ordonnancement sont les tâches, les contraintes potentielles, les ressources, et la fonction économique ».*
- Présentons une autre définition qui est plus explicite : « *Un ordonnancement constitue une solution au problème d'ordonnancement. Il décrit l'exécution des tâches et l'allocation des ressources au cours du temps, et vise à satisfaire un ou plusieurs objectifs. Plus précisément, on parle de problème d'ordonnancement lorsqu'on doit déterminer les dates de début et les dates de fin des tâches, alors qu'on réserve le terme de problème de séquencement au cas où l'on cherche seulement à fixer un ordre relatif entre les tâches qui peuvent être en conflit pour l'utilisation des ressources. Un ordonnancement induit nécessairement un ensemble unique de relations de séquencement. »* [35].
- *« L'ordonnancement concerne l'affectation de ressources limitées aux tâches dans le temps. C'est un processus de prise de décision dont le but est d'optimiser un ou plusieurs objectifs »*[42].
- *« Le problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu de contraintes temporelle (délais, contraintes d'enchainement) et de contraintes portant sur l'utilisation et la disponibilité des ressources requises … une tâche est une entité élémentaire de travail localisée dans le temps par une date de début … ou de fin … dont la réalisation nécessaire site une durée … et qui consomme des moyens … »* [21]

A partir de ces différentes définitions, on peut conclure quelques principes propres à l'ordonnancement :

Un problème d'ordonnancement consiste donc à d'attribuer une date de début et une date de fin d'exécution à chaque tâche à réaliser, tout en tenant en compte des ressources nécessaires à l'exécution de cette tâche.

Un ordonnancement se doit de respecter un ensemble des contraintes dépendant du type d'atelier tout en optimisant un ou plusieurs objectifs. Ces contraintes peuvent être (contraintes de précédence entre activités, date de début d'exécution, contraintes de ressources …)

Un problème d'ordonnancement dans un système de production, se définit comme étant la localisation dans le temps et l'espace la réalisation d'un ensemble de **tâches**, compte tenu des **contraintes** temporelles, et de contraintes portant sur l'utilisation et la disponibilité des **ressources** requises par les tâches (Machines & MP) [35].

1.2. *Les tâches :*

Une tâche est une entité élémentaire localisée dans le temps par une date de début et/ou de fin, dont la réalisation nécessite une durée, et qui consomme un moyen selon une certaine intensité.
Les tâches peuvent être exécutées par morceaux, ou doivent être exécutées sans interruption ; selon le problème (préemptif et non préemptif). Les tâches sont indépendantes lorsqu'elles ne sont soumises à aucune contrainte de cohérence [26].
Une activité peut être constitué de plusieurs tâches, et plusieurs activités peuvent définir un processus.

1.3. *Les ressources :*

Une ressource est un moyen destiné à être utilisé pour la réalisation d'une tâche [4]. On peut distinguer plusieurs types de ressources :

- Une ressource est **renouvelable** si elle est à nouveau disponible après avoir été déjà allouée, en même quantité quel que soit la ressource (hommes, machines, équipements en général), la quantité de ressource utilisable à chaque instant est limitée.
- Une ressource est **consommable** si la consommation globale est limitée au cours du temps (matières premières, budget). Une ressource est doublement contrainte lorsque son utilisation instantanée et sa consommation globale sont toutes deux limitées.

La disponibilité d'une ressource peut varier au cours du temps, qu'elle soit renouvelable ou consommable, Sa courbe de disponibilité est en général connue a priori, sauf dans les cas où elle dépend du placement de certaines tâches génératrices.

1.4. *Les contraintes :*

Les contraintes expriment des restrictions sur les valeurs que peuvent prendre les variables de décision. On peut distinguer [22] :

1.4.1 Des contraintes temporelles

- *Les contraintes de temps alloué*, elles sont issues généralement d'impératifs de gestion et sont relatives aux dates limites des tâches (délais de livraisons, disponibilité des approvisionnements) ou à la durée totale d'un projet.
- *Les contraintes de cohérence technologique*, ou contraintes de gammes, et qui décrivent des relations d'ordre entre les différentes tâches.

1.4.2 Les contraintes de ressources

- *Les contraintes d'utilisation de ressources* qui expriment la nature et la quantité des moyens utilisés par les tâches, ainsi que les caractéristiques d'utilisation de ces moyens.
- *Les contraintes de disponibilité des ressources* (Sujet d'étude), elles précisent la nature et la quantité des moyens disponibles au cours du temps.

1.5. *Les critères :*

Un critère correspond aux exigences qualitatives et quantitatives qui doivent être satisfaits et qui permettent d'évaluer la qualité de l'ordonnancement établi.

Plusieurs critères peuvent être retenus pour une même application. Le choix de la meilleure solution dépend des critères préalablement définis, pouvant être classés selon deux types, réguliers et irréguliers.

Les différents critères ne sont pas indépendants, certains même sont équivalents. Deux critères sont équivalents si une solution optimale pour l'un est aussi optimale pour l'autre et inversement [10].

***1.5.1** Les critères réguliers :* sont des fonctions décroissantes des dates d'achèvement des opérations. Comme :

- 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.

1.5.2 *Les critères irréguliers : Ce* sont des critères non réguliers, c'est-à-dire qu'ils ne sont pas des fonctions monotones des dates de fin d'exécution des opérations, tels que :

- La minimisation des encours.
- La minimisation du coût de stockage des matières premières.
- L'équilibrage des charges des machines.
- L'optimisation des changements d'outils.

La satisfaction de tous les critères à la fois est souvent délicate, car elle conduit souvent à des situations contradictoires [46] et à la recherche de solutions à des problèmes complexes d'optimisation.

2. Aperçu historique sur l'ordonnancement :

La science de « l'ordonnancement » a célébré ses cinquante neuvièmes anniversaires en 2017. La gestion de projet moderne a évolué comme une conséquence directe de la nécessité d'utiliser efficacement les données générées par les planificateurs et ordonnanceurs dans une tentative de gérer et de contrôler la voie critique et les coûts des projets.

Le contrôle efficace et en temps opportun des données de projet n'est devenu gérable que lorsque des ordinateurs centraux étaient à la disposition de quelques-uns, et les planificateurs et ordonnanceurs de projets étaient alors considérés comme des « professionnels » à part entière. Puis est venu le PC et tout le monde et personne n'étaient un « ordonnanceur » dans les années 1980 et 1990.

Cependant, au XXIe siècle, la nouvelle ère de l'entreprise considère une fois de plus l'ordonnancement comme une profession qualifiée, essentiel au succès des projets et donc aux affaires elles-mêmes.

Le défi consiste maintenant à trouver et à former suffisamment de « bons ordonnanceurs » pour répondre à la demande de l'industrie et du commerce.

3. Les types d'ordonnancement

Il existe plusieurs types d'ordonnancement définis comme suit :

Un ordonnancement est semi-actif lorsqu'il est impossible d'avancer une tâche sans échanger la séquence des opérations sur la ressource. Il est dit actif s'il est impossible d'avancer une opération

sans retarder le début d'une autre opération. Il est dit sans retard ou sans délai si et seulement si aucune opération n'est mise en attente lorsqu'une machine est disponible pour l'effectuer. Ainsi, les ordonnancements sans retard sont inclus dans le sous-ensemble des ordonnancements actifs ; qui sont eux même inclus dans le sous-ensemble des ordonnancement semi-actifs [1].

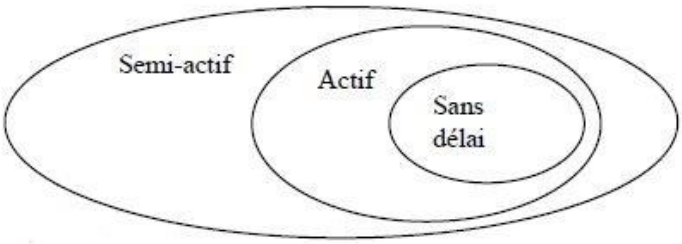

Figure I.1 Ordonnancement Semi-actif/Actif/Sans délai

D'après Baker [3], *l'ensemble des ordonnancements semi-actifs est dominant dans les problèmes d'optimisation d'un critère régulier et le sous-ensemble des ordonnancements actifs est le plus petit ensemble dominant.*

Un ordonnancement est dit sans arrêt (ou sans temps mort) si et seulement si aucune machine n'est mise en attente tant que tous les travaux qui lui sont affectés ne sont pas encore traités.

D'autres types d'ordonnancement sont à distinguer :

- **Ordonnancement statique et dynamique**

Un ordonnancement est ***statique***, si l'ensemble des informations nécessaires à sa résolution est fixé au départ (ensembles des tâches, des contraintes, des ressources, etc.).

Il existe une différence entre la solution proposée qui est généralement accompagnée d'un plan prévisionnel d'exécution des tâches, et l'exécution réelle de ces tâches [38].

L'ordonnancement est ***dynamique***, si le plan prévisionnel n'est pas respecté et les objectifs sont modifiés, et nécessite donc une résolution d'une série de problèmes statiques et chaque étape doit débuter par une prise d'informations permettant d'actualiser le modèle à résoudre.

- **Ordonnancements admissibles**

Un ordonnancement est dit admissible si toutes les contraintes du problème sont respectées (précédence, dates de début, dates de fin, contraintes de ressources, etc.).

- **Ordonnancement centralisée et décentralisé :**
    - **Centralisé** : Chaque centre de charge est défini par un calendrier prévisionnel de fabrication, les ordres de lancement distribués et l'exécution des fabrications contrôlés. Il implique une connaissance précise des gammes de fabrication ainsi que des temps opératoires.
    - **Décentralisé** : qui consiste à gérer devant chaque poste de charge la file d'attente des ordres de fabrications en choisissant l'ordre de passage en fonction de règles de priorité locales.

  On ne peut pas faire de prévisions ni de planning dans un ordonnancement décentralisé, car les décisions sont prises au pied de chaque machine [25].

- **Ordonnancement prévisionnel**

Dans ce type d'ordonnancements, les méthodes classiques ont pour but de générer un ordonnancement optimal, c'est-à-dire minimisant un des critères présentés.

Les problèmes d'ordonnancement ainsi définis sont des problèmes d'optimisation combinatoire [40]. La plupart d'entre eux appartiennent à la classe des problèmes NP-difficiles, et ne peuvent pas être résolus au moyen d'un algorithme polynomial. Ainsi, des algorithmes classiques de l'optimisation combinatoire et un grand nombre d'heuristiques ont été développés.

- **Ordonnancement réactif**

L'ordonnancement réactif est un système qui inclue une méthode pour réagir en temps réel face aux différents risques. Ces risques peuvent être internes survenant dans les ateliers (pannes de ressources, absence de personnel, …), ou externes provenant de son environnement (retard d'approvisionnement, arrivée imprévue d'un ordre de fabrication) [17].

- **Ordonnancement préemptif et non préemptif**

Dépendant du problème, l'ordonnancement est préemptif si les tâches sont exécutées par morceaux (avec interruption).

Un ordonnancement est non préemptif si les tâches sont exécutées sans interruption jusqu'à son achèvement. [38]

III. Les systèmes de production

1. Définition des systèmes de production :

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.

Si l'on définit une entreprise qui utilise des ressources (main d'œuvre, machines, matière première) pour faire de la matière, des produits finis, satisfaisant ainsi ses clients en leur apportant de la valeur ajoutée, le système de production est alors l'ensembles des pratiques, des outils, des règles et méthodes qui font la culture industrielle de l'entreprise [49].

Ces pratiques, règles, outils et méthodes permettent à la main d'œuvre de transformer la matière et faire bon usage des machines, afin de produire de manière efficiente et apporter satisfaction aux clients.

Le système de production permet de structurer la culture industrielle, de mettre en œuvre les mêmes pratiques, d'avoir une approche uniforme de l'organisation et d'aligner toutes les unités et tout le personnel sur une même vision.

2. Caractéristiques d'un système de production

Pour une production efficace, certaines caractéristiques sont indispensables et les entreprises doivent y être munis, telles que la flexibilité, la réactivité, la proactivité et la robustesse.

La figure suivante illustre les caractéristiques d'un système de production.

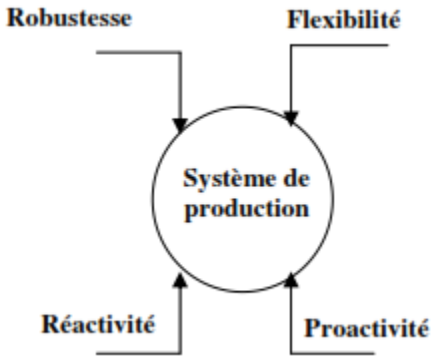

Figure I.2 Caractéristiques d'un système de production

2.1. Flexibilité

On peut distinguer plusieurs définitions de la flexibilité dans la littérature : c'est la capacité du système à s'adopter à des variations de la demande [7], ou aux risques de fonctionnement des ressources. La flexibilité physique est l'aptitude d'un système à modifier ses caractéristiques afin de produire de nouvelles variantes de produits.

L'objectif de ce type de flexibilité (physique) est d'offrir aux clients une grande variété de produits en utilisant les mêmes ressources matérielles.

2.2. Réactivité

La réactivité d'un système de production est l'aptitude à réagir aux changements de son environnement interne ou externe dans un temps requis par rapport au régime de fonctionnement [16].

2.3. Proactivité

L'évolution rapide de l'environnement et la complexité croissante des processus de production conduisent à considérer comme nécessaire une adaptation permanente, dans un monde où les aléas constituent la règle et non l'exception. La réactivité est donc essentielle, mais elle n'est pas suffisante et les systèmes de production doivent présenter une nouvelle propriété : la proactivité.

3. Classification des systèmes de production.

3.1. Classification selon les types de processus

i. Processus continu :

Dans un processus continu, tous les produits sont fabriqués en suivant une séquence d'opération identique ou presque.

Voici quelques particularités de ce type de production :

- Produits relativement semblable.
- Main d'œuvre généralement peu qualifié.
- Machines de production à vocation spécifique (pas de réglage, de changement de série).
- Volumes de production important.
- Exige une bonne synchronisation des cadences des différents postes de travail.

ii. Processus discontinu :

Dans un processus discontinu, les produits sont personnalisés et sont fabriqués en suivant des processus de fabrication différents, un réglage des machines est fait après chaque fabrication d'une série différente.

Quelques particularités de ce type de production :

- Machines de production peu spécialisée exigeant des réglages.
- Volumes de production relativement limités.
- Main d'œuvre très qualifié.

3.2. Classification selon les quantités :

Un ordre de fabrication peut donner lieu à :

- Une production unitaire : où le produit est fabriqué en une unité à la fois.
- Une production en grande série : où la production est de masse.
- Une production en petite série : où la fabrication se fait par lot.

3.3. Classification selon la structure du produit :

i. Structure convergente :

Un seul produit est fabriqué, à partir d'un très grand nombre de matières premières différentes. Ex : industrie de l'automobile.

ii. Structure divergente :

Un grand nombre de produits peut être fabriqués, à partir d'un petit nombre de matières premières différentes. Ex : industrie chimique.

3.4. Classification selon les rapports à la clientèle :

i. Production sur stocks (en flux poussé) :

La production en flux poussé une production basée sur les commandes prévisionnelles à long terme (les dates de lancements sont fixées à l'avance).

ii. Production sur commande (production en flux tiré) :

La production en flux tiré est une production basée sur les commandes fermes à court terme. Les dates de lancements de fabrication dépendent des commandes.

Domaine d'applications	Type de ressources	Type d'opération
Systèmes informatiques	Microprocesseur, bus, interface homme/machine, console, …	Traitement, sauvegarde sur disque, lecture d'un fichier
Systèmes manufacturiers	Machine, opérateur, outils, moyen de transport, …	Transformation, assemblage, transport, maintenance, …
Systèmes hospitaliers	Docteurs, infirmières, lits, bloc opératoire, équipements, …	Opération chirurgicale, accueil du patient, diagnostic, …
Service	Guichetier, Hôtesse d'accueil, …	Traitement de la demande, redirection vers un autre service, …

Figure I.3 Exemples de domaines d'application, leurs ressources et leurs opérations

4. Les caractéristiques d'ordonnancement dans un système de production

Une notation proposée [45] est couramment utilisée pour distinguer un problème d'ordonnancement de manière précise. Cette notation est composée de trois champs d'identification, qui sont notés par le triplet $\alpha / \beta / \gamma$

Champ α :

$\emptyset$: ordonnancement sur une seule machine.

P : ordonnancement sur plusieurs machines parallèles et identiques.

Q : ordonnancement sur plusieurs machines parallèles et uniforme.

R : ordonnancement sur plusieurs machines parallèles et indépendantes.

O : il s'agit d'un problème (Open Shop).

F : il s'agit d'un problème (Flow Shop).

J : il s'agit d'un problème (Job Shop).

Champ β : l'ensemble des contraintes.

Champ γ : le critère à optimiser.

Exemples :

Q2 / pmtn, prec / Lmax : Le problème suivant a pour but de minimiser le retard maximal sur deux machines parallèles uniformes. Où apparaissent des contraintes de précédences et une possibilité d'interruption des tâches.

J3 / RC / Cmax : Ce problème est un problème de minimisation du *Makespan* ou le temps totale d'exécution de tous les jobs en Job Shop à 3 machines avec une contrainte RC (ressource consommable).

n	: Nombre de jobs.
m	: Nombre de machines.
$J = \{J_1, J_2, \ldots, J_n\}$	: Ensemble des travaux ou jobs à réaliser.
$M = \{M_1, M_2, \ldots, M_m\}$	: Ensemble des machines.
n_i	: Nombre d'opérations du job J_i.
O_{ij}	: j^{eme} opération du job i.
P_{ij}	: Durée d'exécution de l'opération O_{ij}.
M_{ij}	: Machine sur laquelle l'opération O_{ij} est exécutée.
$O_i = \{O_{i1}, O_{i2}, \ldots, O_{ini}\}$	: Gamme opératoire du job J_i.
S_{ij}	: Date de début de l'opération O_{ij}.
C_{ij}	: Date de fin de l'opération O_{ij}.
C_i	: Date de fin de job J_i.
C_{max}=max $\{C_i, i = 1,\ldots,n\}$	: Makespan ou date de fin de tous les jobs

Figure I.4 Quelques-unes des notations des problèmes d'ordonnancement

5. Les différents types d'ateliers :

Les ateliers de production sont caractérisés par le nombre de machines qu'ils contiennent et par leur type.

Une classification des problèmes d'ordonnancement dans un atelier se fait en tenant compte du nombre de machines et leur ordre d'exécution pour la fabrication d'un produit.

Comme il est généralement connu, on peut distinguer les trois types d'ateliers suivants : flow-shop, job-shop, open-shop [23].

5.1. Les ateliers de type flow-shop :

Dans ce type d'atelier, les opérations de toutes les tâches passent par les machines dans un même ordre, et la ligne de fabrication est constituée de m machines en série. Ces ateliers sont aussi appelés ateliers à cheminement unique.

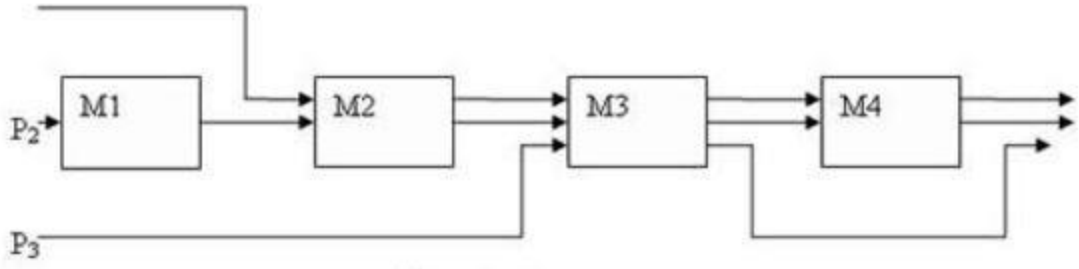

Figure I.5 Atelier du type flow shop

5.2. Les ateliers de type job-shop :

Ce type d'atelier est caractérisé par la réalisation des opérations selon un ordre bien déterminé, et qui varie selon la tâche à exécuter.

Ils sont également appelés ateliers à cheminement multiple [6].

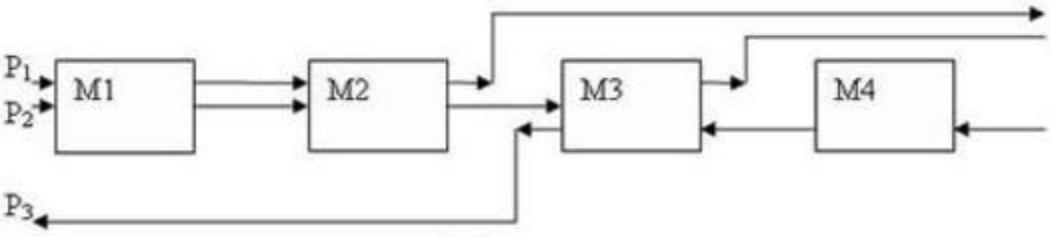

Figure I.6 Atelier du type job shop

5.3. Les ateliers de type open-shop :

Ce type d'atelier est moins contraint que celui des types précédemment cités. Ainsi, l'ordre des opérations n'est pas fixé a priori ; alors, dans ce cas le problème d'ordonnancement consiste, d'une part, à déterminer le cheminement de chaque produit, d'autre part, à ordonnancer les produits en tenant compte des gammes trouvées, ces deux problèmes pouvant être résolus simultanément. Comparé aux autres modèles d'ateliers, l'open-shop n'est pas couramment utilisé dans les entreprises.

IV. Classification des problèmes d'ordonnancement

Les problèmes d'ordonnancement sont très différents d'un système à l'autre, et il n'existe pas de méthode universelle permettant de résoudre efficacement tous les cas.

Une classification peut exister selon le nombre des machines et l'ordre d'utilisation des machines pour réaliser un travail. Un système se définit par le nombre de machines qu'il contient et par son type.

Nous pouvons donc distinguer deux catégories de problèmes ; les problèmes à une machine, et les problèmes multi-machines (machines parallèles, flow shop /flow shop hybride, job shop/job shop flexible et open shop) [36].

1. Problème à une machine

Dans ce genre de problème, l'ordonnancement se fait sur une seule machine, et les jobs sont constitués d'une seule opération. Pour minimiser la durée totale d'exécution ou (makespan) pour des jobs disponibles au tout début t=0, et chaque séquence d'exécution aboutit à une solution optimale. Cependant, l'existence de contraintes et la considération d'autres critères peuvent rendre le problème plus difficile à résoudre.

2. Problème multi-machines à machines parallèles

Ce type de problème d'ordonnancement est une généralisation de celui à une machine, et un cas particulier du problème multi-machines. Une seule opération constitue tous les jobs, et chaque opération est réalisable par n'importe laquelle des machines disposées en parallèle.

Le problème revient alors à affecter les opérations aux machines, ainsi que leurs dates d'exécutions.

Il existe trois types :

- ***Les problèmes à machines identiques :*** où les durées opératoires sont égales et ne dépendent pas des machines.
- ***Les problèmes à machines uniformes :*** où la durée d'une opération varie uniformément en fonction de la performance de la machine choisie.
- ***Les problèmes à machines indépendantes :*** où les durées opératoires dépendent complètement des machines utilisées.

3. Problème d'atelier multi-machines à cheminement unique (Flow shop)

3.1. Problème de type Flow shop

Dans un flow shop, les machines sont alignées en série, et les jobs se composent d'une multitude d'opérations et visitent toutes les machines selon une gamme opératoire unique (dans le même ordre).

Cette gamme opératoire est une donnée du problème. Si le séquencement des jobs est le même sur toutes les machines, le problème est dans ce cas un flow shop de permutation

3.2. Problème du Flow shop hybride (flexible) :

Le flow shop hybride ou (flexible) est une généralisation du flow shop normal et des machines parallèles. Ces ateliers sont constitués en étages d'un ensemble de machines en parallèle. Sauf qu'une opération ne nécessite qu'une seule machine pour l'exécuter. Les jobs à exécuter passent sur tous les étages dans un même ordre. Et donc, cela revient à trouver pour chaque job la machine qui exécute l'opération associée à chaque étage, et aussi les dates d'exécution des différentes opérations.

4. Problème d'atelier multi-machines à cheminement multiple (Job shop)

4.1. Problème de type Job shop

Le problème du job shop est considérer comme une généralisation de celui du flow shop, le passage des jobs sur les machines diffère d'un job à l'autre.

Quand un job passe sur une machine plus d'une fois il y'a une recirculation, et la gamme est dite (bouclante).

4.2. Problème du job shop flexible

Le job shop flexible est disons un alourdissement du job shop classique. La différence est que chaque opération peut être exécutée par une seul machine parmi l'ensemble de machines m.

Le problème consiste alors à déterminer une affectation et un séquencement des opérations sur les machines en fonction de l'objectif à atteindre.

5. Problème d'atelier multi-machines à cheminement libres (Open shop)

Le problème de l'open shop contrairement au job shop, diffère dans les gammes opératoires des jobs, où celles-ci ne sont pas fixées a priori, les opérations sont exécutées dans un ordre non précis. Le problème consiste donc à déterminer le cheminement de chaque job d'une part, et d'autre part à ordonnancer les jobs en prenant en compte les gammes trouvées. Ces problèmes peuvent être résolus simultanément.

L'open shop n'est pas beaucoup étudié dans la littérature, comparé aux autres modèles d'ateliers multi-machines, car il n'est pas aussi courant dans les entreprises.

V. Complexité des problèmes d'ordonnancement

Les problèmes d'ordonnancement d'ateliers sont des problèmes combinatoires difficiles, et il n'existe pas de méthodes universelles précisent permettant de résoudre tous les cas possibles. Une multitude d'algorithmes ont été adoptés et peuvent être utilisés pour la résolution de problèmes d'ordonnancement mais ne sont pas forcément tous efficaces, selon le problème [20].

Les différents algorithmes de résolution peuvent être différencier par le moyen des critères suivants :

- L'efficacité de l'algorithme en termes de durés d'exécution ; un algorithme est dit plus efficace qu'un autre si pour les mêmes données, il s'exécute en un laps de temps plus court.
- L'efficacité de l'algorithme en espace mémoire de stockage ; un algorithme est dit plus efficace qu'un autre si pour résoudre le même problème, il utilise moins d'espace mémoire.
- La fiabilité de l'algorithme ; plus un problème est complexe, plus il y a des risques d'existence de bugs, les bugs étant des erreurs plus au moins évidentes qui se manifestent lors de la mise en exploitation d'un programme. Un programme est jugé plus fiable ou plus stable qu'un autre s'il présente moins de bugs.
- La robustesse de l'algorithme ; elle mesure son degré de tolérance aux erreurs des utilisateurs ; un programme est plus robuste qu'un autre s'il résiste mieux aux erreurs de manipulation des utilisateurs plus ou moins bien attentionnés.

Il n'existe pas de méthodes ou des mesures qui permettent d'évaluer la robustesse et la fiabilité d'un algorithme, c'est à l'usage que ces qualités sont mesurées. Mais on peut trouver des méthodes rationnelles pour évaluer son efficacité, ils sont appelés méthodes d'analyse de complexité des algorithmes.

Deux types de complexité peuvent être cités :

- *La complexité problématique*, elle est liée à la difficulté du problème à résoudre et au nombre des opérations élémentaires qu'un algorithme peut effectuer pour trouver l'optimum en fonction de la taille du problème.
- *La complexité méthodologique*, elle exprime une fonction du nombre d'opération élémentaires de calcul effectuées par la méthode ou par l'algorithme de résolution en fonction du nombre des données du problème traité.

Un problème d'ordonnancement peut appartenir à l'une des quatre classes suivantes, selon son degré de complexité [47] :

- Les problèmes les plus difficiles, qui sont des problèmes pour lesquels il n'existe aucune méthode de résolution ; Ils sont dits indécidables.
- Les problèmes de la classe P, dit polynomiaux, s'il existe un algorithme de complexité polynomiale pour leur résolution.

- Les problèmes de la classe NP, dit problème NP-difficiles, qui ne peuvent à priori être résolus en un temps polynomial que par des méthodes approchées (heuristiques) ; au cours de leur exécution, ces algorithmes font des choix dont l'optimalité n'est pas démontrable.
- Les problèmes NP-complets, un problème de décision, A est dit NP-complet s'il appartient à la classe NP et si pour tout A' de NP :
    - Il existe une application polynomiale qui transforme toute instance I' de A' en une instance I de A.
    - A' admet une réponse « oui » pour l'instance I', si et seulement si A admet une réponse « oui » pour l'instance I.

Autrement dit, s'il existe un algorithme polynomial pour résoudre A, alors, pour tout le reste des problèmes de la classe, il existe des algorithmes polynomiaux pour les résoudre.

VI. Les objectifs à atteindre lorsqu'on utilise l'ordonnancement :

Les entreprises ont diversifié leurs objectifs ainsi que l'ordonnancement est devenu de plus en plus multicritère. Ces critères sont variés. D'une manière générale, on distingue plusieurs classes d'objectifs concernant un ordonnancement [21].

1. Les objectifs liés au temps :

Ce sont les objectifs qui ont une relation avec les différents temps, par exemple la minimisation du temps moyen d'achèvement, du temps total d'exécution, des durées totales de réglage ou des retards par rapport aux dates de livraison.

2. Les objectifs liés aux ressources :

Ce sont les objectifs relatifs aux ressources comme la maximisation de la charge d'une ressource ou minimisation du nombre de ressources nécessaires pour réaliser un ensemble de tâches.

3. Les objectifs liés au coût :

Ces objectifs sont généralement de minimiser les coûts de lancement, de production, de stockage, de transport et tout ce qui a une relation avec les coûts.

VII. Conclusion

Dans ce chapitre, nous avons abordé tout d'abord l'ordonnancement de manière générale, et de manière plus particulière l'ordonnancement de production.

Nous avons présenté aussi des terminologies de tout ce qui est liée aux problèmes d'ordonnancement tels que : tâche, contrainte, ressource et critère…. Ainsi que leurs principales caractéristiques.

Les problèmes les plus difficiles à résoudre sont les problème NP difficile, et qui nécessitent l'utilisation et l'application d'heuristiques pour les résoudre.

Dans un ordonnancement, souvent les objectifs à atteindre sont liés au coût, au temps, et aux ressources.

Pour démystifier notre problème, on va générer un deuxième chapitre, qui sera particulier à l'ordonnancement et au problème du job shop, ou on va citer quelques définitions, ainsi que les problèmes du job shop, leurs types, et les problèmes qui peuvent rencontrer dans ces systèmes.

CHAPITRE II : LE PROBLEME DE JOB SHOP ET MODES DE RESOLUTIONS

I. Introduction

II. Définition et terminologie

III. Le job shop

1. Description formelle du Job Shop
2. Le système de production Job Shop
3. Complexité de management du job shop
4. Appartenance du Job shop à la classe NP-difficile
5. Modélisation graphique du job shop
6. Minimisation de la durée d'un jobshop

IV. Les méthodes de résolution d'ordonnancement

1. Les méthodes exactes
2. Les méthodes approchées ou métaheuristiques

V. Etat de l'art

1. Aperçu historique et état de l'art du job shop
2. Résolution approchée du job shop
3. Le job shop avec ressources consommable

VI. Conclusion

Chapitre II : Le problème de Job shop et modes de résolutions

I. Introduction

Dans ce chapitre, nous présentons les techniques employées pour la résolution du problème de job shop. Nous y portons un grand intérêt car il est classique et est de plus en plus étudié dans littérature de l'ordonnancement, et de plus en plus d'entreprises optent pour ces types d'ateliers. En effet, le job shop :

- Ces ateliers intègrent en plus du problème d'ordonnancement un problème d'affectation des opérations aux machines ou des gammes aux jobs.
- Il généralise les autres problèmes d'ordonnancement, et donc, un algorithme traitant des problèmes de job shop pourrait résoudre des instances de ces problèmes particuliers.

Un problème d'ordonnancement est défini comme un problème consistant à choisir un séquencement des opérations qui s'exécutent sur un ensemble de machines. Ce choix de séquencement doit comporter l'affectation des ressources aux opérations, ainsi que la minimisation ou la maximisation d'un ou plusieurs critères.

Le problème du job shop simple (JSP) s'agit d'un problème NP-difficile ou NP-dur, très difficile à résoudre pratiquement, et pour lequel de nombreuses modélisations existent. Le résoudre consiste à déterminer l'ordre ou le « calendrier » d'exécution de ces opérations en leur attribuant des ressources et des dates de début.

Un problème du type job shop est une modélisation d'une unité de production disposant des moyens polyvalents utilisés suivant des séquences différentes en fonction des produits. L'objectif consiste à ordonner la réalisation des produits de manière à optimiser une fonction objective, en respectant un certain nombre de contraintes sur les machines utilisées pour effectuer chaque opération élémentaire entrant dans la fabrication des produits.

Dans ce chapitre, nous allons évoquer les principaux problèmes du job shop ainsi que les terminologies associées à ce domaine. Et en second lieu, après avoir défini précisément le problème du job shop, nous étudierons ses principales modélisations, et pour terminer, un état de l'art sur le job shop et le job shop sous contrainte de ressources consommables.

II. Définition et terminologie

Le problème du job shop généralise les problèmes classiques de l'ordonnancement tels que le problème à une machine, le flow shop général et le flow-shop de permutation.

Le problème d'ordonnancement des SFP (Systèmes Flexibles de Production) se compose de deux sous problèmes : le problème d'affectation et le problème d'ordonnancement, or, ce dernier est un problème disjonctif dont la forme la plus générale est le problème de job shop.

Il n'est pas facile de trouver une définition universellement acceptée du job shop. Cependant, de nombreuses définitions existantes du job shop couvrent certaines caractéristiques importantes des ateliers job shop. La plupart des unités de fabrication décrites comme des ateliers job shop sont relativement petites en taille et en revenues, et acceptent une variété commandes personnalisées pour des petites quantités. Un job shop est plus général qu'une ligne de production qui peut faire une variété de produits l'un après l'autre séquentiellement. On peut trouver des ateliers job shop dans les industries décrites comme suit :

- Unités de fabrication personnalisées.
- Unités de production Engineer-to-order (ETO).
- Unités de production Make-to-order (MTO).
- Unités de production de grande variété.

Dans ce qui suit nous présentons quelques définitions et notions associés directement au job shop et aux problèmes de job shop :

- **Le job shop généralisé :** un job shop est dit généralisé (étendu) si les produits sont constitués d'un ou plusieurs plans, et que les opérations peuvent être réalisés sur une ou plusieurs machines.
- **Le job shop acyclique :** un job shop est acyclique si aucun produit n'est réalisé en plus d'un exemplaire.
- **Le job shop simple :** le job shop est simple si chaque produit est constitué d'un seul plan de fabrication, et si chaque opération ne peut être effectuée que sur une seule machine. En général un job shop simple est acyclique et la préemption n'est pas autorisée.
- **Le makespan :** Soit Em(x), la date de fin de la dernière opération réalisée sur la machine Mm selon l'ordonnancement x. le makespan Cmax de l'ordonnancement x est défini par : $C_{max}(x)=\max Em(x), 1 <= m <= M$.

Le makespan est la durée nécessaire à la fabrication de tous les produits de l'instance considérée selon l'ordre de fabrication imposé par l'ordonnancement x.

- **L'opération critique** : une opération est critique si elle provoque l'augmentation du makespan de l'ordonnancement, et qu'elle est retardée, sans pour autant que l'ordre d'exécution des opérations défini change [33].
- **Le chemin critique** : est une suite d'opérations critiques liées par des relations de précédence. La longueur d'un chemin critique représente la somme des durées des opérations qui le composent.
- **Le bloc critique** : est une succession d'opérations critiques qui sont exécuté sur la même machine.
- **Problème NP** : un problème est NP (Nondeterministic Polynomia time) s'il peut être résolu en un temps polynomial par une machine de turing non déterministe.
- **Problème NP-complet :** un problème est dit NP-complet s'il est dans NP et si tout problème de NP peut être polynomialement réduit en ce problème [14].

III. Le problème « job shop » :

1. Description formelle du Job Shop :

Ensemble de jobs : $J = \{ j_1, j_2, \dots j_n \}$

Ensemble de machines : $M = \{m_1, m_2, \dots m_m\}$

Les opérations : $O = \{o_1, o_2, \dots o_n\} \quad O_i = \{o_{i1}, o_{i2}, \dots o_{im_i}\}$

Chaque opération à un temps de traitement : $\{\tau_{i1}, \tau_{i2}, \dots \tau_{im_i}\}$

Dans O, on définit A, une relation binaire représente une priorité de précédence entre les opérations. Si, $(v, w) \in A$ alors v doit être effectué avant w.

A induit l'ordonnancement totale appartenant au même job, il n'existe pas de priorité entre les opérations des différents jobs.

Un ordonnancement est une fonction, $S : O \rightarrow IN \cup \{0\}$ qui pour chaque opération v, on définit une heure de début S (v).

Un calendrier S est réalisable si : $len(S) = \max_{v \in O}(S(v) + \tau(v))$

$$\forall v \in O: \quad S(v) \geq 0$$
$$\forall v, w \in O, (v, w) \in A: \quad S(v) + \tau(v) \leq S(w)$$
$$\forall v, w \in O, v \neq w, M(v) = M(w): \quad S(v) + \tau(v) \leq S(w) \text{ or } S(w) + \tau(w) \leq S(v)$$

La durée d'un ordonnancement S est

L'objectif est de trouver un calendrier optimal, un ordonnancement possible de longueur minimale., Min. (Len (S)) [44].

1.1. Présentation :

II est communément admis que c'est le livre « Industrial Scheduling » de Muth et Thompson qui a été le premier à regrouper tous les résultats autour de ce sujet et a servi de base aux recherches qui ont suivies [39].

Un job shop est constitué d'un ensemble fini M, de m machines différentes qui doivent exécuter un ensemble J, de n tâches. Chaque tâche, notée j_i avec i∈{1….,n}, est constituée d'une gamme opératoire, aussi appelé suite linéaire, de n_i opérations. Cette séquence dépend de la tâche et peut donc varier d'une tâche à l'autre. Elle correspond à l'ordre de passage prédéterminé sur chaque machine. Ainsi, une opération O_{ij} (i machine, j tâche) de temps d'exécution ρ_{ij}sera nommée première opération si celle-ci est la première de la séquence formant la tâche. Au total, il y'a donc un ensemble d'opérations à exécuter dans l'atelier. Où chaque machine ne peut effectuer qu'un seul type d'opération.

Les contraintes associées au job shop sont :

- Une machine ne peut effectuer qu'une seule opération à la fois.
- Les tâches sont indépendantes les unes des autres.
- La préemption n'est pas autorisée.
- Les machines n'utilisent pas d'outils en commun ; elles sont indépendantes.
- Deux opérations d'une même tâche ne peuvent être exécutées simultanément.
- Une machine doit être disponible pendant tout l'ordonnancement, même les pannes éventuelles sont prises en compte.

Notons qu'une recirculation est possible dans un job shop, c à d : il est possible qu'une tâche puisse visiter plusieurs fois la même machine, ce phénomène est appelé « gamme bouclante » [8].

1.2. Les critères à optimiser :

Il est souvent difficile dans un job shop, à s'approcher d'une solution optimale en optimisant un critère ou plus, ce qui est surement due à la complexité de résolution du job shop et son appartenance à la classe NP-difficile.

Ces critères à optimiser sont le plus souvent :

- Le makespan – temps total d'exécution de tous les jobs (et c'est le plus commun).
- La durée moyenne des jobs dans l'atelier.
- Le nombre moyen de jobs dans l'atelier.
- L'utilisation des machines.
- L'utilisation des ressources.
- Le retard.

2. Le système de production Job Shop

2.1. Le système job shop classique

Le problème de type job shop est l'un des problèmes les plus étudiés dans la littérature [27] de l'ordonnancement. Son importance théorique ainsi que la modélisation de nombreuses applications industrielles sous forme de systèmes de type job shop le rendent très intéressant.

La particularité des problèmes de type job shop par rapport aux problèmes de production de type flow shop, ainsi que le nombre d'opérations qui n'est pas forcément le même pour tous les jobs. Ces systèmes sont considérés comme des systèmes **fortement combinatoires**.

Ils sont composés d'un ensemble de jobs, chacun composé d'un ensemble de d'opérations qui peuvent être exécutées sur *m* machines en respectant les contraintes suivantes :

- L'ordre de passage des opérations d'un job sur les différentes machines est fixe pour chaque job et peut être différent d'un job à un autre.
- Une machine ne peut exécuter qu'une seule opération à la fois.
- Toutes les machines sont disponibles à l'instant initiale t=0.
- Une opération ne peut être exécutée que sur une seule machine.
- La capacité de stockage entre les machines est considérée comme infinie.
- La préemption des opérations n'est pas autorisée.

La figure suivante représente un problème type de job shop classique composé de 3 jobs et 5 machines dont les gammes opératoires sont :

- J1 : M1, M2, M3, M4, M5.
- J2 : M1, M5, M4.
- J3 : M2, M3, M4.

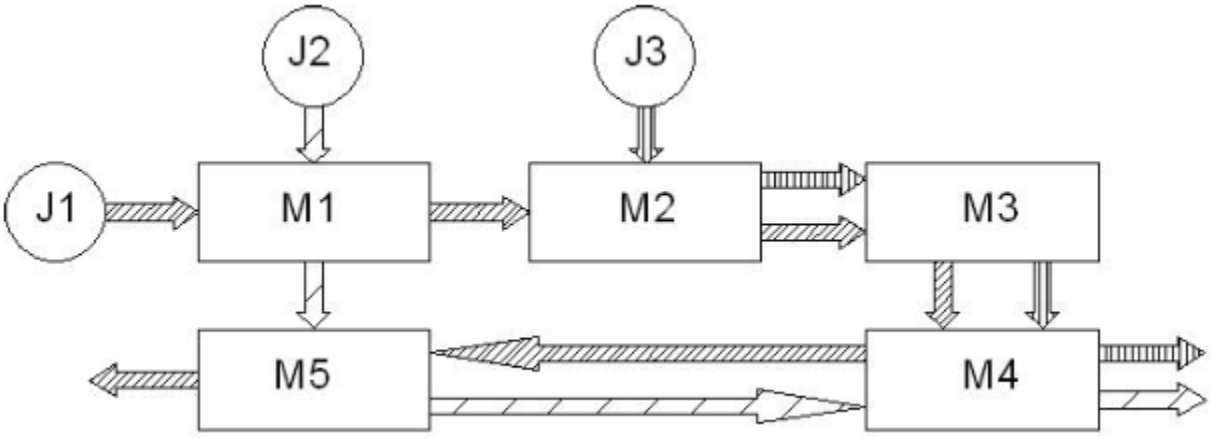

Figure II.1 Problème de job shop classique composé de 3 jobs et 5 machines

L'objectif du problème d'ordonnancement consiste à trouver le meilleur enchaînement des opérations afin de minimiser ou maximiser un certain critère, ainsi que les instants de début des opérations sur chaque machine. Parmi les critères à optimiser, la minimisation de la date de fin de toutes les opérations sur toutes les machines, et qui est considérée dans la majorité des études de la littérature. Ce critère est appelé *Cmax* ou *makespan*.

2.2. Le système Job-Shop Hybride

Une façon d'augmenter la productivité d'un atelier est d'augmenter sa flexibilité. Pour cela, nous pouvons multiplier le nombre de machines qui réalisent la même tache. Le modèle résultant est connu dans la littérature comme un job-shop hybride. Dans ce modèle, les machines qui effectuent la même opération sont groupées dans un même étage. Les problèmes de type job-shop hybride (*JSH*) sont une extension de deux problèmes d'ordonnancement : le problème de job-shop et le problème d'ordonnancement des machines parallèles. Le *JSH* peut être vu comme un problème de job-Shop qui possède une ou plusieurs machines parallèles par étage. La machine nécessaire pour exécuter une opération n'est pas connue a priori. Par conséquent, le problème se compose de deux parties dont la première est de trouver la séquence des jobs sur les étages, la deuxième partie est de trouver la machine nécessaire à l'exécution d'un job tout en respectant les contraintes du job-shop.

Un exemple de ce problème à *m* étages et trois machines maximum par étage, est donné dans la Figure ci-dessous.

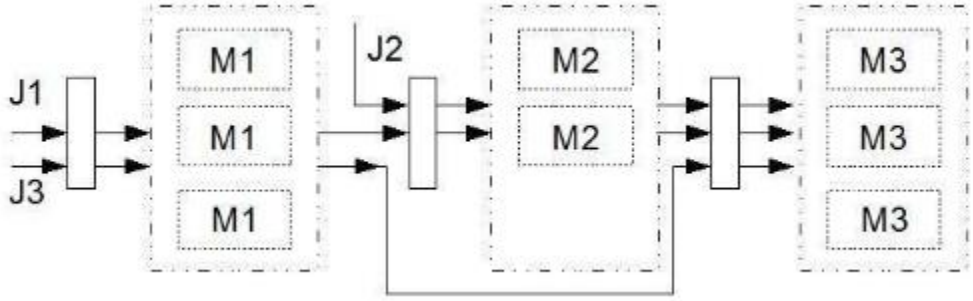

Figure II.2 Représentation d'un système de type Job-shop hybride à trois étages

3. Complexité de management du job shop :

Certaines personnes dans les ateliers job shop ne sont pas confiant quant au contrôle et la gestion efficace de leur production de manière sure, car elles estiment que leur système de production est soumis à trop de variables qui ne peuvent pas être prises en compte. Nombre de ces variables proviennent de la grande variation connue dans les exigences des jobs pour les matériaux, les processus et les ressources, la main d'œuvre, changements dans les propriétés des jobs, les pannes de machines, les retards matériels…etc [48].

Il n'est donc pas facile de prendre en compte toutes ces variables pour la gestion de la production complexe de grande variété. En outre, de nombreux job shop doivent faire un produit

principalement en réponse à une commande reçue d'un client. Les job shops doivent régulièrement être confrontés à une bonne gestion de la production en raison de :

1. La variation élevée des quantités et des exigences de processus des commandes.
2. Les exigences imprévisibles des commandes futures pour les matériaux, les processus et les ressources.
3. Le temps de commande imprévisibles.
4. Les changements fréquents dans les priorités d'un job ou les dates d'échéance.
5. Le mélange de produits à variation continue.
6. Le traitement simultané de jobs multiples et diversifiés utilisant des ressources partagées.
7. Capacité finie des ressources.

4. Appartenance du Job shop à la classe NP-difficile :

Il est très important de connaitre la classe du problème étudié pour mieux l'appréhender. Pour ce qui concerne le job-shop composé de n jobs à exécuter sur m machine il existe $(n!)^m$ solutions possibles. Le job-shop est un problème NP-Difficile et NP-complet et plusieurs démonstrations existent. Pour ce qui concerne le Job-Shop on a les résultats suivants :

• les variantes de Job-Shop J2||C_{max} et J3 | $p_{ij} = 1$ | Cmax sont prouvés NP-Difficiles, dans Lenstra et al.

• les variantes suivantes sont polynomialement solvables :

− n jobs (avec au plus 2 opérations) et 2 machines Jackson (1956).

− J2|pij=1, ri| Cmax Timkovsky et Rubinov (1956).

− le job-shop avec 2 jobs et m machines Brucker (1988).

Supposons qu'on démontre qu'un problème A est NP-Difficile, et que ce problème est réduction d'un problème plus complexe B, alors B est forcément NP-Difficiles. Le job shop avec n jobs et m machines et par conséquent de la classe NP-Difficile, vu que le problème J2||Cmax a été prouvé dans Lenstra et al comme étant déjà NP-Difficile.

Si nous comparons le problème job shop au problème du voyageur (TSP) bien connue en ordonnancement, où les villes sont des jobs et le voyageur est considéré comme étant la machine ;

où on va considérer que chaque job va passer une seule fois sur la machine. Et vu que le TSP est NP complet et difficile, donc le Job shop l'est forcément [34].

5. Modélisation graphique du job shop :

La modélisation d'un problème aussi complexe que celle du job shop doit être précise et claire. On peut distinguer plusieurs façons de modéliser ce problème. La modélisation la plus répondue et connu est celle par graphe disjonctif, tandis que les plus anciennes sont des modélisations en programmation mathématique. Mais il existe aussi des modélisations algébriques, par des graphes potentiels-tâches, à l'aide de réseaux de pétri, modélisations basées sur UML…etc.

5.1. Modélisation par graphe disjonctif :

Le graphe disjonctif est très utilisé pour la représentation des problèmes d'ordonnancement surtout le problème du job shop. Il a été présenté pour la première fois par B. Roy et B. Sussmann en 1964, cette formulation a contribué au développement des premières méthodes de résolution du problème.

Le job shop est modélisé par graphe disjonctif G (X, C*D) où [33] :

- X représente l'ensemble des sommets, chaque opération correspond à un sommet avec deux opérations fictives S (source) et P (puits) qui désignent le début et la fin de l'ordonnancement.
- C représente l'ensemble des arcs conjonctifs représentant les contraintes d'enchaînement des opérations d'une même tâche (gammes opératoires). Il y'a un arc entre tous les sommets (i,j) , (ij+1) pour i=1…n et j=1…n_i ; n est le nombre de tâches, n_i est le nombre d'opérations de la tâche i.
- D représente l'ensemble des arcs disjonctifs associés aux contraintes et aux conflits d'utilisation d'une machine. Pour un arc (ij,ik) de la partie disjonctive on a :

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

Ceci est modélisé par arc et arc retour entre les sommets (t_{jk}, t_{ik}) représentant deux opérations utilisant la même ressource. Pour construire un ordonnancement admissible à l'aide d'un graphe, il est nécessaire de choisir pour chaque paire d'arcs, l'arc qui déterminera l'ordre de passage des deux opérations sur la machine.

Comme est figuré sur l'image ci-dessous, le graphe disjonctif- conjonctif d'un problème job shop avec 3 jobs et 3 machines définit comme suit :

J1 [M1 :4 ; M2 :3 ; M3 :3] ; J2 [M1 :1 ; M3 :5 ; M2 :3] ; J3 [M2 :2 ; M1 :4 ; M3 :1]

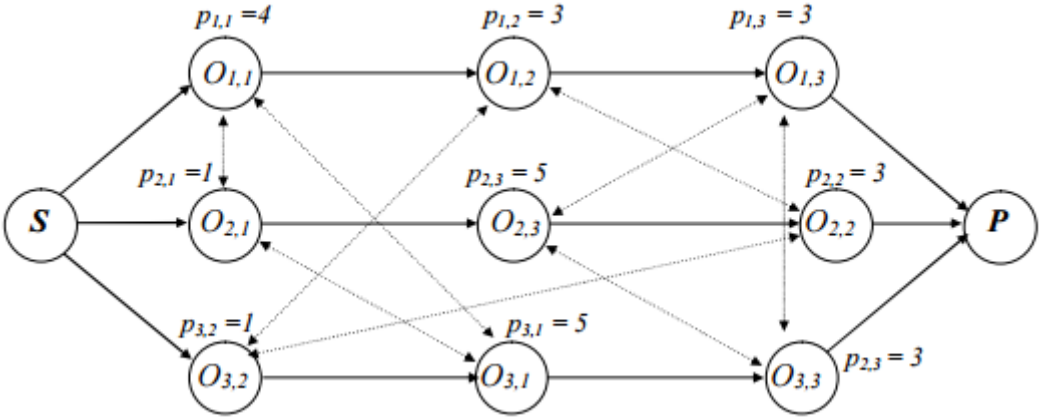

Figure II.3 Exemple d'un graphe disjonctif

5.2. Modélisation par diagramme de Gantt :

Le diagramme de Gantt est le moyen le plus utilisé pour représenter les problèmes d'ordonnancement vu sa simplicité. Développé par Henry Gantt (1861-1919) [27].

Le diagramme est schématisé par un repère dont la ligne horizontale représente les machines, les opérations sous forme de barres avec des longueurs appropriées à leurs durées d'exécution, et l'axe des abscisses représente le temps. Ainsi le diagramme de Gantt détermine l'occupation des machines dans le temps, la séquence de traitement des machines et le temps de fin de traitement des jobs sur les machines. Comme l'est figuré sur l'exemple suivant où les temps opératoires sont représentés comme suit :

	Gamme opératoire		
Job	1	2	3
1	M1 :3	M2 :3	M3 :3
2	M1 :2	M3 :4	M2 :3
3	M2 :2	M1 :3	M3 :1

Tableau II 1 Exemple de la gamme opératoire

Le diagramme de Gantt de ce problème est représenté comme suit :

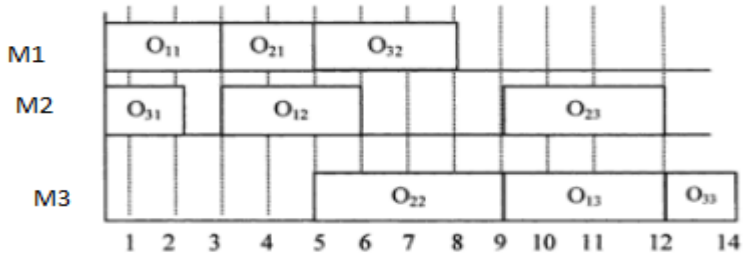

Figure II.4 Diagramme de Gantt (pour l'exemple)

Néanmoins pour de nombreux problèmes, notamment ceux qui contiennent des ressources, on peut distinguer deux types de diagramme de Gantt proposés [33] :

- *Diagramme de Gantt tâches* : il permet de visualiser les séquences des opérations des tâches en représentant chaque tâche par une ligne sur laquelle sont visibles des périodes d'exécution des opérations et les périodes où la tâche est en attente de ressources.
- *Diagramme de Gantt ressources* : est composé d'une ligne horizontale pour chaque machine, où sont visualisées les périodes d'exécution des opérations en séquence et les périodes d'arrêts de la ressource.

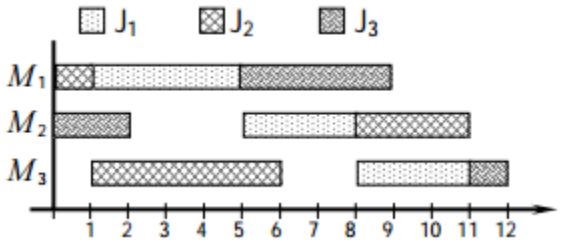

Figure II.5 Gantt ressources

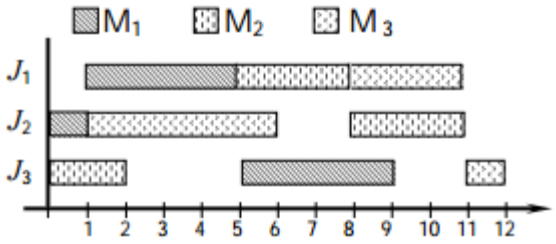

Figure II.6 Gantt tâches

5.3. Graphe potentiels-événements (PERT) :

La méthode PERT est schématisée dans un graphe où les sommets représentent les débuts et fins d'opérations (les événements), et les arcs sont des opérations réelles ou fictives.

A chaque opération o_{ji} sont associés deux sommets F_{ij} et D_{ji} qui correspondes respectivement à la fin et au début de l'opération o_{ji}. Ces sommets sont reliés par un arc, allant de D_{ji} à F_{ij}, et la durée correspond à la durée des opérations. Les contraintes de précédence sont représentées par des arcs fictifs de longueur nulle [43].

La méthode PERT procède en trois phases pour établir les données de l'ordonnancement des tâches [2] :

Phase 1 : consiste à effectuer un parcours du réseau représentant le problème dans le sens des contraintes de précédence pour déterminer les dates au plus tôt des tâches. Puis calculer le plus long chemin entre une tâche et les autres à l'aide d'algorithmes. Dans cette phase on peut connaitre la durée minimale pour la résolution du problème

Phase 2 : est assez similaire à la première. La différence est qu'on commence par la dernière tâche. Puisque on veut minimiser la durée, la date de fin ne sera pas changée, donc on suppose l'égalité la date de fin au plus tard de la dernière des tâches et sa date de fin au plus tôt. Ensuite il suffit de rechercher les chemins les plus longs entre la dernière tâche et les autres.

Phase 3 : cette phase a pour but de déterminer les marges des tâches et les chemins critiques. Qui représente la différence entre sa date de début au plus tard et sa date de début au plus tôt. Ces tâches critiques sont les tâches qu'on ne peut retarder sans augmenter la durée totale.

En plus d'autres modèles de représentation des problèmes d'ordonnancement comme MPM et CPM…etc

6. Minimisation de la durée d'un jobshop

Dans le problème de base de type jobshop, n travaux doivent être exécutés sur m machines, sous des hypothèses quasi-identiques à celles du flowshop.

La seule différence est qui maintenant les séquences opératoires relatives aux différents travaux peuvent être distinctes.

Deux cas particuliers fondamentaux du problème de jobshop sont résolus polynomialement : le cas de 2 machines et le cas de 2 travaux.

Job shop à 2 machines

Cet algorithme, dû à Jackson, constitue une extension de celui de Johnson pour la résolution d'un problème de flow shop n'utilisant que deux machines (n/2/F/Cmax). Il traite de plus le cas où certains travaux peuvent n'avoir à passer que sur une des deux machines. Notant M1 et M2 les deux machines, on peut alors partitionner l'ensemble des travaux en 4 classes :

- O1 : ensemble des travaux ne passant que sur M1.
- O2 : ensemble de ceux ne passant que sur M2.
- O12 : ensemble des travaux passant sur M1 puis sur M2.
- O21 : ensemble de ceux passant sur M2 puis sur M1.

On montre alors qu'un ordonnancement optimal peut être déterminé en adoptant les séquencements O12-O1-O21 sur M1 et O21-O2-O12 sur M2, où les travaux de O21 et O12 sont ordonnancés par la règle de Johnson.

IV. Les méthodes de résolution d'ordonnancement

Après la description des problèmes d'ordonnancement et la représentation des différents ateliers, nous nous intéressons maintenant aux méthodes de résolutions de ces problèmes.

Les problèmes d'ordonnancement étant des problèmes d'optimisation, on distingue deux grandes catégories de méthodes de résolutions : les méthodes exactes, et les méthodes approchées. L'utilisation d'une méthode de l'une ou de l'autre de ces familles dépend de la taille du problème et de sa complexité.

La principale difficulté à laquelle est confronté un ordonnanceur, en présence d'un problème d'optimisation est celui du choix d'une méthode efficace capable de produire une solution optimale en un temps de calcul raisonnable.

Les méthodes d'optimisation sont séparées en deux grandes familles : les méthodes exactes et les méthodes approchées. La figure suivante présente un schéma de l'ensemble des méthodes connues :

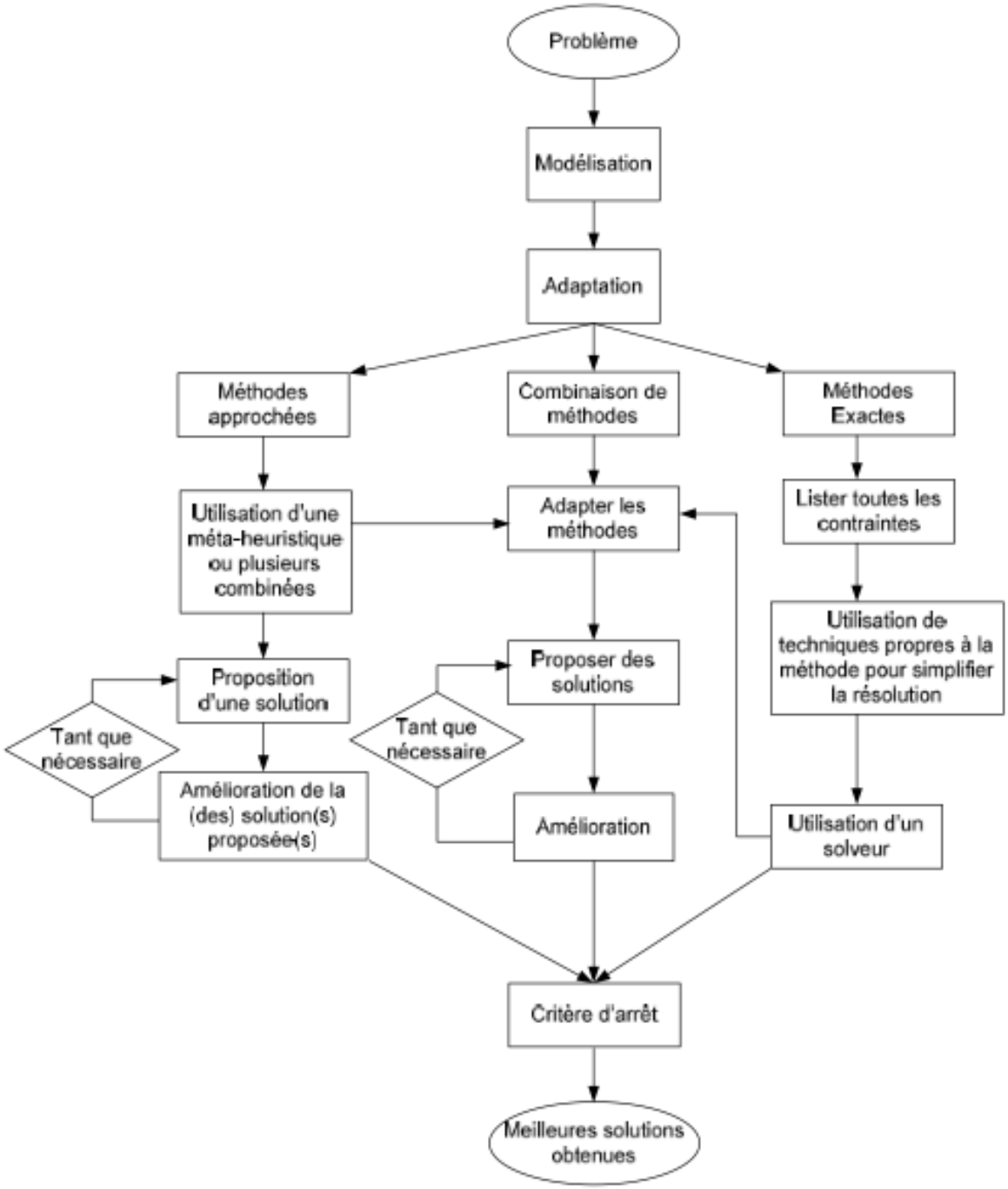

Figure II.7 Schémas d'optimisation [34]

1. Les méthodes exactes :

Les méthodes exactes sont des méthodes qui fournissent une solution optimale pour un problème. Ce genre de méthodes a des limites à cause du grand nombre de solutions qui pourrait exister pour un problème de taille raisonnable [1]. Les plus utilisés sont :

1.1. Les procédures par séparation et évaluation (branch & bound) :

Le Branch & Bound consiste à placer les tâches sur les ressources en explorant un arbre de recherche décrivant toutes les combinaisons possibles. Afin de trouver la meilleur configuration en se séparant des branches de l'arbre qui mènent à de mauvais résultats.

Les procédures par séparation et évaluation (PSE) procèdent par l'énumération de l'ensemble des solutions. Elles sont basées sur la décomposition du problème de façon arborescente en sous problème.

En utilisant cette méthode, on augmente considérablement la chance de trouver la meilleure solution car une analyse de toutes les solutions possibles est faite. La manière d'effectuer la décomposition constitue le principe de séparation qui conditionne la performance de la méthode avec le calcul de bornes qui permet d'éliminer des sous-ensembles de solutions.

La démarche de l'algorithme Branch and Bound consiste à :

- Reproduire les étapes précédentes jusqu'à l'obtention de l'optimum global.
- Chercher une borne minimale en termes de fonction objectif associée à chaque sous espace de recherche.
- Eliminer les mauvais sous-espaces.
- Diviser l'espace de recherche en sous espaces.

1.2. La programmation dynamique

En programmation dynamique, on procède à la décomposition des décisions relatifs aux problèmes en sous ensemble, afin de pouvoir résoudre des sous problème plus grands en utilisant la solution de sous problèmes plus petits. Mais ce n'est applicable seulement si la fonction objective possède la propriété de décomposition. Le problème est alors traité par étapes du plus petit sous problème jusqu'au problème général, tout en tenant compte des informations obtenues lors de chaque étape.

La programmation dynamique ne permet pas alors d'aborder des problèmes de grandes tailles, car elle énumère implicitement toutes les solutions, et de ce fait elle est très exigeante en termes de temps de calculs et de mémoire.

1.3. La programmation linéaire :

La programmation linéaire est une des techniques classiques de recherche opérationnelle. Elle repose sur la méthode du simplexe et les algorithmes de points intérieurs.

Elle consiste à minimiser une fonction cout en respectant des contraintes, le critère et les contraintes étant des fonctions linéaires des variables du problème.

1.4. Les heuristiques

Ce sont des méthodes qui donnent généralement de bons résultats, se basant sur des règles simplifiées afin d'optimiser un ou plusieurs critères.

Le principe d'une heuristique est de construire une solution proche de l'optimale tout en ayant un temps de calcul des plus raisonnable, on peut distinguer parmi ces heuristiques les plus utilisées et qui sont :

- **FIFO** (First In First Out) où la première tâche arrivée est la première à être ordonnancée.

j	1	2	3	4	5	6	7	8	9	10
pj	5	6	3	8	7	2	3	5	4	2

FIFO : 1-2-3-4-5-6-7-8-9-10

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

j	1	2	3	4	5	6	7	8	9	10
pj	5	6	3	8	7	2	3	5	4	2

SPT : 6-10-3-7-9-1-8-2-5-4

- **LPT** (Longest Processing Time) où la tâche ayant le temps opératoire le plus important est traitée en premier.

j	1	2	3	4	5	6	7	8	9	10
pj	5	6	3	8	7	2	3	5	4	2

LPT : 4-5-2-8-1-9-7-3-10-6

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

2. Les méthodes approchées ou métaheuristiques

Les méthodes approchées (métaheuristiques) sont très utilisées car elles donnent un bon compromis entre la qualité des solutions et leur complexité. Contrairement aux approches exactes qui ne peuvent traiter que des problèmes beaucoup plus petits que ceux envisagés dans les applications réelles.

Les métaheuristiques recherchent de bonnes solutions souvent proches de l'optimum en un temps très raisonnable, sans pour autant avoir la garantie de l'optimalité d'une solution, car on ne connaît pas la distance entre la solution trouvée et la solution optimale.

Elles sont ainsi des méthodes de recherche générales, dédiées aux problèmes d'optimisation difficile, qui sont présentées sous forme de concepts généralement.

Les métaheuristiques les plus utilisés sont basées sur la recherche globale comme les algorithmes génétiques, d'autres allant de la recherche locale à des algorithmes de recherche globale tel que les colonies de fourmis et ceux qui utilisent une optimisation par essaim particulaire.

Les méthodes approchées les plus utilisées sont [34] :

2.1. Les méthodes par voisinage : (Appelés aussi méthodes à base de recherche locale)

Dans ces méthodes, on explore l'espace de solution en passant d'un voisin à un autre, à partir d'une solution initiale obtenue d'une façon aléatoire. Le choix d'une solution parmi les solutions voisines est basé sur des critères qui changent d'une méthode à une autre.

Les deux méthodes les plus connues sont la recherche Tabou et le recuit simulé.

a) La recherche Tabou

Cette méthode a été proposé par Fred Glover en 1977, le chercheur s'est basé sur l'utilisation d'une mémoire pour surmonter le problème des optima locaux.

La recherche Tabou est basée sur la notion de mouvements interdits. Elle examine les soluti1ons voisines de la solution courante et choisit la meilleure parmi celles-ci.

C'est une procédure itérative qui à partir d'une solution initiale, essaye de converger vers la solution optimale en exécutant à chaque pas, un mouvement dans l'espace de recherche. Chaque pas consiste à engendrer des solutions voisines, la solution courante en premier lieux, et ensuite choisir la meilleure, même si ce choix provoque une augmentation de la fonction objective à minimiser.

Ci-dessous l'exemple du cas général d'un algorithme de recherche tabou :

Début
Poser x = solution initiale aléatoire
Poser x_{min} = x, f_{min} et TL= Ø (TL : liste tabou) ;
Répéter
Générer un sous-ensemble N de voisinage tel que :
$S_i(x) \subset S(x)\ et\ (x, s_i x))\ \notin TL$
Trouver $f_{1\leq i\leq N}(s(x)) = \min[f(s_i(s))]$;
Ajouter {x, $s_i(x)$} dans TL ;
Remplacer x par s(x) tel que s(x) est la meilleure solution de $s_i(x)$;
Si $f(x) < f_{min}$ **alors**
$f_{min} = f(x)$
x_{min} = x
Jusqu'à condition d'arrêt satisfaites ;
Fin

Figure II.8 Algorithme général de recherche Tabou

b) **Le recuit simulé** :

Le recuit simulé est un algorithme basé sur la simulation de recuit des métaux, il est inspiré des méthodes de la physique statique. Cette méthode est utilisée en aval d'une méthode de recherche locale afin de l'améliorer, par rapport à l'optimum local trouvé.

L'algorithme du recuit simulé permet de résoudre les problèmes de minima locaux, l'acceptation d'une solution sera déterminée en tenant compte de la différence entre les couts et du facteur température comparé à la solution courante. Ce paramètre sert à prendre en compte le fait que plus le processus d'optimisation est avancé, moins une solution qui est plus couteuse est susceptible d'être accepté. Par contre, l'acceptation de solution plus couteuse permet au début de

mieux explorer l'espace des solutions possibles et donc, d'accroitre les chances d'approcher le minimum global. Ci-dessous l'exemple général d'un algorithme de recuit simulé :

Début
Déterminer aléatoirement une solution initiale s_0 et une température initiale T_0 ;
Poser $s$min = s_0 et Ti = T_0 ;
Calculer $f(s_0)$;
Répéter pour chaque itération i/i, ..., n
Choisir $s \in N(s_i)$;
Calculer $\Delta f = f(s) - f(s_i)$;
Si $\Delta f < 0$ alors $s$min= s_i ;
Sinon s_i est rejeté ;
Jusqu'à Ti est proche de 0
Calculer $T = \gamma T$;
Retourner $s$min ;
Fin

Figure II.9 Algorithme général d'un recuit simulé

2.2. Les algorithmes évolutionnistes : algorithmes génétiques

Les Méthodes évolutives, contrairement aux méthodes de recherche locale, manipulent un groupe de solutions admissibles à chacune des étapes du processus de recherche. L'idée consiste à utiliser régulièrement les propriétés collectives d'un ensemble de solutions, appelé population, afin de guider avec efficacité la recherche vers de bonnes solutions dans l'espace de recherche.

Les Algorithmes génétiques sont à base de population évoluant durant un processus évolutionniste, qui est inspiré de la théorie de l'évolution. Les individus sont améliorés à l'aide d'opérations de croisement et de mutation. Ces méthodes sont très pratiques car elles exploitent une multitude de solutions et non pas une seule, et sont souvent faciles à implémenter

Les algorithmes génétiques sont des algorithmes itératifs dont le but est d'optimiser une fonction prédéfinie, appelée fonction fitness.

L'utilisation d'un algorithme génétique nécessite la présence d'un espace de recherche dont les éléments de base sont les chromosomes et d'une fonction définie sur cet espace (fonction fitness) dont la valeur optimale est évaluée en rapport avec les opérateurs de croisement et de mutation choisis [29].

L'algorithme travaille sur un ensemble de points, appelé population d'individus. Chaque individu (chromosome) représente une solution éventuelle du problème. Les éléments le constituant sont des gènes, dont les valeurs sont appelées allèles.

Cinq éléments de base sont nécessaires pour l'utilisation des algorithmes génétiques [18] :

1. **Le principe de codage** des éléments de la population, qui consiste à associer à chacun des points de l'espace d'état une structure de données définit, bien que le codage binaire ait été très utilisé à l'origine, les codages réels sont de plus en plus utilisés, surtout dans les domaines applicatifs pour l'optimisation de problèmes à variables réelles.
2. **Le mécanisme de génération** de la population initiale, doit être capable de produire une population d'individus non homogène servant de base pour les générations futures, là, le choix de la population initiale joue un rôle important car il influence la rapidité de la convergence vers l'optimum global, dans le cas où l'on ne dispose que de peu d'informations sur le problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.
3. **La fonction à optimiser**, appelée fitness ou fonction d'évaluation de l'individu.
4. **Les opérateurs** qui permettent de diversifier la population au cours des générations et d'explorer l'espace d'état, l'opérateur de croisement recompose les gènes d'individus existant dans la population alors que l'opérateur de mutation garantit l'exploration de l'espace d'état.
5. **Les paramètres de dimensionnement**, représentés par la taille de la population, le nombre total de générations, ou le critère d'arrêt, ainsi que les probabilités d'application des opérateurs de croisement et de mutation.

Ci-dessous la structure générale d'un algorithme génétique.

Début
Génération d'une population aléatoire initiale de k individus ;
Répéter
Evaluation de la fonction objective f pour chaque individu ;
Sélection des meilleurs individus ;
Croisement des individus sélectionnés ;
Mutation de certains individus de la population ;
Jusqu'à conditions d'arrêt satisfaites ;
Retourner la meilleure solution ;
Fin

Figure II.11 Algorithme général d'un AG

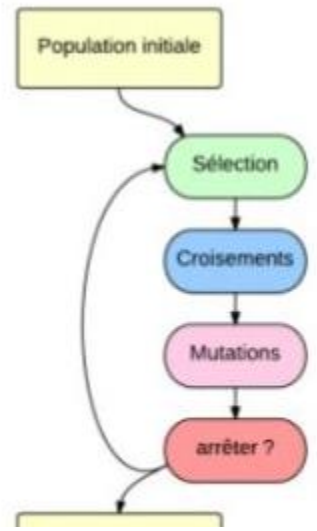

Figure II.10 Principe de fonctionnement d'un AG

2.3. Les algorithmes de recherche globale

a) Les algorithmes de colonies de fourmis :

Les algorithmes de colonies de fourmis sont inspirés du comportement des fourmis, qui sont capable de trouver le chemin le plus court, du nid à une source de nourriture, et de s'adapter aux changements de l'environnement.

Ces algorithmes ont donc pour but de reproduire le comportement naturel des fourmis pour retrouver le meilleur chemin possible vers un objectif donné.

b) Les algorithmes d'optimisation par Essaim Particulaire :

Le concept d'optimisation par Essaim Particulaire (OEP) fut développé par James Kennedy et Russel Eberhart en 1995 [6], leurs recherches portaient sur la simulation de la capacité des oiseaux à voler de façon synchronisée, ainsi que leur aptitude à changer de direction d'une manière brusque, tout en restant en formation optimale.

La procédure de cette méthode (OEP) est basée sur une population d'individus, appelés particules, et qui changent leur position (état) avec le temps.

Les particules se déplacent dans un espace de recherche lors d'une OEP. Ainsi il est possible de combiner des méthodes de recherche globale (métaheuristiques) avec des méthodes de recherche locale.

A la différence des autres méthodes, L'OEP n'utilise pas l'opérateur de sélection qui choisit les individus gardés dans la prochaine génération.

En effet, tous les membres de la population sont maintenus par le procédé de recherche de sorte que l'information soit toujours mise en commun entre les individus pour diriger la recherche vers les meilleures positions trouvées dans l'espace de recherche.

V. Etat de l'art

1. Aperçu historique et état de l'art du job shop :

Le concept du job shop est née il y'a plus de quarante ans. Et c'est dans les années 1960 qu'est apparu le problème d'ordonnancement aujourd'hui bien connu, celui de Fisher et Thompson avec 10 jobs et 10 machines (1963). Ce problème résista pendant près de 25 ans et suscita la curiosité et la compétition entre les quelques chercheurs du domaine tel que Blazwicz et al (1996)[34].

Dans leur article, ils ont dressé un état de l'art du job shop. Ils ont aussi introduit les méthodes de résolutions à l'aide d'équation linéaires ainsi que la modélisation sous forme d'un graphe disjonctif-conjonctif. Et comme nous le savons déjà, les méthodes de résolution sont séparées en deux familles :

1. Les méthodes exactes pour lesquelles ils ont donné tous les éléments nécessaires à la construction et l'exécution d'un algorithme exact ainsi que les méthodes de branchement et les inégalités valides.
2. Les méthodes approchées pour lesquelles ils décrivent quelques règles de priorité, les algorithmes de recherche locale, la procédure à machine goulot ainsi que des procédures d'ordonnancement opportunistes. Ils ont également énoncé les principaux voisinages utilisés dans la littérature.

Dans les publications de Jain et Meeran (1999), ils listent les articles en les classant par méthode de résolution utilisée pour le JS, ils proposent aussi une liste d'instances qui sert de base de comparaison entre les méthodes de résolution.

Les méthodes exactes ont débuté avec les travaux menés par Manne (1960), puis par Brooks et white (1965) suivi en 1668 par greenberg qui utilisa une formalisation linéaire en nombre entiers. De nombreuses publications ont fait leurs apparitions : celles de Ficher qui utilisa les multiplicateurs de Lagrange. En 1975 Florian et McMahon développent un algorithme dépassant la performance des algorithmes de Ficher.

Lors des 20 dernières années, des algorithmes plus performants ont été créer utilisant la relaxation lagrangienne qui consiste dont l'idée est d'abandonner certaines contraintes pour résoudre des problèmes plus proches de la réalité avec plus d'une centaine de jobs et 50 machines. Cette technique de relaxation fut travaillée pour associer à chaque contrainte relâchée une pénalité appelée technique de relaxation lagrangienne augmenté.

En 1989, Carlier et Pinson ont décrit un algorithme bien connu aujourd'hui basé sur le branch and bound, qui a pu résoudre de manière optimale le problème 10 jobs et 10 machines de Fisher et Thompson ainsi que d'autres plus complexes.

Caseau et Laburthe (1995) et Blazewicz et al (1998) ont proposés de nouvelles approches, bien qu'elles fournissent des résultats exacts, mais nécessitent souvent un temps d'exécution très long pour des grands problèmes. C'est pour cela que d'autres méthodes approchées ont été développées.

Ces méthodes approchées offrent une alternative intéressante, car souvent, les résultats sont obtenus rapidement. Ces dernières peuvent être envisagées pour le traitement de problèmes de grande taille.

Les premières méthodes sont des heuristiques de construction ont été proposés par Gliffer et Thompson et Fisher et Thompson à la fin des années 50 et début des années 60. Malgré le grand intérêt pour les méthodes exactes, d'autres méthodes de relaxation ont vu le jour lors des années 70, suivi par l'apparition des métaheuristiques au début des années 80, telles que la recherche Tabou, le recuit simulé et les algorithmes génétiques.

Ce n'est qu'après, qu'ont émerger d'autres méthodes telles que les réseaux de neurones, la programmation par contraintes, les systèmes experts ainsi que le recours à la simulation, afin de résoudre les problèmes d'ordonnancement [14].

Les travaux sur les problèmes de job shop n'ont cessé de progresser au fur des années, notamment l'utilisation de métaheuristiques tels que les AG pour leur résolution, citons :

- La développement d'une application qui permet de minimiser le makespan pour un problème de job shop flexible à l'aide d'un AG, l'article fut publié par Kheireddine MERHOUM et Messaoud DJGHABA [37], où ils ont testé les performances des algorithmes génétiques dans la résolution de ce problème. Leurs solutions globales étaient liées seulement au nombre de machines et au nombre de job, leur étude portait sur le changement à chaque fois du nombre de job, avec un nombre de machines toujours fixes. Les résultats obtenus étaient très satisfaisant en termes de qualité/temps, mais l'optimalité reste une solution approchée.
- Une autre application des métaheuristiques pour traiter la problématique de l'ordonnancement job shop, les travaux ont été menés par KEBABLA_MEBAREK [33],

les méthodes retenues dans sa thèse sont : les algorithmes génétiques, la Recherche Tabou et le recuit simulé.

La structure théorique de ce mémoire a consisté à présenter : les problématiques de l'ordonnancement en général, le problème d'ordonnancement des ateliers de type Job shop et les métaheuristiques.

Le travail se concentre sur l'implémentation informatique de ces méthodes dans leurs formes simples (standards) avec plusieurs alternatives pour chacune, pour dégager des résultats prouvant leur efficacité.

Le travail permet de constater que les méthodes utilisés (simples) ont parvenus à des solutions pratiquement très satisfaisantes, ce qui prouve que le recours à des techniques trop sophistiqués n'est pas toujours fiable, l'utilisation de ces trois heuristiques ne peuvent conduire à une conclusion de supériorité d'une de ces méthodes par rapport aux autres [33].

- En 1992 Van Laarhoven et al ont proposé un recuit simulé pour le problème de job shop. Ils ont utilisé le principe du recuit simulé en le chemin critique dans le graphe conjonctif - disjonctif lors du voisinage. Ce voisinage est dit guidé car il cible les ordonnancements qui peuvent être améliorés au cours d'une itération, les propriétés des chemins critiques ont été adopté pour l'obtention de ce voisinage.
- Dans les publications de Jensen (2001) et Mattfeld (1995), des algorithmes génétiques hybrides ont été proposé. Les deux AG ont pourtant des structures différentes, mais utilisent tous deux un voisinage qui exploite le chemin critique. Ces deux algorithmes travaillent sur l'ensemble des optimaux locaux qui est de taille restreinte, mais comporte des ordonnancements de bonne qualité en moyenne.
- Dans la thèse de Davide DUVIVIER [19], en vue de l'obtention du grade de doctorat à l'université de littoral côte d'opale sous, le titre « Etude de l'hybridation des métaheuristiques, application à un problème d'ordonnancement de type job shop », année 2000. L'auteur a utilisé des modèles hybrides basés sur la collaboration de plusieurs méthodes de résolution : bien qu'elle ne soit pas une notion nouvelle dans le monde de la RO, il s'agit d'une technique récemment apparue dans le domaine des algorithmes évolutifs. Elle permet d'obtenir d'excellentes performances en tirant au mieux profit des caractéristiques des algorithmes évolutifs d'une part, et des méthodes de recherche intégrées dans les algorithmes hybrides d'autres part.

- D'autres travaux sur le job shop ont été présentés par Anthony CAUMOND [14] pour obtenir le grade de docteur à l'université Blaise pascal- Clermont ferrand 2, sous le titre « le problème de job shop avec contraintes : modélisation et optimisation », en 2006. Dans cette thèse, l'auteur a choisi de mettre en œuvre la démarche sur deux problèmes : un problème d'ordonnancement d'atelier de forge de l'entreprise Aubert & Duval, et le problème d'ordonnancement des systèmes flexibles de production. Le travail a été réparti en deux phases : Modélisation et Optimisation.

  Il a retenu le problème de job shop comme modèle théorique sous-jacent à ces deux problèmes qui sont modélisés par des extensions : le job shop avec time legs, job shop avec transport et contraintes additionnelles. Pour leurs résolutions, A.C se focalise sur le graphe conjonctif-disjonctif, et à l'aide d'un algorithme du plus long chemin, il a pu proposer des heuristiques et des métaheuristiques pour ces problèmes.
- La thèse présentée par Hela BOUKEF BEN OTHMAN [6], en vue d'obtenir le grade de docteur de l'école centrale de Lille en 2009, sous le titre : « Sur l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharmaceutiques, optimisation par algorithmes génétiques et essaims particulaires ». Où elle adopte des algorithmes génétiques et la méthode d'optimisation par essai particulaire qui sont des méthodes évolutionnistes pour la résolution de multi-objectifs d'ordonnancement en industrie pharmaceutiques (assurer la production en quantité et surtout de qualité irréprochable, dans les délais impartis et tenant compte des différentes saisons tout en minimisant les coûts).

  Les résultats obtenus ont confirmé la capacité des algorithmes génétiques à s'approcher de la solution optimale, ainsi que la MOEP, car les résultats obtenus comparés avec ceux de la méthode des AG sont satisfaisants Vu que le Cmax obtenu pour tous les cas est le même pour les deux méthodes, surtout que la méthode BOEP converge plus rapidement que la méthode des AG.
- Une autre thèse présentée par Ali GORINE [27] pour l'obtention du grade de docteur de l'université Paul Varlaine-Metz, sous le titre : « Ordonnancement de systèmes flexibles avec contrainte de blocage », année 2011, les principaux travaux que comporte cette thèse sont :
  - Présentation d'un état de l'art porté sur les problèmes du type job-shop et flow-shop.
  - Modélisation du problème de job shop avec la contrainte de blocage.

- Proposition d'un modèle PLNE (Programmation Linéaire en Nombre Entiers).
- Développement d'une méthode de résolution exacte pour le problème.
- Développement d'une métaheuristique basée sur l'algorithme de recuit simulé pour le problème étudié.
- La plus grande partie de l'étude est consacré à la contrainte de blocage dans les systèmes du type job shop hybride

Développement d'une méthode de résolution exacte pour ce problème.

2. Résolution approchée du job shop :

De nombreuses tentatives de résolution du cas général par un algorithme de liste en utilisant des règles de priorité ont été proposées depuis 1950. Le principe général de ces heuristiques par construction est le suivant : on ordonnance à chaque instant t (où une machine et au moins une tâche sont disponibles) la tâche de priorité maximale conformément à la règle de priorité retenue. Parmi les différentes règles de priorité testées, on peut citer FIFO : sélection de l'opération disponible le plus tôt, SPT : sélection de l'opération de plus petit temps opératoire, MTR : sélection de l'opération ayant le plus grand nombre de tâches restant à exécuter dans sa séquence opératoire et MWKR : sélection de l'opération correspondant à la plus grande quantité de travail restant à exécuter.

Parmi les travaux réalisés dans ce sens, on peut citer Lawerence, et une synthèse comparative de Adams, Balas et Zawack, montrant que le dénominateur commun de ces performances en fonction des jeux de données testés.

Plus récemment, ces derniers proposèrent une méthode de résolution approchée de ce même problème, « The Shifting Bottleneck Procedure » dont les résultats sont très satisfaisants même sur des problèmes de taille importante. Le principe de cette méthode est le suivant : partant du problème global, ils ordonnancent localement les machines une à une, chacune de manière optimale (en utilisant la méthode arborescente proposée par Carlier [11] en 1982). Ils propagent pour chaque nouveau séquencement les contraintes résultantes à l'aide du graphe conjonctif associé au problème global. L'ordre dans lequel les machines seront séquencées dépend d'une évaluation des goulets d'étranglement associés à chacune d'entre elles. Chaque fois qu'une nouvelle machine est ordonnancée, ils réoptimisent le séquencement de toute machine déjà ordonnancée dont l'ordre induit peut-être améliorer. Dans la méthode arborescente associée,

chaque nœud correspond donc à un sous-ensemble de machines ordonnancées. Dans le but de limiter la taille de l'arborescente générée, une fonction de pénalité, calculé en chaque nœud, mesurant sa déviation par rapport au goulet d'étranglement, et pondérée en fonction de son niveau dans l'arbre de recherche, est utilisé.

3. Le job shop avec ressources consommable :

Dans ce qui va suivre, nous allons présenter quelques travaux qui ont traité les problèmes d'ordonnancement avec la contrainte de ressource consommable.

A.Janiak, C.N.Potts et T.Tautenhahn [30] ont travailler sur un problème d'ordonnancement pour la minimisation de Cmax, où les contraintes de précédences sont imposées, ainsi que les dates de disponibilités de tâches, et qui dépendent d'une ressource additionnelle. Or, la date de disponibilité de chaque tâche est en fonction de la ressource qui lui est allouée avec une quantité déterminée.

On rencontre ce genre de problème dans les ateliers de production des feuilles métalliques, l'étape principale est le laminage à chaud, où les tâches sont préchauffées par un gaz durant un certain moment dépendant de l'intensité du flux du gaz avant de passer au laminage.

Si les quantités allouées aux tâches sont fixes et déterminées, le problème particulier est équivalant au 1| *prec,r* |*C* et, il est résolu par la règle de Jackson $O(n^2)$. Les auteurs ont développé différentes heuristiques à ce problème, en fixant un ensemble de valeurs possibles des dates de disponibilités. Le cas le plus simple est lorsque toutes les tâches ont des dates de disponibilités égales.

Dans le cas où 2 valeurs sont allouées, le programme est à valeurs et devient un problème de knapsack (sac à dos). Dans le cas général où *k* valeurs sont allouées et où les contraintes de précédence sont supprimées ; les auteurs ont montré qu'une LP-relaxation du programme à valeurs entière peut être utilisée pour ordonnancer (*n-k*) tâches de façon optimale.

K.De Bontridder [31] a utilisé des méthodes de recherche locales pour la résolution de problème d'approvisionnement et de planification de la production en présence d'une ressource non renouvelable, dans le problème job shop proposé, les dates de disponibilités, les règles de précédence, les dates de début et de fin des tâches, ainsi que la contrainte de ressource non renouvelable sont imposées, afin de trouver un ordonnancement qui minimise la somme des retards. Dans ce contexte, la ressource est livrée avant l'exécution des tâches, ou est produite par d'autres tâches dans le même système, l'auteur propose une heuristique basée sur « list

scheduling » pour ordonnancer et respecter les contraintes de précédence, ensuite il a utilisé une approche de recherche tabou pour trouver un ordonnancement de meilleure qualité.

A.Grigiriev, M.Holthuijsen et J.V. De Klundert [28] dans leur article publié en 2005, ont étudié un ensemble de problèmes d'ordonnancement sur une machine en présence des matières premières.

Les auteurs ont étudié plusieurs problèmes particuliers à contrainte de matières premières où les temps d'exécution sont unitaires ou égaux.

Deux grandes classes de problèmes sont étudiées, une première a pour objectif la minimisation du plus grand retard $L_{max,}$ une deuxième classe a été abordée pour la minimisation du makespan ;

Le tableau suivant résume l'ensemble de ces travaux :

Minimiser le L_{max}	**Minimiser le makespan C_{max}**
La matière première est dédiée à chaque tâche.	La matière première est dédiée à chaque tâche
Les temps d'exécution sont unitaires, en présence d'une seule matière première.	Les temps d'exécution sont égaux, en présence d'une seule matière première.
Les temps d'exécution sont égaux, en présence de deux matières premières	Plusieurs matières premières

Tableau II 2 travaux de A.Grigiriev, M.Holthuijsen et J.V. De Klundert [28]

Les travaux de Carlier et Rinnooy Kan [11] et Carlier [12] associent les ressources consommables aux ressources financières en raison des similarités observées dans les contraintes de disponibilité que ces deux types de ressources induisent. Carlier prouve par la suite que le problème d'ordonnancement de projet (RCPSP) avec des ressources financières et avec des contraintes de précédence arbitraire est polynomial si l'on ne considère pas de machine (ou autre ressource renouvelable) ; Carlier démontre aussi que ce problème devient NP-difficile si l'on admet la production et la consommation de ressources.

Patterson et al [41] proposent une procédure exacte pour résoudre le problème d'ordonnancement de projet avec des hypothèses d'interruption des opérations, des contraintes de précédence et de ressources qui peuvent être produites et consommées. Carlier et al [13] ont étudié le problème d'ordonnancement de projet où des unités de ressource peuvent être produites et consommées lors de l'exécution de certains événements. Les autres proposent un algorithme de liste pour minimiser la durée totale d'ordonnancement (makespan).

Cochand et al [15] considèrent des ressources consommables dont l'approvisionnement varie avec le temps. Les auteurs généralisent l'algorithme à deux phases pour machines parallèles, avec l'hypothèse de splitting et pour ce type de ressources.

Gafarov et Lazarev [24] étudient le problème d'ordonnancement de machine avec une ressource consommable dans le cadre de la minimisation de la somme des retards. Ils fournissent des résultats de complexité pour différentes variantes de ce problème.

Les travaux de H. Belouadah [5] ont traité le problème d'ordonnancement sur machine à contrainte de ressource non renouvelable pour minimiser la somme pondérée des dates de fin. Un algorithme optimal a été proposé pour le cas particulier du problème où les temps d'exécution sont unitaires ainsi que les quantités de la ressource. Le problème général a été résolu par la méthode Branch-and-Bound, en proposant quelques règles de dominance, ainsi qu'une borne inférieure basée sur la relaxation lagrangienne.

La thèse présentée par Adnen EL AMRAOUI [20], sous le titre « Ordonnancement cyclique Multi-Produits des lignes de traitement de surface : méthodes exacte et approchées » en 2011. Dans cette thèse l'auteur s'intéresse au fonctionnement cyclique multi-produits des ateliers de traitement de surface, et au problème d'ordonnancement associé, caractérisé par des contraintes fortes et atypiques dont certaines sont liées aux ressources consommables. L'auteur a opté pour une approche basée sur un modèle linéaire et une méthode de résolution arborescente de type séparation et évaluation dans un premier temps. Ensuite la proposition d'une heuristique dédiée au cas multi-produits étudié, qui est basée sur un algorithme de liste.

La thèse de Sadia AZEM [1] pour l'obtention du grade de docteur, « Ordonnancement des systèmes flexibles de production sous contraintes de disponibilité de ressource », en 2010, dans son article, l'auteur propose des modèles mathématiques pour le problème d'ordonnancement sous contrainte de disponibilité de ressources. Ces travaux se concentrent d'une manière largement détaillée en 3 parties :

- Une approche de modélisation mathématique.
- Des méthodes approchées.
- Une approche par génération de colonnes.

De nombreuses expérimentation ont été menées pour valider les méthodes proposées.

VI. Conclusion

De nombreuses méthodes et outils permettent de résoudre efficacement les problèmes d'ordonnancement et surtout les problèmes du job shop.

Ce chapitre est particulièrement accordé au job shop, nous avons mis en évidence les différentes résolutions graphiques connues (Gantt, PERT, Graphes conjonctifs-disjonctifs).

Les problèmes d'ordonnancement job shop sont classés NP-difficile et sont très délicat à résoudre vue leurs complexités et le grand nombre de variables à prendre en compte.

Dans ce qui va suivre nous allons proposer nos démarches de résolution pour la résolution du problème de job shop sous contrainte de ressources consommables, ainsi que plusieurs exemples pour tester la fonctionnalité de la fonction qui calcule Cmax, et une comparaison entre les heuristiques utilisés comme règles de priorité.

CHAPITRE III : LE PROBLEME JOB SHOP AVEC CONTRAINTE RESSOURCE CONSOMMABLE

I. Introduction

II. Présentation du problème

1. Description du système étudié
2. Description du problème
3. Notation
4. Formulation du problème

III. Démarche de résolution (algorithme proposé)

IV. Méthode choisie (critère de choix)

V. Application et étude de performance

VI. Conclusion

Chapitre III : Le problème job shop avec contrainte ressource consommable

I. Introduction :

La plupart des travaux dédiés à l'ordonnancement de production, les ressources sont supposées être disponible en continu pour effectuer des jobs. Ceci n'est pas toujours vrai : les différentes ressources aussi bien matérielles qu'humaines peuvent être indisponible pour diverses raisons. Les dates et les durées des périodes d'indisponibilité sont connue dans certains cas : congé du personnel, activités de maintenance des machines, ou absence des composants d'assemblage dans le cas de ressources consommables.

Dans ce chapitre, nous abordons l'étude du problème d'ordonnancement job shop sous contrainte de ressource consommable.

Notre travail sera dans un premier lieux de générer un algorithme compatible au problème, qui sera capable de calculer le makespan (C_{max}), tout en satisfaisant la contrainte de ressource.

Le code est modélisé sous le logicielle Matlab, compte tenu de l'efficacité de ce dernier et à la complexité d'implantation.

L'étape suivante consistera en l'application de plusieurs heuristiques (règles de priorités) qui sont : (SPT, LPT, SRC, LRC), cette approche a pour but de générer des solutions proches de l'optimal (satisfaisantes). Ceci afin de démontrer quelle heuristique nous donnera les meilleurs résultats en matière de Cmax, tout en respectant les contraintes.

II. Présentation du problème :

La résolution des problèmes rencontrés pour la gestion de leur poste de production, constitue une des principales préoccupations des industriels. En effet, comment réussir à produire le plus possible, le plus rapidement possible avec le moins de coût possible, tout en respectant toutes les contraintes liées au produit ?

Notre travail rentre dans ce cadre et consiste donc à développer des outils d'aide à l'ordonnancement permettant la minimisation du Makespan. Il s'agit donc, de prendre en considération les différentes contraintes relatives aux ressources et aux précédences entre tâches,

ainsi qu'au temps, et ceci pour réaliser une optimisation reposant sur l'objectif principal relatif au temps (le temps total de production).

1. Description du système étudié :

Le système étudié dans ce manuscrit est un job shop inspiré du système existant au laboratoire de productique MELT au pôle de technologie de l'université de Tlemcen.

Cet atelier est composé de 4 machines (M1, M2, M3, M4), comme figuré ci-dessous :

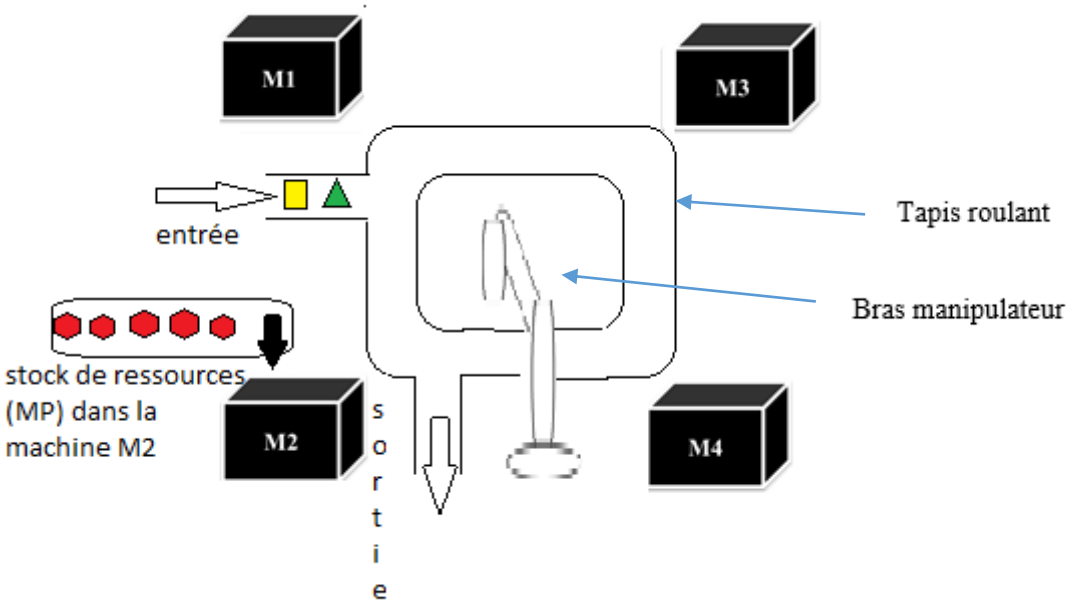

Figure III.1 Schéma du système étudié.

Les 4 machines sont réparties sous forme d'un carrée, tous relié par l'intermédiaire d'un tapis roulant pour le transport des produits entre les différentes machines, et des bras manipulateurs qui servent au déplacement des pièces du tapis roulant aux machines.

Notons que les produits transportés dans le tapis roulant sont en sens unidirectionnel, c-à-d qu'il n'y a pas de retour en arrière possible.

Dans notre cas, nous avons défini la machine 2 qui est la machine d'assemblage, et où on retrouve les ressources consommables.

2. Description du problème :

Le problème auquel nous nous intéressons plus particulièrement est le problème du job shop, sous la contrainte de ressources consommables. Ces ressources diminuent à chaque opération sur la machine dédiée, l'arrivé de ces ressources est supposé périodique (on reçoit 4 ressources chaque 3 minutes) comme le démontre la figure suivante.

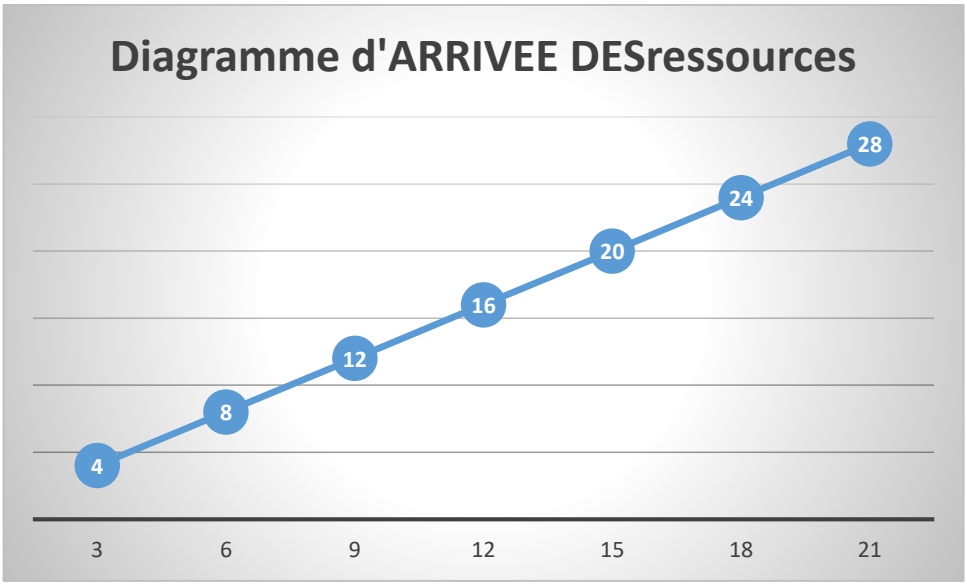

Figure III.2 Diagramme de ressources

Si le nombre de ressources disponible satisfait le nombre demandé par le job j, l'exécution de ce dernier est réalisable, sinon l'opération reste en attente jusqu'à ce que la quantité demandée soit satisfaite.

Et donc le problème consiste en l'ordonnancement d'un certain nombre de tâches sur les machines tel que :

a- Chaque job est constitué d'un ensemble connu d'opérations O.

b- Chaque job a sa gamme opératoire propre.

c- Chaque opération sur une machine k possède un temps de traitement TO_{ij}.

d- Chaque job nécessite un nombre de ressources RCJ_j connu à priori (sur la machine 2).

L'objectif est de trouver le meilleur séquencement des tâches tout en minimisant la durée totale d'exécution (makespan), en tenant compte de la disponibilité des ressources.

3. Notation

Ici, nous allons définir les différentes notations exprimées dans la formulation du problème ainsi que l'algorithme proposé.

k	désigne la $K_{ième}$ machine.
P_{ik}	positions « i » libres dans la machine k
j	Nombre de jobs
NO_k	Nombre d'opérations dans la machine « k »
O_{jk}	Opération du job j dans la machine « k »
TDJ_j	Temps de disponibilité des jobs « j »
TDM_k	Temps de disponibilité des machines « k »
RCJ_j	Ressource consommable demandé par le job « j »
TO_{ij}	Temps opératoires du job « j » dans la machine « k »
O_{ij}	Opération du job « j » à la machine « k »
t_j	Date de fin d'opération
t_i	Date de début d'opération
Y_{ij}	Yi, représente l'allocation des composants à l'opération i
M	Ensemble des machines
Ek	Ensemble des pairs d'opérations sur la machine k
Bk	Ensemble d'opérations qui attendent la ressource h
A	Ensemble des pairs d'opérations contraintes par des relations de précédences
N	Ensemble des opérations
Fk	Ensemble de ressources consommables
a_n	le temps d'arrivé des ressources

4. Formulation du problème

Le problème est défini par **J4 /RC/Cmax** selon les notations les plus utilisés en ordonnancement.

On considère le problème d'ordonnancement de 3 jobs (j=3) sur 4 machines (k=4), une machine ne peut exécuter qu'une seule tâche *i* à la fois. Les tâches possèdent des temps d'exécutions TO_{ij}.

Une quantité RCJ_j de ressources consommables (Matière Première) est sollicitée sur la machine k=2 pour que les tâches soient exécutées. L'arrivée de ressources est supposée périodique. Et donc chaque opération O_{ij} requiert les ressources en quantité déterminée RCJ_j. Le nombre de ressources au départ est de 0.

Le problème consiste à trouver un ordonnancement qui minimise le temps total d'exécution tout en satisfaisant la contrainte de disponibilité de ressources et les contraintes de précédence.

On prend en considération *les contraintes* suivantes :

- Une machine ne peut traiter qu'un job à la fois.
- Pas de préemption possible.
- Un job ne peut être exécuté sur la machine 2 si le nombre de composants est insuffisant.
- Un produit ne peut être exécuté que sur une seule machine à la fois.
- Un produit ne peut exécuter sur une machine plus d'une fois (pas de recirculation).
- Les produits sont disponibles à l'instant t=0.
- Chaque machine possède une file d'attente de capacité illimité.
- La machine m2 possède un stock de composants (MP) de capacité illimitée.

D'après les informations présentées, nous pouvons établir une formulation du modèle comme suit :

La fonction objective :

$\text{Min} \quad \sum_{n=0}^{k} C_{ij}$ tel que : $(i,j) \in A$----------(1)

Les contraintes :

$t_j - t_i \geq TO_{ij}$, $(i, j) \in A$, ----------(2)

$t_i \geq 0$, $i \in N \cup R$ ----------(3)

$t_j - t_i \geq p_i \vee t_i - t_j \geq p_j$, $(i, j) \in Ek$, ----------(4)

$\sum Y_{ij} \leq RCJ_j$ $i \in Bk, j \in Fk$, ----------(5)

$t_i \geq a_n$ $i \in Bk, n \in Fk, Y_i = 1$ ----------(6)

- L'équation (1) est la fonction objective du problème, qui désigne la minimisation du temps totale d'exécution Cmax (makespan).
- Les équations (2) et (3) représentent les relations de précédence entre les opérations.
- (2) : la date de fin de l'opération – la date de début de l'opération est supérieur ou égale au temps opératoire de l'opération i.
- (3) : la date de début de l'opération i est supérieure ou égale à 0.
- L'équation (4) représente la contrainte de capacité et qui stipule que les opérations appartenant à la même machine ne peuvent être exécutés simultanément.
- Les équations (5) et (6) sont des contraintes qui désignent l'allocation des ressources et garantissent que les opérations ne peuvent pas procéder avant l'arrivée des composants nécessaires.
- (5) s'assure que toutes les exigences matérielles sont remplies.
- (6) la date de début de l'opération est supérieur à l'arrivé des composants, elle garantit qu'une opération ne peut démarrer qu'après l'arrivé de toutes les ressources nécessaires.

A : représente l'ensemble des pairs d'opération O_{ij} contraints par une relation de précédence, Pi, ti, qi représentent successivement le temps opératoire, date de début, et la quantité demandée de composants pour l'opération i. Yi, représente l'allocation des composants à l'opération i.

Dans la section suivante nous allons aborder la résolution de ce problème par une méthode de recherche locale basée sur les heuristiques ; ensuite, en faisant des analyses sur un ensemble d'exemples, avant de clôturer le chapitre avec une étude expérimentale des heuristiques élaborées.

III. Impacte des ressources consommables

Dans ce qui suit, nous allons proposer un petit exemple (exemple 1.1) composé de 3 jobs qui vont être réalisés dans notre système afin de présenter le calcul du Cmax dans les deux cas respectivement ; sans ressources consommables et avec ressources consommables.

Les temps d'exécution des opérations de chaque job sont représentés dans le tableau suivant

Tableau III 1 Temps opératoire de l'exemple 1.1

Job/Machine	*M1*	*M2*	*M3*	*M4*
J1	5	8	-	4
J2	2	11	5	12
J3	10	15	7	-

Le séquencement des tâches sur l'ensemble des machines est défini comme suit :

J1 : M2→M1→M4

J2 : M3→M4→M1→M2

J3 : M1→M3→M2

La quantité demandée de ressources pour chaque job : RCJ :

J1 : 3

J2 : 25

J3 : 19

L'arrivé des composants PR: chaque 3mn

La quantité de composants pour chaque arrivé NR=4

a. Solution sans ressources consommables

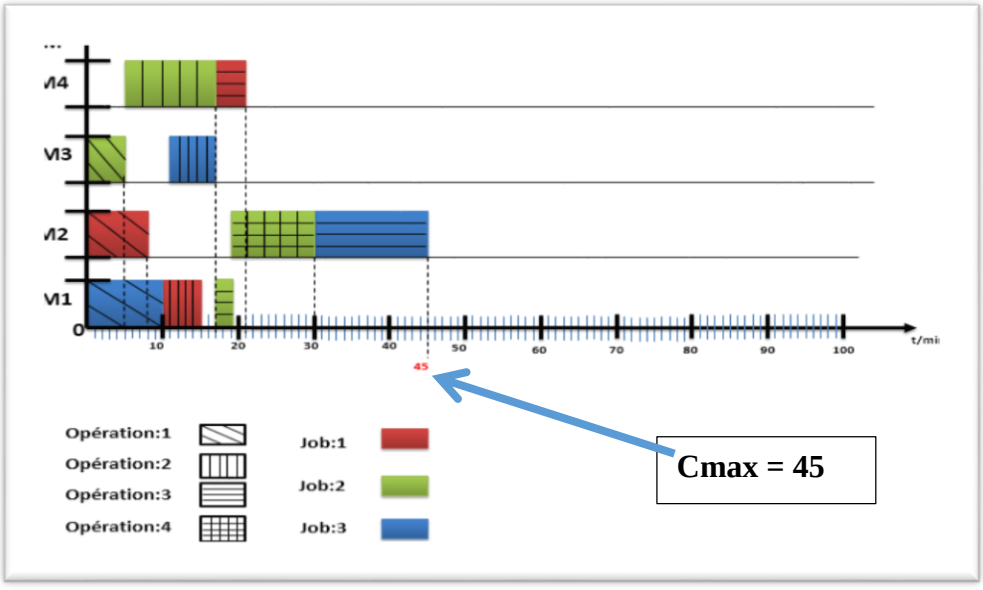

Figure III.3 Diagramme de Gantt –Exemple 1.1- sans ressource

b. Solution avec ressources consommables

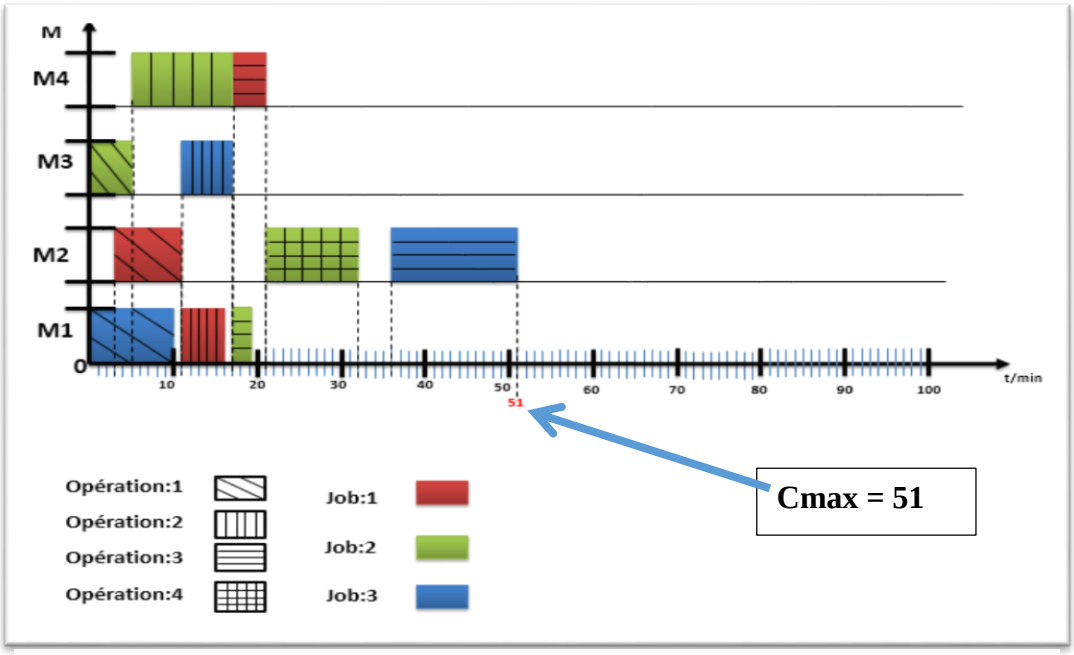

Figure III.4 Diagramme de Gantt –Exemple 1.1- avec ressource

Analyses des résultats :

Nous constatons d'après les deux diagrammes établis que l'existence de la ressource consommable influe directement sur le temps total d'exécution des tâches qui augmente de 6mn.

IV. Méthode choisie (critère de choix)

Parmi les nombreuses méthodes approchées pour résoudre le job shop, *les méthodes sérielles* sont très souvent utilisées car elles sont peu coûteuses. Cependant la qualité des solutions obtenues (en termes de makespan) est très variable. *La méthode Shifting Bottleneck Procedure* est plus performante que les méthodes sérielles mais son coût (en termes du temps d'exécution) est beaucoup plus élevé [19].

D'après notre description du Problème, et les méthodes de résolution dédiée à ce type de problème, nous avons choisis des méthodes sérielles (ou des heuristiques) pour notre cas. Les heuristiques Proposées sont :

- LPT : Longest Processing Time : la tâche ayant le temps opératoire le plus long est traitée.
- SPT : Shortest Processing Time : la tâche ayant le temps opératoire le plus court est traitée en premier.

- LRC : Longest Resource Consumable : la tâche ayant le nombre de ressources le plus élevé est traitée en premier.
- SRC : Shortest Resource Consumable : la tâche ayant le nombre de ressources le plus bas est traitée en premier.

La fonction minimisant le makespan avec la contrainte de ressources consommables est défini dans l'algorithme suivant :

Tant que $k \leq 4$

Pour $j := 1 \ldots 3$

Si $P_{ik} = O_{jk}$

Temps de départ = max (TDJ_j, TDM_k)

Si k=2

RC

Si $RC < RCJ_j$

Tant que $RC < RCJ_j$

$RC = RC + NR$

Fin de tant que

Fin de si

$RC = RC - RCJ$

Fin de si

TDJ_j

TDM_k

Fin de si

Fin de tant que

Figure III.5 Algorithme proposé pour la fonction Cmax avec la disponibilité des ressources

L'algorithme proposée représente la fonction à minimiser min Cmax, il est programmé sous le logiciel de programmation mathématique MATLAB R, qui est l'outils de modélisation par excellence.

Cette fonction permet de calculer la durée d'exécution totale des jobs Cmax (makespan) d'un problème donné, tout en respectant les contraintes de disponibilité des machines et disponibilités des jobs, les contraintes de précédences entre tâches dues à la gamme opératoire, et surtout la contrainte de disponibilité de ressources requises par un job pour son exécution.

V. Application et étude de performance

Après l'explication du principe de fonctionnement de notre algorithme, et après avoir vu le principe des méthodes proposées (LPT, SPT, SRC, LRC). Nous allons dans un premier temps tester l'efficacité de la fonction, pour ce, le même problème dans l'exemple 1 sera appliqué.

En second, nous allons appliquer les règles de priorités citées en haut à notre fonction, et générer plusieurs exemples de paramètres différents, afin d'évaluer et de définir quel est l'approche qui nous donne les meilleurs résultats.

*Remarque*1 :

L'implantation de ces heuristiques s'est également faite sous MATLAB.

Remarque2 :
Afin de simplifier le travail, dans les heuristiques LPT et SPT on calcule la somme des temps opératoire pour chaque job et puis on applique les règles. Et donc le job ayant les temps opératoires les plus long ou court (selon la règle) sera le premier à s'exécuter sur les 4 machines.
Remarque3 :
Le fonctionnement des heuristiques se fait sur l'ensemble du système et non pas sur les machines.
Chaque exemple possède sa propre *gamme opératoire*, son ensemble de *temps opératoires*, ainsi que *le nombre de quantités de ressources requises* par les jobs.

Exemple 1.2 :

En reprend les données de l'exemple 1.1 on va l'appliquer sur notre algorithme ;

```
rcj =

     3     25     19


Sol =

    31     11     21     22
    12     24     32     13
    23     33      0      0


TDJ =

    21     32     51


TDM =

    19     51     17     21

Cmax/Makespan=
    51
```

Nous constatons que la fonction donne le même résultat pour le Cmax (**Cmax=51)** que le résultat obtenu avec le diagramme de Gantt dans Exemple 1.1, ce qui justifie la qualité de la solution obtenue.

```
rcj =

     0      0      0


Sol =

    31     11     21     22
    12     24     32     13
    23     33      0      0


TDJ =

    21     30     45


TDM =

    19     45     17     21

Cmax/Makespan=
    45
```

Tandis que dans l'exemple où nous négligeons la ressource consommable, on obtient aussi le même résultat du Cmax (**Cmax=45)** établi par le diagramme de Gantt.

Figure 7 III.6 Exemple 1 (sans et avec ressource)

Passons maintenant à l'application des ***heuristiques*** :

Pour les exemples qui suivent, les ressources sont supposées périodique, nous avons défini que les ressources consommables arrivent par lot de 3 ressources toutes les 5 minutes.

Exemple 2 : Pour cet exemple, on va définir les paramètres comme représentés dans le tableau suivant :

Tableau III 2 Données exemple 2

Jobs	Machines	M1	M2	M3	M4
To=	J1	5	9	2	8
	J2	11	4	0	7
	J3	0	6	10	12
Go=	J1	1	2	3	4
	J2	4	3	2	1
	J3	3	1	4	2
RCJ=		J1=15	J2=8	J3=25	

Où **To** représente les temps opératoires.

Et **Go** représente la gamme opératoire pour chaque job.

RCJ la quantité de ressources demandées par les trois jobs.

Résultat avec SPT :

TDJ = [59 26 108]

TDM= [98 86 96 109]

C_{max} = max(TDJ) = max(TDM)

C_{max} = 108

La matrice « Sol » contient l'ensemble des opérations O_{ij} ordonnancer selon la règle SPT, tel que le « i » est le numéro du job, et le « j » représente l'ordre des opérations des jobs

TDJ est le temps de disponibilité des jobs

TDM est le temps de disponibilité des machines

S : est la somme des temps opératoires de chaque job ordonnancé selon la règle.

In : est le séquencement des jobs sur les machines.

```
S =

    22    24    28


in =

     2     1     3


Sol =

    24    23    22    21
    11    12    13    14
    33    31    34    32


TDJ =

    59    26   108


TDM =

    16    86    96   108

Cmax/Makespan=
   108
```

Figure 8 III.7 Exemple 2 (SPT)

Résulta avec LPT:

TDJ = [89 96 73]

TDM= [16 84 84 96]

C_{max} = max(TDJ) = max(TDM)

C_{max} = 96

```
S =

    28    24    22


in =

     3     1     2


Sol =

    33    31    34    32
    11    12    13    14
    24    23    22    21


TDJ =

    89    96    73


TDM =

    16    84    84    96

Cmax/Makespan=
    96
```

Figure 9 III.8 Exemple 2 (LPT)

SRC

TDJ = [81 91 73]

TDM= [16 84 84 91]

C_{max} = max(TDJ) = max(TDM)

C_{max} = 91

```
rcj =

     8    15    25


in =

     2     1     3


Sol =

    31    32    33    34
    11    12    13    14
    21    22    23    24


TDJ =

    81    91    73


TDM =

    16    84    84    91

Cmax/Makespan=
    91
```

Figure 11 III.9 Exemple 2 (SRC)

LRC

TDJ = [74 91 43]

TDM= [16 84 84 91]

C_{max} = 91

```
rcj =

    25    15     8


in =

     3     1     2


Sol =

    31    32    33    34
    11    12    13    14
    21    22    23    24


TDJ =

    74    91    43


TDM =

    16    84    84    91

Cmax/Makespan=
    91
```

Figure 10 III.10 Exemple 2 (LRC)

Pour cet exemple nous constatons que les règles SRC et LRC nous offrent le même résultat, Cmax= 91, tandis que SPT et LPT donne des résultats moins satisfaisants

Exemple 3 :

Tableau III 3 Données exemple 3

Jobs	Machines	M1	M2	M3	M4
To=	J1	9	4	0	11
	J2	11	6	5	10
	J3	7	8	9	14
Go=	J1	2	4	1	3
	J2	1	3	4	2
	J3	4	2	3	1
RCJ=		J1=07	J2=12	J3=16	

SPT

```
S =

    24    32    38


in =

     1     2     3


Sol =

    12    14    11    13
    21    23    24    22
    34    32    33    31


TDJ =

    30    56    91


TDM =

    27    68    77    91

Cmax/Makespan=
    91
```

Figure 13 III.11 Exemple 3 (SPT)

LPT

```
S =

    38    32    24


in =

     3     2     1


Sol =

    34    32    33    31
    21    23    24    22
    12    14    11    13


TDJ =

    82    71    61


TDM =

    27    64    64    82

Cmax/Makespan=
    82
```

Figure 12 III.12 Exemple 3 (LPT)

SRC

```
rcj =

     7    12    16


in =

     1     2     3


Sol =

    31    32    33    34
    11    12    13    14
    21    22    23    24


TDJ =

    72    82    61


TDM =

    27    66    71    82

Cmax/Makespan=
    82
```

Figure 15 III.13 Exemple 3 (SRC)

LRC

```
rcj =

    16    12     7


in =

     3     2     1


Sol =

    31    32    33    34
    11    12    13    14
    21    22    23    24


TDJ =

    57    81    46


TDM =

    27    66    71    81

Cmax/Makespan=
    81
```

Figure 14 III.14 Exemple 3 (LRC)

Pour ce cas, SRC et LPT offrent des résultats similaires, pourtant la règle LPT ne tient pas en compte les ressources. Cependant LRC offre un meilleur résultat que les autres avec Cmax= 81

Exemple 4 :

Tableau III 4 Données exemple 4

Jobs	Machines	M1	M2	M3	M4
To=	J1	13	0	16	11
	J2	16	9	14	0
	J3	0	18	18	1
Go=	J1	3	2	1	4
	J2	1	2	4	3
	J3	4	1	3	2
RCJ=		J1=16	J2=24	J3=11	

SPT

```
S =

    37    39    40

in =

     3     2     1

Sol =

    34    31    33    32
    21    22    24    23
    13    12    11    14

TDJ =

   112    83    57

TDM =

    29    85   101   112

Cmax/Makespan=
   112
```

Figure III.15 Exemple 4 (SPT)

LPT

```
S =

    40    39    37

in =

     1     2     3

Sol =

    13    12    11    14
    21    22    24    23
    34    31    33    32

TDJ =

    57    93   122

TDM =

    29   103   121   122

Cmax/Makespan=
   122
```

Figure III.16 Exemple 4 (LPT)

SRC

```
rcj =

    11    16    24


in =

     3     1     2


Sol =

    31    32    33    34
    11    12    13    14
    21    22    23    24


TDJ =

   103   108    77


TDM =

    29    94   108   108

Cmax/Makespan=
   108
```

Figure III.17 Exemple 4 (SRC)

LRC

```
rcj =

    24    16    11


in =

     2     1     3


Sol =

    31    32    33    34
    11    12    13    14
    21    22    23    24


TDJ =

    87   108    57


TDM =

    29    94   108   108

Cmax/Makespan=
   108
```

Figure III.18 Exemple 4 (LRC)

Les règles SRC et LRC donnent les meilleurs résultats avec un Cmax= 108 pour les deux méthodes, tandis que SPT et LPT donnent des résultats un peu plus élevés.

Exemple 5 :

Tableau III 5 Données exemple 5

Jobs	Machines	M1	M2	M3	M4
To=	J1	2	1	4	5
	J2	4	2	1	7
	J3	9	5	4	6
Go=	J1	1	4	2	3
	J2	4	3	2	1
	J3	1	4	3	2
RCJ=		J1=0	J2=15	J3=24	

SPT

```
S =

    12    14    24


in =

     1     2     3


Sol =

    11    14    12    13
    24    23    22    21
    31    34    33    32


TDJ =

    12    35    80


TDM =

    15    70    74    80

Cmax/Makespan=
    80
```

Figure III.19 Exemple 5 (SPT)

LPT

```
S =

    24    14    12


in =

     3     2     1


Sol =

    31    34    33    32
    24    23    22    21
    11    14    12    13


TDJ =

    80    75    55


TDM =

    15    68    72    80

Cmax/Makespan=
    80
```

Figure III.20 Exemple 5 (LPT)

SRC

```
in =

     1     2     3


rcj =

     0    15    24


Sol =

    31    32    33    34
    11    12    13    14
    21    22    23    24


TDJ =

    60    75    55


TDM =

    15    67    68    75

Cmax/Makespan=
    75
```

Figure III.21 Exemple 5 (SRC)

LRC

```
rcj =

    24    15     0


in =

     3     2     1


Sol =

    31    32    33    34
    11    12    13    14
    21    22    23    24


TDJ =

    50    75    24


TDM =

    15    67    68    75

Cmax/Makespan=
    75
```

Figure III.22 Exemple 5 (LRC)

SRC et LRC proposent un makespan relativement plus bas que le SPT et LPT avec Cmax= 75.

Exemple 6 :

Tableau III 6 Données exemple 6

Jobs	Machines	M1	M2	M3	M4
To=	J1	29	35	33	26
	J2	36	30	29	31
	J3	0	28	22	32
Go=	J1	1	3	4	2
	J2	4	2	1	3
	J3	3	1	2	4
RCJ=		J1=33	J2=40	J3=29	

SPT

```
S =

    82   123   126


in =

     3     1     2


Sol =

    33    31    32    34
    11    13    14    12
    24    22    21    23


TDJ =

   146   180   106


TDM =

    65   117   149   180


Cmax/Makespan=
   180
```

Figure III.23 Exemple 6 (SPT)

LPT

```
S =

   126   123    82


in =

     2     1     3


Sol =

    24    22    21    23
    11    13    14    12
    33    31    32    34


TDJ =

   219   160   252


TDM =

    65   198   220   252


Cmax/Makespan=
   252
```

Figure III.24 Exemple 6 (LPT)

SRC

```
rcj =

    29    33    40


in =

     3     1     2


Sol =

    31    32    33    34
    11    12    13    14
    21    22    23    24


TDJ =

   152   186   112


TDM =

    65   123   155   186

Cmax/Makespan=
   186
```

Figure III.25 Exemple 6 (SRC)

LRC

```
rcj =

    40    33    29


in =

     2     1     3


Sol =

    31    32    33    34
    11    12    13    14
    21    22    23    24


TDJ =

   209   260   132


TDM =

    65   200   229   260

Cmax/Makespan=
   260
```

Figure III.26 Exemple 6 (LRC)

Pour cet exemple, la règle qui donne le meilleur temps d'exécution est SPT avec Cmax= 180, suivi de SRC avec Cmax=186. Tandis que LRC et LPT offrent un résultat beaucoup plus élevé.

Analyse et interprétation des résultats :

D'après les exemples testés, on remarque que les règles SRC et LRC nous donnent presque à chaque fois les meilleurs résultats.

Dans 4 cas sur 5 (Exemple 2 → exemple 5), LRC fournit la meilleure solution, souvent pour des problèmes où les temps opératoires sont moyens (entre 1 et 18), et des quantités de ressources RCJ allant jusqu'à 24.

La règle SRC fournit les mêmes résultats que LRC dans 3 cas seulement (Ex 2, Ex 3 et Ex 4), pour les deux autres exemples, les solutions sont satisfaisantes par rapport aux meilleurs résultats trouvé.

Pour un exemple de taille plus grande (exemple 5), avec des temps opératoires et quantités de ressources largement supérieures aux autres, la règle SPT nous donne le meilleur résultat en matière de Cmax.

Conclusion

Dans ce chapitre et d'après les tests et les résultats, nous pouvons déduire que pour notre problème qui est le job shop sous contrainte de ressources consommables, l'utilisation de la règle LRC et SRC aboutissent à de meilleurs résultats pour des temps opératoires relativement courts. Alors que la règle SPT est plus adapté pour des problèmes où les temps de traitements des opérations sont plus grand.

Conclusion générale

De nos jours, on trouve les problèmes d'ordonnancement dans divers secteurs de l'économie, un problème d'ordonnancement consiste donc à allouer dans le temps des tâches aux ressources existantes, tout en satisfaisant certains critères.

L'un des problèmes d'ordonnancement les plus étudiés est le problème du job shop, car il est extrêmement complexe, et nécessite une grande connaissance du domaine. Le problème du job shop est classé parmi les problèmes combinatoires difficiles au sens fort (NP-difficile) à cause du grand nombre de variables pris en considération. Le nombre de solutions possible pour ce type de problème accroit exponentiellement avec la taille du problème.

L'utilisation de méthodes approchés comme les heuristiques et métaheuristiques est plus que nécessaire pour la résolution de ce type de problèmes et pour s'approcher le plus de la solution optimale.

La prise en considération de la contrainte de ressource consommable a fait que le problème a été plus difficile à saisir et à modéliser.

Dans un premier temps, nous avons abordé l'ordonnancement de manière générale ainsi que les différents problèmes existants, en second lieu nous nous somme intéresser au problème du job shop, aux différents modes de résolution de ce dernier, ainsi qu'un état de l'art sur le job shop et le job shop sous contrainte de ressource consommables. Enfin, nous mettons en évidence la démarche adoptée pour la résolution du problème de job shop sous contrainte de ressources consommables.

La démarche mis au point consiste tout d'abord en la création et modélisation sous MATLAB d'une fonction qui permet de calculer le makespan sur 4 machines et 3 jobs, tout en tenant compte des différentes contraintes associées à notre problème. Puis à l'application des règles (SPT, LPT, SRC, LRC) pour trouver laquelle nous permet de minimiser le makespan.

Le résultat obtenu concernant la fonction est très satisfaisant, puisque le programme nous permet de calculer le Cmax exact. Pour ce qui est des règles de priorités, les exemples traités nous montrent l'efficacité de la règle LRC et SRC qui s'approchaient le plus souvent de la solution optimale, tandis que la règle SPT prouve son efficacité sur des problèmes avec des temps opératoires largement plus élevées.

Comme perspectives pour d'éventuels travaux sur ce sujet, nous proposons l'utilisation d'autres approches pour la résolution de ce problème notamment :

L'application de la fonction sur un espace de recherche globale avec des algorithmes génétiques, cela permettra d'obtenir un nombre considérable de solutions, et de déterminer la solution optimale. Puis procéder avec d'autres métaheuristiques tel que la recherche Tabou et les colonies de fourmis.

La modélisation du système en graphes, auxquels seront appliqués des algorithmes d'optimisation de graphes (Belmane, Djikstra…) est aussi à prendre en considération.

Une élévation de la complexité du problème pourrait être envisagé, en augmentant le nombre de machines et de jobs, et en l'ajout de nouvelles contraintes tel que le transport.

Ces propositions pourraient être envisagées sur le long terme, cela permettra de mieux appréhender le problème du job shop et d'avoir des réponses précises sur le domaine.

Référence bibliographiques :

[1] AZEM S, (2010), Ordonnancement des systèmes flexibles de production sous contraintes de disponibilité des ressources. Thèse pour obtenir le grade de Docteur, Ecole Nationale Supérieure des Mines de Saint-Etienne.

[2] BADR E, (2016), Problèmes d'ordonnancement de projet et métaheuristiques. Thèse, Université BEN ABDELLAH.

[3] BAKER K.R, (1974), Introduction to Sequencing and Scheduling. Wiley, New York.

[4] BEBOUZID SITAYEB F (2005), Contribution à l'étude de la performance et de la robustesse des ordonnancements conjoints production/maintenance- cas du flow shop. Thèse pour obtenir le grade de docteur de l'université de FRANCHE-COMTE.

[5] BELOUADAH H, (2004), Single Machine Scheduling Problem with non-renewable Resources, Working Paper, Université Med Boudiaf de M'sila.

[6] BOUKEF H, (2009), Sur l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharmaceutique : optimisation par algorithmes génétiques et essaims particulaires. Thèse, Université de Lille.

[7] BRINZEI N, (1998), L'état de l'art sur la conduite de systèmes flexibles de production. Rapport, Université Polytechnique de Timisoara, 1998.

[8] CARINE R, (2007), Etude du problème de job shop avec un convoyeur. Thèse à l'université du QUEBEC.

[9] CARLIER J & CHRETIENNE. P, Problèmes d'ordonnancement, modélisation, complexité, algorithmes, MASSON, ISBN 2-225-81275-6.

[10] CARLIER J et CHRÉTIENNE P et GIRAULT C, (1984), Modelling scheduling problems with Petri nets. Advanced studies in Petri nets. Lecture notes in Computer Science, Springer Verlag, Paris.

[11] CARLIER J et RINNOOY K, (1982), Scheduling subject to nonrenewable resource constraints. Operation Research Letters.

[12] CARLIER J, (1984) Problème d'ordonnancement à contrainte de ressources : Algorithmes et complexité. Thèse, Université Pierre et Marie Curie (Paris VI).

[13] CARLIER J, MOUKRIM A et XU H, (2009), The project scheduling problem with production and consumption of resources: A list-scheduling based algorithm. Discrete Applied Mathematics, 157(17):3631-3642.

[14] CAUMOND A, (2006), Le problème de job shop avec contraintes modélisation et optimisation. Thèse de doctorat, université Blaise Pascal.

[15] COCHAND M, DE WERRA D et SLOWINSKI R, (1989), Preemptive scheduling with staircase and pricewise linear resource availability. Methods and Models of Operations Research, 33:297-313.

[16] DHOUIBI H, (2005), Utilisation des réseaux de Petri à intervalles pour la régulation d'une qualité : Application à une manufacture de tabac. Thèse de Doctorat, Université des Sciences et Technologies de Lille.

[17] DUCHESNE A, (2013), Ordonnancement de projet avec contraintes de Ressources dans un contexte incertain. Thèse, Université de MONTREAL.

[18] DURAND N, ALLIOT J.M et NOAILLES J, (1994), Algorithmes génétiques : un croisement pour les problèmes partiellement séparables. Journées Evolution Artificielle Francophones, JEAF, Toulouse.

[19] DUVIVIER D, (2000), Etude de l'hybridation des méta-heuristiques : application à un problème d'ordonnancement de type jobshop. Thèse, Université du Littorale côte d'opale.

[20] EL AMRAOUI A, (2011), Ordonnancement cyclique multi-produits des lignes de traitement de surfaces : méthodes exactes et approchées. Thèse, Université de Belfort - Montbeliard.

[21] ESQUIROL P. et LOPEZ P, (1999). L'ordonnancement. Economica.

[22] FABRE F (2009), Conduite orientée ordonnancement d'un simulateur dynamique hybride : application aux procédés discontinus. Thèse, université de TOULOUSE, France.

[23] FONDREVELLE J, (2005), Résolution exacte de problèmes d'ordonnancement de type flow shop de permutation en présence de contraintes d'écarts temporelles entre opérations. Thèse de doctorat, Université de Lorraine.

[24] GAFAROV E.R et LAZAREV A.A, (2010), Single machine scheduling with a non-renewable financial resource. Working paper.

[25] GEORGES K, TONGAMBOU W, (1999), Effet des règles d'ordonnancement sur la performance d'une ligne de production en JAT. Thèse présentée à l'université du Quebec à Trois Rivières.

[26] GIARD V, (1999), Production et techniques quantitatives appliquées à la gestion. Collection Gestion Série, Paris.

[27] GORINE A, (2011), Ordonnancement des systèmes flexibles avec contrainte de blocage. Thèse en vue de l'obtention du grade de doctorat, Université Paul-Verlaine de Metz.

[28] GRIGORIEV A, HOLTHUIJSEN M and DE KLUNDERT J.V, (2005), Basic Scheduling Problems with Raw Materials Constraints, Thèse, Naval Research Logistics.

[29] IYER S.K et SAXENA B, (2004), Improved genetic algorithm for the permutation flow shop scheduling problem. Computers and Operations Research.

[30] JANIAK A, (2000), Single Machine Scheduling with Nonlinear Resource Dependencies of Release Times, Article, AMS.

[31] K.M.J. De B, (2001), Integrating Purchase and Production Planning, thèse, Université of Eindhoven.

[32] KAI-PEI CHEN N, (2007), Assembly job shop scheduling problems with component availability constraints. Thèse, Université de OKLAHOMA.

[33] KEBABLA M, (2008) Utilisation des Métaheuristiques pour l'ordonnancement des ateliers de type job shop, Thèse présentée en vue de l'obtention du diplôme de magistère, Université de Batna.

[34] LARABI M, (2010), Le problème de job-shop avec transport : modélisation et optimisation. Thèse pour l'obtention de doctorat, Université Blaise Pascal – Clermont Ferrand.

[35] LOPEZ P, (2003), Approche par contraintes des problèmes d'ordonnancement et d'affectation : structures temporelles et mécanismes de propagation. Thèse, Institut polytechnique de Toulouse, France.

[36] MEHENNI T, (2006), Utilisation des métaheuristiques pour résoudre un problème d'ordonnancement sur machine à contrainte de ressource non renouvelable. Thèse, université MOHAMED BOUDIAF DE M'SILA.

[37] MERHOUM K et DJEGHABA M, (2015), Algorithme génétique pour le problème d'ordonnancement de type job-shop. Article publié.

[38] MOUHOUB N, (2011), Algorithmes de construction de graphes dans les problèmes d'ordonnancement de projet. Thèse, Université Ferhat Abbas – Setif.

[39] MUTH J.F and L.T.G, (1963), Industrial sheduling. Emglewood Cliffs, Prentice hall.

[40] OURARI S, (2011), de l'ordonnancement déterministe à l'ordonnancement distribué sous incertitudes. Thèse, Université Pau Sabatier – Toulouse 3.

[41] PATTERSON J.H, SLOWINSKI R, TALBOT F.B et WEGLARZ J, (1989), An algorithm for a general class of precedence and resource constrained scheduling problems. In Advances in Project Scheduling, Amsterdam.

[42] PINEDO M, (1995), Scheduling - Theory, Algorithms and Systems, Prentice Hall.

[43] PREUX P, (2000), Etude de l'hybridation des méta-heuristiques, application à un problème d'ordonnancement de type jobshop. Thèse en vue de l'obtention du grade de doctorat, université du Littoral Côte d'opale.

[44] QIANJUN X, (2001), Introduction to Job Shop Scheduling Problem.

[45] RINNOOY k, (1976), Machine scheduling problems: classification, complexity and computations. Martinus Nihoff, The Hague.

[46] ROY B et BOUYSSON D, (1993) Aide multi-critères à la décision : Méthodes et cas. Collection Gestion Série : Production et technologie quantitatives appliquées à la gestion. Edition Economica, Paris.

[47] SAKAROVITCH M, (1984), Programmation Discrète. Edition Hermann, Paris.

[48] VELAGA P, (2016), Scientific Management of High- Variety, Complex Production in Job Shops. Article publié.

[49] XIAO Y, (2013), Modélisation et simulation des systèmes de production : une approche orientées- objets. Thèse de doctorat, INSA de Lyon.

ANNEXES

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).

Approximatives (suite)	Les heuristiques constructives (suite)	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).
		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).
	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).

Résumé

La majeure partie des travaux sur les problèmes d'ordonnancement se dirige vers le contexte où les ressources (consommables) sont disponibles en permanence. Ce qui en réalité n'est pas toujours le cas. Nous nous plaçons dans le contexte d'indisponibilités des ressources consommable (matière première); le problème que nous allons traiter est donc du type job shop avec contrainte de ressource consommable de tel sorte que les tâches peuvent être en attente par manque de quantité de ressources. L'intégration de ces contraintes rend les problèmes d'ordonnancement nettement plus difficiles à résoudre. Dans ce sens, et pour écarter notre problème, on ne prend en compte que les contraintes de ressources, les contraintes de disponibilité et de précédence. Le modèle inspiré du laboratoire MELT de l'université de Tlemcen contient 4 machines et 3 jobs ; tandis que, les ressources sont consommées seulement sur une machine dédiée. Notre mémoire vise à proposer une fonction modélisé sous Matlab, qui permet de calculer le makespan avec les contraintes proposées. Les heuristiques : SPT, LPT, SRC, LRC sont utilisés afin d'évaluer leurs performances à notre problème. De nombreuses expérimentations ont été menées pour valider les méthodes d'une part, et la fonction de calcule d'autre part.

Abstract

Much of the work on scheduling problems is moving towards the context where resources (consumables) are permanently available. This is not always the case. We place ourselves in the context of unavailability of resources consumable (raw material); The problem we are going to deal with is therefore of the job shop type with consumable resource constraint so that the tasks may be waiting for lack of quantity of resources. The integration of these constraints makes scheduling problems much more difficult to solve. In this sense, and to rule out our problem, we only take into account resource constraints, availability constraints and precedence constraints.

The model inspired by the MELT laboratory of the university of Tlemcen contains 4 machines and 3 jobs; While, resources are consumed only on a dedicated machine. In this thesis, we propose a function modeled under Matlab, which makes it possible to compute the makespan with the proposed constraints. Heuristics: SPT, LPT, SRC, LRC are used to evaluate their performance to our problem. Numerous experiments have been carried out to validate the methods on the one hand, and the function of calculates on the other hand.

ملخص

معظم الأعمال الرئيسية على مشاكل الجدولة يذهب نحو الموارد الاستهلاكية المتوفرة بشكل دائم. عكس ما يحويه الواقع. في هذا السياق، وضعنا أنفسنا في حالة عدم توافر هذه المواد (المواد الخام)، المشكلة أننا سنتعامل مع المشكل المرتبط بـ <<job−shop>> الملحق بالقيود التي فرضنها على الموارد الاستهلاكية مثال على ذلك أن المهام التي يمكن أن تكون في الانتظار بعدم وجود كمية الموارد، دمجنا لهذه القيود جعل مشاكل الجدولة أكثر صعوبة للحل و على ذلك و من أجل نشر مشكلتنا، أخذنا القيود المفروضة على الموارد و قيود التوافر و الأسبقية بعين الاعتبار.النموذج مستوحى من مختبرجامعة تلمسان و الذي يحتوي على أربعة آليات و ثلاثة وظائف في حين أن الموارد تكون مستهلكة من طرف آلة مخصصة.تهدف مذكرتنا الى اقتراح وظيفة مصممة ببرنامج Matlab و التي تسمح بحساب الـ« Makespan » مع أخذ الاعتبار القيود المقترحة. ان قواعد الأولوية : SPT, LPT, SRC, LRC تستخدم لتقييم أدائها لمشكلتنا. و قد أجرينا العديد من التجارب للتحقق من صحة الأساليب من جهة، وظيفة الحساب من جهة أخرى.